CROSSLIGHT
Software Inc.

Crosslight Device Simulation Software

PARS OPTICS GROUP

Crosslight Device Simulation Software General Manual

Trademark Notice

Copyright Notice

PARS OPTICS GROUP

Contents

PARS OPTICS GROUP

Part I

OVERVIEW

PARS OPTICS GROUP

Chapter 1

INTRODUCTION

1.1 How to use this manual

If you have not yet installed the software, you should read Chap. 2 and then install the software. Afterwards, you should read Chap. 3 to get started.

If you already have experience with the software and want to know more about the models implemented in the packages, please go to the relevant chapters in the manual:

- Part I contains a rough overview of how the software works and how to fix common convergence problems. We strongly urge all new users of the software to read these chapters thoroughly.

- Part II gives a description of the physical and numerical models common to all our device simulators and is recommended for all users.

- Part III gives a description of certain physical and numerical models that are used only in specific device simulators. Users of our other tools can safely skip the chapters that do not concern them.

- Part IV contains product-specific tutorial examples. Since some concepts re-occur in various devices, users are encouraged to read though all the tutorials: even if an example cannot be run, it can still provide valuable insight into the software.

- Part V explains the various commands used in our software and their syntax. All users should refer to it as needed.

1.2 What is Crosslight Software ?

Crosslight Software provides a number of technology computer aided design (TCAD) simulation packages: PICS3D, LASTIP and APSYS. These packages are designed to simulate semiconductor electronic or optoelectronic devices. We shall also refer to these simulation software packages as **simulators**. It also provides growth process modeling tools (CSUPREM, PROCOM) which are beyond the scope of this document.

Crosslight simulators are based on finite element analysis in 2/3 dimensions. They involve a large number of sophisticated physical and numerical models; our purpose here is to give a complete description of these models.

1.3 What is APSYS ?

APSYS is a general purpose 2D/3D modeling software program for semiconductor devices. Based on finite element analysis, it includes many advanced physical models such as hot carrier transport, heterojunction models and thermal analysis. APSYS offers a very flexible and simulation environment for modern semiconductor devices.

1.3.1 Applications

APSYS has a wide range of applications and can handle almost all semiconductor devices:

- 1. Silicon MOSFET, bipolar transistors and CCD.
- 2. HBT based on SiGe, AlGaAs and InGaAsP.
- 3. GaAs MESFET and Photodectors.
- 4. GaN HEMT.
- 5. Light Emitting Diodes (LED).
- 6. Electro-absorption modulators.
- 7. Organic semiconductor devices (OLED).
- 8. Compound, thin film and multi-junction Solar Cells.

The main restriction of APSYS is that it does not include the photon rate equation necessary for laser modeling. As such, it is a complimentary tool to LASTIP and PICS3D which are specifically designed for laser simulations.

1.3.2 Capabilities

APSYS can solve self-consistently the hydrodynamic equations, the heat transfer equations as well as the conventional drift-diffusion equations. Data generated by APSYS include the following:

- 1)Current versus voltage (I-V) characteristics.
- 2)2D potential, electric field and current distributions.
- 3)2D distributions of electron and hole concentrations.
- 4)2D distributions of hot carrier temperatures in the hydrodynamic model.
- 5)2D distributions of lattice temperature for the heat transfer model.
- 6)Band diagrams under various bias conditions.
- 7)Results of AC small signal analysis for any frequency range. Extraction of 2-port AC parameters such S- and Y- parameters.
- 8)Quantum well subband structure with valence mixing model for quantum devices.
- 9)2D distributions of occupancy and concentration of deep level traps in a semiconductor.
- 10)2D optical field distribution for photonic devices such as photodetectors.
- 11)Spontaneous emission spectrum as a function of current for LED.
- 12)All of the above as a function of time (transient model).
- 13)All of the above at different environment temperatures.

1.4 What is LASTIP ?

LASTIP (LASer Technology Integrated Program) is a powerful device simulation program designed to simulate the operation of a semiconductor laser in two dimensions (2D). Given the structural and material properties, it produces a large amount of simulation data to describe the lasing characteristics.

Based on well established physical models, it provides the user with a quantitative insight into various aspects of a semiconductor laser. It can be used as a computer aided design (CAD) tool to optimize existing lasers or to assess new designs. With the physical models and advanced capabilities of LASTIP, the user can concentrate on device optimization and design while leaving all the numerical modeling work to the computers.

1.4.1 Applications

LASTIP can be used to model the electrical and optical behaviors of semiconductor lasers on a 2D cross section. Most cavity designs are supported and there are no restrictions on the emission wavelength or material composition, provided that the material parameters are known.

Since it is limited to a 2D analysis, LASTIP should only be used to model devices with little longitudinal variation. In practice, this means Fabry-Perot lasers.

1.4.2 Capabilities

LASTIP solves the appropriate differential equations for both quantum well and bulk semiconductor lasers. LASTIP can be used to analyze a large number of physical variables. These include (but are not limited to) the following:

- 1) Light versus current (L-I) characteristics.
- 2) Current versus voltage (I-V) characteristics.
- 3) 2D potential, electric field and current distributions.
- 4) 2D distributions of electron and hole concentrations.
- 5) Band diagrams under various bias conditions.
- 6) Quantum well subband structure with valence mixing model.
- 7) 2D distributions of occupancy and concentration of deep level traps in a semiconductor.
- 8) 2D optical field distribution.
- 9) 2D local optical gain distribution.
- 10) Modal gain spectrum as a function of current.
- 11) Spontaneous emission spectrum as a function of current.
- 12) Far-field distribution.
- 13) All of the above as a function of time (transient model).
- 14) All of the above at different temperatures.

1.5 What is PICS3D ?

PICS3D (Photonic Integrated Circuit Simulator in 3D) is a three dimensional (3D) simulator for laser diodes and related waveguiding photonic devices/circuits.

Based on 3D finite element analysis, it solves the semiconductor and optical wave equations to provide an accurate description of the device characteristics. When calibrated with specific material/process, it can be used as a computer-aided design tool to optimize existing devices or to assess new designs.

1.5.1 Applications

PICS3D is designed to simulate a variety of semiconductor optoelectronic devices including the following:

- 1. Fabry-Perot (FP) lasers.
- 2. Distributed Feedback (DFB) lasers.
- 3. Distributed Bragg Reflector (DBR) lasers.
- 4. Semiconductor Optical Amplifiers (SOA).
- 5. Waveguide photodetectors.
- 6. Vertical Cavity Surface Emitting Lasers (VCSELs).
- 7. External cavity lasers.
- 8. Fiber grating lasers.
- 9. Electrode absorption modulators (EAM).
- 10. Multisection/Multielectrode DFB or DBR lasers.
- 11. Multisection photonic integrated circuit combining more than one of the above devices.

Note that FP lasers can also be modeled with LASTIP so there is an overlap in modeling capabilities. There are important differences though: the 2D model of LASTIP is quicker and easier to converge than PICS3D but does not include any longitudinal variation effects. LASTIP will also assume lasing occurs at the modal gain peak where as PICS3D considers the position of the longitudinal cavity modes.

It is up to the user to determine which of these tools best suits their needs.

1.5.2 Capabilities

PICS3D is a provides a full 3D analysis by coupling multiple 2D cross-sections of a waveguiding device with a round-trip gain equation based on the propagation in the longitudinal direction. Both quantum well and bulk semiconductor lasers can be modeled. Advanced quantum well models (such as k.p theory) are implemented, which allows the user to model strained quantum wells. The semiconductor materials implemented include ternary/quaternary III-V materials, nitrides and many others: additional materials can be defined as needed by the user.

The capabilities of simulation include the following:

- 1) Light versus current (L-I) characteristics.
- 2) 3D potential, electric field and current distributions.
- 3) 3D distributions of electron and hole concentrations.
- 4) Band diagrams under various bias conditions.
- 5) Quantum well subband structure with valence mixing model.
- 6) 3D distributions of occupancy and concentration of deep level traps in a semiconductor.
- 7) 3D optical field distribution.
- 8) 3D local optical gain distribution.
- 9) Full multiple mode emission spectra at different power levels.
- 10) Lasing wavelength, output power and longitudinal photon density distribution as a function of bias current.
- 11) Characteristics of DFB lasers with spatial and spectral hole burning effects.
- 12) Full multi-mode simulation of DFB lasers.
- 13) Relative Intensity Noise (RIN), Frequency Noise (FM) and spectral linewidth under different bias conditions.
- 14) Static tuning and dynamic modulation characteristics of single- and multi-electrode DFB or DBR lasers.
- 15) Second harmonic distortion in a laser system under direct current modulation.

Chapter 2

INSTALLATION

2.1 System Requirements

The minimum system requirements are a Pentium 4 CPU (or equivalent) with 256 MB of memory, at least 200 MB of available hard disk space and 250 MB of swap space. The recommended configuration has at least 1 GB of RAM to limit the use of the slower virtual memory/swap space.

For large-scale simulations and/or very dense meshes, you may need to run a 64-bit version of the software in order to access sufficient memory. This will require both a 64-bit capable CPU and a 64-bit operating system. In these situations, you may also benefit from using the multi-CPU solvers to speed up certain calculations. Please consult Crosslight for details.

2.2 Installation

The installation program will guide you through the installation process. Version-specific instructions may come with your installation CD or download. After installation, the program may be launched from the Start menu.

Note that unless specifically instructed to do so, you should not install multiple software packages in the same directory. This may cause version errors between shared files.

If you are using a version of the software controlled by a dongle key, you also need to install the dongle driver at this time. The driver file should be included on your installation CD.

If you are using a remote license, please make sure that your computer has access to the internet. Some of the programs will need to access a server in our office to

validate your license so make sure to configure your firewall appropriately. Consult your IT department or Crosslight for assistance.

2.3 Additional Software

By default, Crosslight uses Gnuplot (http://www.gnuplot.info/) to view simulation results. This is done by generating Postscript (.ps) files: we recommend installing Ghostview (http://pages.cs.wisc.edu/~ghost/) if you do not have a viewer for .ps files.

Results may also be plotted without additional software by using our CrosslightView program.

Chapter 3

BASIC FILES AND PROCEDURES

3.1 For the Impatient

To start the Crosslight software, (supposing you are using LASTIP; APSYS and PICS3D follow the same routine), click on `Start -> Programs -> Lastip_20xx.xx`. The Simucenter window will pop up, which is the main Graphical User Interface of Crosslight Software: most simulation tasks can be accomplished from within this interface. Depending on the software you are using, this window may be labeled SimuLASTIP, SimuAPSYS or SimuPICS3D: we will use SimuCenter as a common name for all these programs.

At this point, any problems with your license will stop the program. If this should occur, contact the Crosslight support staff for assistance.

We have produced several introductory movies to help users get started. Please click on HELP from the Simucenter main menu and select "Movies: Getting Started". Have fun watching this and the other included movies !!!

After watching the movies, you may want be tempted to start modeling your own devices right away. However, we strongly urge you to take the time to work through the included tutorial examples to learn how to use the software. No matter how eager you are, the learning curve can be quite steep and you will benefit in the long run from a slower approach.

Even after you work on the examples, it is preferable to use an existing example as a basis for your own simulations. Whenever possible, you should modify existing designs step by step to fit your needs rather than make an entirely new simulation from scratch.

To run an existing simulation (such as the tutorial examples), click on File->Open

Project from within Simucenter to browse and select an existing project in the examples directory. Load the project by clicking on the .sol, .gain or .spj file. To act on the input files, right-click and select the appropriate action from the pop-up menu:

1. Run the "*.layer" files to 'process and generate the .geo files.
2. Run the "*.geo" files to generate the mesh.
3. (Optional) Run the "*.mplt" files to inspect the mesh generated if you wish to know if the mesh is reasonable.
4. (Optional) Run "*.gain" files to preview the gain spectrum or other critical physical variables. If no .gain files exist, you can create a basic file by right-clicking the .mater file.
5. Run the main equation solver with the "*.sol" input file .
6. (Optional) Analyze and plot the results with the " *.plt" input files.
7. (Optional) Results can also be plotted by using CrosslightView. This can launched by right-clicking on the .std files in SimuCenter or with the main menu: "Action → View Results → CrosslightView". The program can also be launched independently from the Start Menu.

3.2 How Crosslight Simulation Program Works

A simulation is controlled by one or more input files which define the device structure and the simulation parameters. There are three basic input file types with different file extensions: .layer, .sol and .plt . These are used (respectively) to input the device structure/generate the mesh, solve the equations and plot the results. In some cases, intermediate input files are also needed for the simulation; this will be described below.

Each input file contains a collection of statements recognized by the program; these statements allow the user to interact with the program. Although the input files have been designed to be processed by a stand-alone command-line program, Crosslight has also created a number of user interface programs to facilitate the creation of these input files; these are also described in the following sections.

3.2.1 Input / Output Files

The overall file structure is show in Fig. 3.1. As we discussed previously, the main input files are .layer, .sol and .plt. There are also other auxiliary input/output files

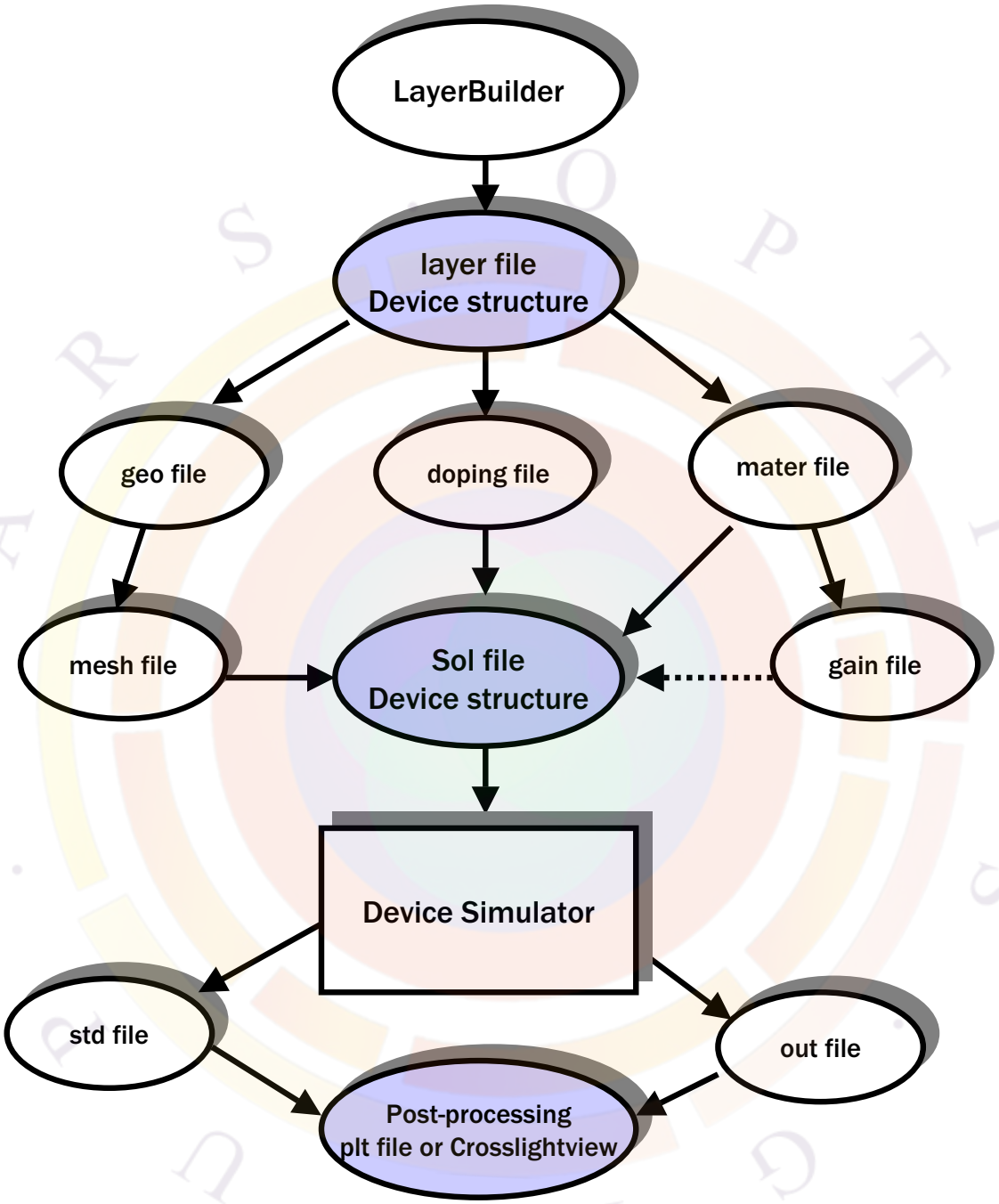


Figure 3.1: Overall file structure and flow chart of Crosslight device simulators

that the user needs to know in order to use the simulation software effectively. We give a detailed description of the most important ones below:

1. .geo. The main input file that describe the full details of the device geometry and the initial mesh allocation.
2. .sol. The main input file that defines the material properties and controls the bias and other conditions of main equation solver.
3. .layer. An important auxiliary input file that uses the layer structure description to generate the .geo, .doping and .mater files,
4. .doping which contains doping information that is to be included in the .sol file.
5. .mater which contains material information that is to be included in the .sol file.
6. .mplt can be used to plot the mesh generated from .geo file.
7. .gain file is another important auxiliary input file that can be used to preview the optical gain spectrum, spontaneous emission rate spectrum, quantum well subbands, and other critical physical properties. This may be used by the user to do some preliminary estimates before the full simulation is performed.
8. .out files may appear as .out_0001, .out_0002, etc... These are numerical output data from the main equation solver. They can be used by the .plt program to be plotted. These output files are not meant to be understood by the user.
9. .std files may appear as .std_0001, .std_0002, etc.. These are another form of numerical output data from the main equation solver. They can be used by the CrosslightView program to be displayed in 3D color graphics.
10. .plt file is used to plot the data in .out output files. The program calls the public domain software GNUPLOT to display the graphics in various computer platforms and printers. Typically, Postscript files (.ps) are used as the GNUPLOT output.

3.2.2 The direct approach

Since the core simulation software interacts directly with the input files (.layer, .sol and .plt), the direct approach is to directly edit these files.

While at first, a basic input file may be created with one of the setup tools provided, direct control has several advantages for advanced users:

1. Allows the user to use the full capabilities of the software.
2. Can create of new simulations by modifying an older example or merging parts of different examples.
3. Easier to reproduce problems when technical support is required. If an error is encountered, the basic input files can simply be emailed for assistance.
4. Text files are portable across systems: being able to use the direct approach means not having to rely on setup tools which may not be available on other computer platforms.

3.2.3 Input Statement Syntax

To modify the input files, it is helpful to grasp the basic syntax used for statements:

- Comment lines start with " $" and will be ignored during file processing.
- Each line can only have a maximum of 80 characters and any information after this limit will be ignored during processing.
- Statements that need to be split up due to their length should use the continuation symbol "&&". Characters after the continuation symbol will be ignored.
- Each line can only contain one statement.
- Each statement starts with a keyword which may or may not be followed by parameters specified with "=". Some parameters may have a default value and do not need to be specified while others are mandatory and must be defined by the user.
- Each parameter has its own type (character string, integer or floating point value). Using a value of the wrong type is an error.
- Certain values need to be grouped for input (e.g. a set of (x,y) coordinates). The following symbols are invisible to the parser used for processing and may be used to group data: space, " ,", " (", ")"," [", "]", " {" and " }".
- Certain statements (esp. those used to describe material properties such as **band_gap**) can be used more than once. If so, then the last value issued takes precedence.

In some cases, a formula or table may be used instead of defining a value for a material parameter. Please consult Part V and Appendix B for detailed information on the syntax of statements and parameters as well as rules for mathematical functions that can be used in the input.

3.2.4 The Setup Utility Programs

At first, creating the main input files (.layer, .sol and .plt) from scratch may be difficult because of the need to learn the syntax and the large number of parameters that need to set to define a simulation.

However for many structures, it is possible to create a simplified input file where fewer parameters will need to be adjusted to meet user requirements. The command-line setup programs allow you do this:

1. SetupLayer prompts the user to enter material layer thickness, doping and material composition parameters. It will generate the .layer file which may then be used to generate .geo, .doping and .mater

2. SetupApsys, SetupLastip, or SetupPics3d can be used to produce the .sol, .gain and .plt input files.

3.2.5 The graphic user interface

In addition to the command-line setup programs, Crosslight has developed GUI tools to facilitate the process of creating input files and running a simulation.

At the time of printing of this manual, the following tools are available:

1. SimuCenter: at its heart, a simple text editor for the input files. However, it also integrates various device setup utilities and graphic input/output programs and can launch the main solver. It also includes online help capabilities as well as a "Wizard" to assist with use of the various commands. This should be the starting point of any simulation.

2. LayerBuilder which generates layers of material used to build a device. Reads and creates .layer files to define simple device structures.

3. GeoEditor which allows the user to draw complex devices with irregular shapes that are compatible with the .geo input file format. Should be used instead of LayerBuilder when the device structure does not follow the typical layer/column pattern found in most devices.

4. CrosslightView. A 3D color graphic display tool using the OpenGL technology. Compatible with the .std file format.

Please refer to the online manuals for specific help on these programs. They can be accessed from the "Help" menu in SimuCenter.

3.3 Defining the device structure

Although the main simulator recognizes only the finite element mesh, it is important to correctly define the device structure first. The basic input file used for this is the ".geo" file. It provides a versatile and flexible geometric data specification for finite element analysis. Virtually any device geometry or material variation may be handled by the ".geo" data format. However, it relies on absolute point coordinates to define polygon corner points so it may be tedious to use this format for simple devices.

Therefore, we have created a separate program called LAYER to help generate the ".geo" file easily. The LAYER program is controlled by a simpler input file with file extension ".layer". The .layer specification allows simultaneous definition of the device geometry, doping, material information and mesh parameters.

The following subsections describe the ".geo" and and ".layer" input file formats. Serious users of the simulator should study the ".geo" input file format. For beginners or casual users, ".layer" input file may be sufficient.

In this section, we will also show how to use some of the command-line tools to define and process these input files. Note that these tools can also be accessed via SimuCenter by right-clicking on the input file and selecting the appropriate action. We will also use LASTIP for this example but the process is essentially the same for APSYS and PiCS3D.

3.3.1 SetupLayer

To create a basic .layer file, we can use the setuplayer command-line program. Let us illustrate its use by using this very simple 1D laser diode structure:

1 μm	$Al_{0.5}Ga_{0.5}As$	p=1e24 m^-3
0.2 μm	GaAs	undoped, active region
1 μm	$Al_{0.5}Ga_{0.5}As$	n=1e24 m^-3

Please note that when we enter the layer structure, we start from bottom as layer #1 and work our way up until the top layer is reached.

The program is called as follows:

```
C:\Work>c:\crosslig\lastip\setuplayer.exe -layer test1


 Setting up file:test1.layer
```

```
 We assume a 1D layer structure on the xy-plane.

 Please enter the device width (um)
 (Or half width if it is a symmetric device)
 [typical:1-5 um]
1.5

 Please enter the total number of mesh points
   for all the layers;  typical [20-40]
30
 Enter the maximum mesh spacing:
   typical [0.1 - 0.5 um]
0.25
   Please enter the layer structure starting from
   bottom as layer 1

 For layer number           1

 Please enter layer thickness (um)
1

 Please select doping (1/m^3):
 n-doping: (1) 2.0e24; (2) 1.5e24
           (3) 1.0e24; (4) 0.5e24; (5) other n-doping
           (6) undoped
 p-doping: (7) 2.0e24; (8) 1.5e24
           (9) 1.0e24; (10) 0.5e24; (11) other p-doping
3

 Is there DFB/DBR grating structure in this layer? [y/n]
n

 Is this layer an active layer (y/n)?
n

 Please enter layer bulk material macro to be used
  (1) gaas -- bulk GaAs
  (2) inp -- bulk InP
  (3) ingaas -- bulk InGaAs
  (4) algaas -- bulk AlGaAs
  (5) ingaasp -- bulk InGaAsP (matched to InP)
  (6) ingaasp_xy -- bulk InGaAsP (strained)
  (7) Enter another macro name other than the above
```

```
4
 Do you wish to grade Al [y/n] ?
n
 Please enter the Al composition [0-1]
0.5
 Next?
   (1) next layer
   (2) go back to a previous layer
       (so that I can re-enter parameters)
   (3) repeat the previous few layers
       (as in a MQW device)
   (4) finish.
   (5) cancel without generating input file.
1

 For layer number           2

 Please enter layer thickness (um)
0.2

 Please select doping (1/m^3):
 n-doping: (1) 2.0e24; (2) 1.5e24
           (3) 1.0e24; (4) 0.5e24; (5) other n-doping
           (6) undoped
 p-doping: (7) 2.0e24; (8) 1.5e24
           (9) 1.0e24; (10) 0.5e24; (11) other p-doping
6

 Is there DFB/DBR grating structure in this layer? [y/n]
n

 Is this layer an active layer (y/n)?
y

 Please enter active layer material macro
   quantum subbands, gain, etc.

 Please enter the active region material system
 (1) AlGaAs/AlGaAs --- quantum well
 (2) InGaAs/AlGaAs --- strained quantum well
 (3) InGaAsP/InP --- InGaAsP/InGaAsP strained
     quantum well/barrier; substrate=InP.
 (4) AlGaAs --- bulk AlGaAs
```

```
 (5) InGaAsP --- bulk InGaAsP lattice matched to InP
 (6) InGaAsP/GaAs --- InGaAsP/InGaAsP strained
     quantum well/barrier; substrate=GaAs.
 (7) InGaAlAs/InP  --- InGaAlAs/InGaAlAs strained
     quantum well/barrier; substrate=InP.
 (8) GaN --- bulk GaN (9) AlGaN --- bulk AlGaN
 (10) InGaN --- bulk InGaN
 (11) InGaN/AlGaN -- strained well/barrier;
      substrate = GaN.
 (12) InGaAlP/GaAs --- In(1-xw-yw)Ga(xw)Al(yw)P
      /In(1-xb-yb)Ga(xb)Al(yb)P; substrate=GaAs
 (13) InGaN/InGaN --- Strained well grown on material
      lattice matched to GaN.
 (14) Active macro other than the above
4
 Enter Al composition (0 to 1)
0

 Please enter layer bulk material macro to be used
  (1) gaas -- bulk GaAs
  (2) inp -- bulk InP
  (3) ingaas -- bulk InGaAs
  (4) algaas -- bulk AlGaAs
  (5) ingaasp -- bulk InGaAsP (matched to InP)
  (6) ingaasp_xy -- bulk InGaAsP (strained)
  (7) Enter another macro name other than the above
4
 Please enter the Al composition [0-1]
0
 Next?
   (1) next layer
   (2) go back to a previous layer
       (so that I can re-enter parameters)
   (3) repeat the previous few layers
       (as in a MQW device)
   (4) finish.
   (5) cancel without generating input file.
1

 For layer number            3

 Please enter layer thickness (um)
1
```

```
 Please select doping (1/m^3):
 n-doping: (1) 2.0e24; (2) 1.5e24
           (3) 1.0e24; (4) 0.5e24; (5) other n-doping
           (6) undoped
 p-doping: (7) 2.0e24; (8) 1.5e24
           (9) 1.0e24; (10) 0.5e24; (11) other p-doping
9

 Is there DFB/DBR grating structure in this layer? [y/n]
n

 Is this layer an active layer (y/n)?
n

 Please enter layer bulk material macro to be used
  (1) gaas -- bulk GaAs
  (2) inp -- bulk InP
  (3) ingaas -- bulk InGaAs
  (4) algaas -- bulk AlGaAs
  (5) ingaasp -- bulk InGaAsP (matched to InP)
  (6) ingaasp_xy -- bulk InGaAsP (strained)
  (7) Enter another macro name other than the above
4
 Do you wish to grade Al [y/n] ?
n
 Please enter the Al composition [0-1]
0.5
 Next?
   (1) next layer
   (2) go back to a previous layer
       (so that I can re-enter parameters)
   (3) repeat the previous few layers
       (as in a MQW device)
   (4) finish.
   (5) cancel without generating input file.
4
 Generating .layer file

 For your convenience, the setup parameters you have just
 typed in have been saved in a file named setup_script.txt.
 You may modify the script file and use it
 (to avoid typing the same parameters all over again)
```

```
 when you run the SetupLayer program next time, as follows:
 [path]\setuplayer.exe -vlayer mydevice < setup_script.txt
 This may be launched from the SimuCenter or from DOS prompt
```

This is the resulting file (test1.layer):

```
$file:test1.layer
begin_layer
column column_num=1 w=  0.150000E+01 mesh_num=2  r=1.
bottom_contact column_num=1 from=0 to=  0.150000E+01 &&
  contact_num=1 contact_type=ohmic
$
layer_mater  macro_name=algaas &&
  var_symbol1=x var1=  0.500000E+00 &&
  column_num=1
layer d=  0.100000E+01 n= 13 &&
  n_doping1=  0.100000E+25 &&
  r=  0.800000E+00
$
layer_mater  macro_name=algaas &&
  var_symbol1=x var1=  0.000000E+00 &&
  active_macro=AlGaAs &&
  avar_symbol1=xw avar1=  0.000000E+00 &&
  column_num=1
layer d=  0.200000E+00 n=  5 &&
  shift_center= -0.111111E-01 &&
  r= -0.120000E+01
$
layer_mater  macro_name=algaas &&
  var_symbol1=x var1=  0.500000E+00 &&
  column_num=1
layer d=  0.100000E+01 n= 11 &&
  p_doping1=  0.100000E+25 &&
  r=  0.120000E+01
$
top_contact column_num=1 from=0 to=  0.150000E+01 &&
  contact_num=2 contact_type=ohmic
end_layer
```

We will explain the .layer file in the following subsection.

3.3.2 Growing the layers: .layer files

The ".layer" format is a simplified way to define the device structure. It is meant to be processed and generate the .geo file that is required for mesh generation. It also produces other input files that will be used for the main solver (.mater and .doping). From the command-line, this processing is done with the layer.exe program:

```
C:\Work>c:\crosslig\lastip\layer.exe test1.layer
```

Note that if you are using the SimuCenter GUI, this can also be accomplished by right-clicking on the .layer file and choosing the "Process ..." option.

The .layer format is based on the idea that most devices are grown by depositing several parallel layers on top of each other. Extra features such as etching and regrowth can be modeled by allowing lateral variation of the materials through the use of multiple columns.

These layers and columns define the grid lines that serve as polygon boundaries in the .geo file. The .layer format simplifies the .geo definition since it automatically numbers the polygons and calculates the corner point's absolute coordinates. The only thing required is knowledge of the the layer thicknesses and column widths.

Before proceeding any further, let us look at the structure of the .layer file we are using as an example. The relevant file contents are bracketed by **begin_layer** and **end_layer**. To process layers, the program needs to know how many columns are involved so the next set of commands will be the column statements: columns must be numbered 1,2,3,... starting from the left. Layers are not numbered: they are added automatically from the bottom up as the .layer file is processed from beginning to end.

column statements define the width of the columns and also control the lateral mesh allocation. In this particular case, we only have one column with 2 mesh points so there is no real lateral variation allowed: we refer to this as a 1D simulation. This is the simplest kind of device we can build and also the quickest to model.

As each layer is being processed, there must be one **layer_mater** statement for each column to define the material. For example, to define a ridge or regrowth, the material in column 2 would be different than in column 1. Material numbers in .geo are generated automatically when new macro definitions are found and assigned to the relevant polygon.

If no material is to be defined in a particular column, use the "void" macro; this will prevent any mesh from being allocated in this region. Note that in some cases, this will affect the physics of the simulation by shifting the position of the boundaries. In that case, the "vacuum" or "air" macro may be used instead; this will define an insulator region so mesh points will be allocated but there will be no current flow.

Active regions can be recognized by the two different macros they define. The reason for this will be explained in more detail in section 3.5.

After the **layer_mater** statements, there must also be a **layer** statement to define the layer thickness as well as the vertical mesh information. In that sense, **layer** is very similar to the **column** statement which provides size and lateral mesh information for the columns.

The **layer** statement also provides doping information with **n_dopingj** or **p_dopingj** where j is the column number. It is also possible to define doping for each column in the **layer_mater** statement but **layer** provides additional tools to linearly grade doping levels. Also, note that doping is defined in m^{-3} for input purposes; care should be taken to convert from the commonly used cm^{-3} units as low doping levels can lead to poor convergence.

The last statements define equipotential boundary regions and may be placed anywhere in the file after the **column** statements. **top_contact** and **bottom_contact** define contact regions in a given column on the top or bottom of the layer stack. Position information defines the extent of the contact with respect to the contact width.

Note that our usual convention is to grow devices n-side down and to use electrode #1 as the n electrode. This choice mostly influences the sign of the current on the electrodes when bias is applied and is for convenience's sake only.

3.3.3 Dealing with polygons: .geo files

This section is for experienced users, beginning uses can safely skip it.

The .layer example above produces this output file (test1.geo) when processed:

```
begin_geometry
point label=a001 xy=[  0.000000000000E+000  0.000000000000E+000 ]
point label=a002 xy=[  0.000000000000E+000  0.100000000000E+001 ]
point label=a003 xy=[  0.000000000000E+000  0.120000000000E+001 ]
point label=a004 xy=[  0.000000000000E+000  0.220000000000E+001 ]
$
point label=b001 xy=[  0.150000000000E+001  0.000000000000E+000 ]
point label=b002 xy=[  0.150000000000E+001  0.100000000000E+001 ]
point label=b003 xy=[  0.150000000000E+001  0.120000000000E+001 ]
point label=b004 xy=[  0.150000000000E+001  0.220000000000E+001 ]
$
polygon name=p001 4_points=[ a001 b001 b002 a002 ] material=  1 &&
  boundary_1=[ a001 b001 ] &&
  limits_1=[   0.000000000000E+000  0.150000000000E+001 ]
```

```
polygon name=p002 4_points=[ a002 b002 b003 a003 ] material=  2
polygon name=p003 4_points=[ a003 b003 b004 a004 ] material=  1 &&
  boundary_2=[ b004 a004 ] &&
  limits_2=[   0.000000000000E+000  0.150000000000E+001 ]
$
end_geometry
begin_meshgen
internal_xpoint xp_size=  0.2000E-03
put_mesh polygon=p001 edge=[ b001 b002 ]  &&
  point_from=  0.000000000000E+000 point_to=  0.100000000000E+001 &&
  number= 13 ratio=  0.800000000000E+000
put_mesh polygon=p002 edge=[ b002 b003 ]  &&
  point_from=  0.000000000000E+000 point_to=  0.200000000000E+000 &&
  shift_center= -0.111111000000E-001 &&
  number=  5 ratio= -0.120000000000E+001
put_mesh polygon=p003 edge=[ b003 b004 ]  &&
  point_from=  0.000000000000E+000 point_to=  0.100000000000E+001 &&
  number= 11 ratio=  0.120000000000E+001
$
put_mesh polygon=p001 edge=[ a001 b001 ]  &&
  point_from=  0.000000000000E+000 point_to=  0.150000000000E+001 &&
  number=  2 ratio=  0.100000000000E+001
$
mesh_output mesh_outfile=test1.msh order=yes
end_meshgen
```

The first half of a .geo file is bracketed by the **begin_geometry** and **end_geometry** statements.

Within this section, point coordinates are assigned labels using the **point** statement. Polygons are then defined by using point labels in counter-clockwise order with the **polygon** statement.

Each polygon is also identified by its material number. Different materials require different polygons so that material boundaries can be modeled accurately. As a rule, an insulator macro must be given a larger material number than a semiconductor.

One major limitation of the .geo format is that it requires that if two polygons touch each other, the shared edges must fully overlap. For complex geometries, it is recommended to sketch out the polygons and point labels to avoid errors caused by this limitation. In certain cases, it may be necessary to sub-divide polygons in order to ensure this rule is always enforced.

Boundary regions occupy part of an edge and can be defined either in the **polygon** or **add_boundary** statements. In our device simulators, these are used to define

equipotential boundaries (i.e. contacts). In the above example, these boundaries have been inherited from the contact definitions in the .layer file.

While polygons can be triangles, trapezoids are preferred since it is easier to refine the mesh afterwards. All .geo files obtained by processing .layer will be based on trapezoids due to its grid format.

For rounded edges, the **edge_curve** statement may be used to add a concave or convex arcs.

The second half of the .geo format deals with mesh allocation and will be discussed in the next section.

3.4 Creating a good mesh

In order to run the simulation, we will need a finite element mesh based on the geometry and material properties of our device. Usually, the mesh data file is given a .msh file extension and it is generated by processing a .geo file. In our example, the mesh file is generated with this command:

```
C:\Work>c:\crosslig\lastip\geometry.exe test1.geo
```

Note that if you are using the SimuCenter GUI, this can also be accomplished by right-clicking on the .geo file and choosing the "Generate ..." option.

This will produce a .msh file that contains mesh point and triangle element information as well as a .mplt file that can used to plot the mesh.

3.4.1 Mesh definition in .layer files

In the previous section, the mesh allocation was done automatically by the setup-player program. To modify the number of mesh points, the mesh_num parameter in **column** and the n parameter in **layer** must be changed. The number of mesh points in this system will be roughly $\sum(mesh\_num) \times \sum(n)$; some extra mesh points may be added at certain interfaces.

To control the distribution of mesh points within a layer or column, use the r and shift_center parameters of **column** and **layer**. These parameters will be part of various **put_mesh** statements in the .geo file when the .layer file is processed. For a more detailed explanation of these parameters, consult the reference section for the **put_mesh** statement.

3.4.2 Mesh definition in .geo files

This section is for experienced users, beginners may safely skip it.

The second half of a .geo file controls the mesh generation and is bracketed by the **begin_meshgen** and **end_meshgen** statements.

The basic statement for this section is the **put_mesh** statement which directly allocates the mesh along the edges of a polygon. If your .geo file is a converted .layer file, these statements will be initialized with the mesh parameters from the **column** and **layer** statements. Note that just like in .layer, the distribution of mesh points along an edge can be controlled.

To mesh the inside of the polygons, mesh lines will automatically propagate from one side of a polygon to its opposite edge and continue on to neighboring polygons until termination. For triangles, it assumed that each edge is opposed to the other two but trapezoid edges oppose each other in pairs. Crossing mesh lines are then sub-divided into mesh triangles.

It is therefore not necessary to mesh all polygon edges as the mesh lines propagate automatically. Conflicting or redundant **put_mesh** statements will be overridden and the last statement issued will have precedence. Note that un-meshed edges will also be assigned a default minimum mesh when detected; this should be shown as a warning message when the .geo file is processed.

It is recommended that the user practice setting up a simple mesh to get used to the mesh generator before attempting a more complicated design.

Optionally, you can also use **double_mesh** to double the mesh density in a specified region of a polygon or **half_mesh** to reduce it. This gives users additional control on the local mesh density without over-meshing the entire device.

3.4.3 Troubleshooting mesh

An unsatisfactory mesh is a major cause of non-convergence. The first step to troubleshooting a mesh is plot it as shown above and check that the mesh distribution is OK.

The next step is to check if the mesh is too coarse. While it can be tempting to put fine mesh everywhere in a device, this is not practical. For example, if the mesh is too dense, the simulation time may be prohibitively long. Also, the memory requirements may go beyond what the mesh generator supports or even beyond what your computer allows. The latter is unlikely in a 1D/2D simulation but often occurs with 3D simulations on 32-bit systems.

Therefore, we must make an effort to allocate mesh points intelligently and concentrate them only where needed. In general, the mesh must be dense near sharp

material interfaces (e.g. contacts, boundaries between different materials or different doping levels, etc...) and near other regions where the electrical properties change rapidly over a small distance (e.g. areas where there is current crowding, tunnel junctions, etc...).

There is no systematic error analysis that can help you determine if a mesh is dense enough. It is recommend that you plot available simulation results to ensure that material properties and important quantities like the band diagram are well-sampled.

However, there is a **refine_mesh** statement that can assist you in this task. This command can add mesh points anywhere the change in material parameters between neighboring mesh points exceeds a specified amount. This will generate a new mesh file that can be used in future simulations.

Additional discussions on this topic can be found in Chap. 4.

3.5 Material parameters

3.5.1 Material Macros

In the Crosslight system, many material parameters are needed to model the physical behavior of devices. Each of these parameter is defined through a separate statement such as **band_gap**. For convenience, these parameters are lumped together into a "macro". In this context, a "macro" is a groups of material parameters commands lumped together under the same name: this makes it easy to set all of the material properties for a given material using a single command.

In order to be used by the software, each material property must be assigned to the right mesh points. This is done by using material numbers (1,2,3,...) which are defined along with the mesh polygons in the .geo file. Note that we do the same for contacts to facilitate the application of bias to the simulation: voltage_1 or current_1 refers to bias on electrode # 1.

The link between the material numbers and the macro name can be seen in the .mater file that is produced when a .layer file is processed:

```
contact num= 1 type=ohmic
contact num= 2 type=ohmic
load_macro name=algaas mater=   1 &&
  var_symbol1=x  var1=  0.5000E+00
load_macro name=algaas mater=   2 &&
  var_symbol1=x  var1=  0.0000E+00
get_active_layer name=AlGaAs mater=   2 &&
  var_symbol1=xw  var1=  0.0000E+00
```

```
active_reg mater=   2 &&
  thickness=  0.200000000000E+000
```

When using the .layer format to input the device layout, material numbers are assigned automatically by layer.exe and transferred to the .geo file. If the .geo format is used to input the device layout directly, the material numbers for each mesh polygon must be defined manually; material macros must also be assigned manually for each material.

In this example, we see that there are two kinds of macros: passive and active. In our convention, passive macros are recognized by their lowercase names and are loaded with **load_macro**. Active macros can have mixed-case names and are loaded with **get_active_layer**.

Passive macros define bulk material properties such as bandgap, carrier mobility and refractive indexes. Active macros define material parameters needed to calculate the optical properties (gain/absorption, spontaneous emission) of this material. Active macros also come in different flavors: bulk, QW and complex QW. The differences between the different models are discussed further in Sec. 8.1.

Typically, only the active regions of light-emitting devices need to use active macros: the optical transitions in other layers can safely be assumed to be at wavelengths that are not of interest. The main reason to use active macros in a certain region is that the light emission from that layer is of great interest to the device simulation: for example the active region of a laser device or LED controls almost all of the interesting device properties.

An exception to the above rule is regions where quantum confinement occurs (e.g. HEMTs or certain MOSFET channel designs) and a 2D electron gas (2DEG) is expected. In that case, the active macro for a QW must be used in the channel as the active macros enable the quantum-mechanical solvers. This is purely a historical accident as Crosslight tools were originally designed to model laser diodes; new users may find it beneficial to keep this tidbit of information in mind as it explains many of our design choices.

The actual parameters defined in a macro are outside the scope of this section. However users who wish to examine the content of the default macros should look to their installation directory (default is c:\crosslig). The default macros are contained in the crosslight.mac (silicon, GaAs and most III-V compounds) and more.mac (nitrides, organic semiconductors, metals) files. A few sample macros are also included in Appendix B.

Additional help regarding macro syntax is included in the header of these files. This can be of help to advanced users who wish to create new material definitions or who have access to better material data and want to override the default material parameters. However, it is ***STRONGLY*** recommended that the default macro files not be altered in any way since that would affect all the simulations that use these

files.

Instead, the **use_macrofile** statement can be used to load a custom or modified macro file for use in a simulation. Material parameters can also be overwritten directly in the simulation file by re-issuing the statement that defines it once the macros have been loaded.

Simplified Complex QW Library (SCXLIB)

Starting with the 2014 version of the software, a simplified model is being introduced to define material parameters. We have found that new users often make mistakes when invoking the passive and active material macros separately and sometimes define conflicting sets of material parameters: for example, the QW active macro declaration may define a certain barrier composition but the barrier layers themselves have another.

To avoid this common problem and ease the learning curve of the software, we have introduced the concept of a "library" which is simply a wrapper around the "macro" concept. When a material is declared with a library, a single set of input parameters is given in the .layer file. The file is then processed and macro invocations are adjusted to correctly send the parameter values to the right variable names.

The definition of quantum well regions is also simplified using the library system. In the .layer file, each quantum well layer is simply tagged (**model**=*quantum_well*) as follows:

```
layer_mater column_num=1 mater_lib=AlGaAs &&
  var_symbol1=x grade_var=1 &&
  grade_from=0.71 grade_to=0.33
layer d=0.15 n=10 r=1

layer_mater column_num=1 mater_lib=AlGaAs &&
  var_symbol1=x grade_var=1 &&
  grade_from=0.33 grade_to=0.0 &&
  model=quantum_well
layer d=0.001 n=5 r=1

layer_mater column_num=1 mater_lib=AlGaAs &&
  var_symbol1=x var1=0.  model=quantum_well
layer d=0.005 n=25 r=1

layer_mater column_num=1 mater_lib=AlGaAs &&
  var_symbol1=x grade_var=1 &&
  grade_from=0.0 grade_to=0.33 &&
```

```
  model=quantum_well
layer d=0.001 n=5 r=1

layer_mater column_num=1 mater_lib=AlGaAs &&
  var_symbol1=x grade_var=1 &&
  grade_from=0.33 grade_to=0.71
layer d=0.15 n=10 r=1
```

The software will automatically look to the neighboring layers without the quantum well tag to define the outer barrier regions; if multiple neighboring layers are tagged as a quantum well, then they all belong to the same quantum-confined region. This system thus automatically uses complex QW macros for the tagged layers (and outer barriers) to support this kind of aggregation. The resulting band profile and wavefunctions can be seen in Fig. 3.2.

Note that existing users of the software may continue to use the old macro system indefinitely and that older input files will continue to work as before. For new users however, we strongly recommend the use of this new library method: we will progressively update our tutorial examples in the coming years to reflect this. There is one small caveat though: the "macro" and "library" systems cannot co-exist in the same .layer file.

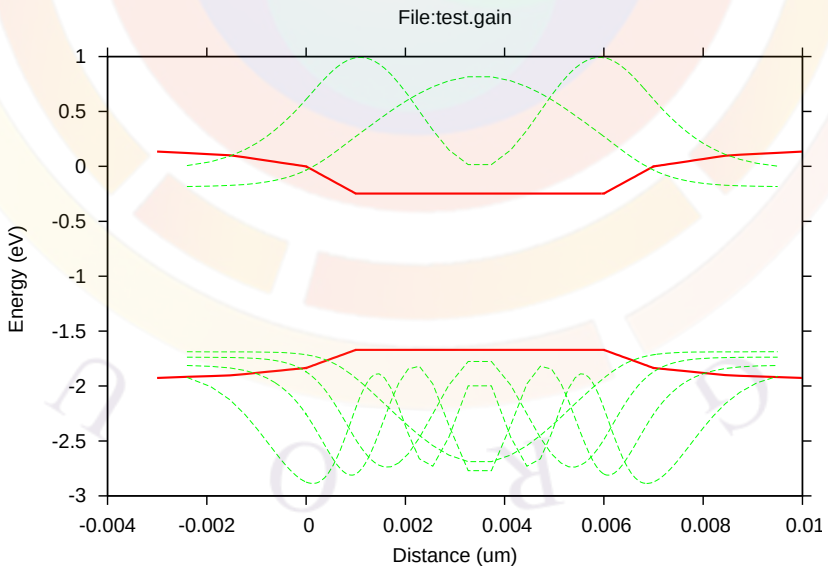


Figure 3.2: A sample complex QW set up using the simplified "library" method.

3.6 Previewing Physical Properties

Before the main equation solver is used it may be a good idea to preview physical properties such as the optical gain/absorption spectrum. For example, this can help confirm that the gain curve is centered on the right wavelength in a laser design.

This can be done with a ".gain" file. A basic input file can be created by right-clicking on the .mater file in SimuCenter or by running the setuplastip/setupapsys/setuppics3d program as follows:

```
C:\Work>c:\crosslig\lastip\setuplastip -gain test1

 Setting up input file:
 test1.gain
 You may use it to plot the gain/loss spectrum
   and other physical properties

 Enter estimated operating wavelength in um
   for example 0.82:
0.82
 Enter carrier density range in m**(-3)
   for example:
   5.e23  1.e25
1e23 5e24
 Enter temperature in degrees Kelvin
300
 Please enter the graphics device
   (1) postscript (2) window (3) x11
 Your choice?
1
 Please select the physical properties to plot
 (1) gain spectrum
 (2) spontaneous emission rate
 (3) index change spectrum
 (4) current vs. active region density
 (5) alpha factor spectrum
 (6) gain vs. active region density
 (7) quantum well subbands (k.p)
 (8) dipole moment vs. k (k.p)
 (9) 1 - 4 above
 (10) 1 - 6 above
 (11) finish
10
```

```
 Enter your next choice (1-11)
11
 Done
```

This will produce the following file:

```
$file:test1.gain
begin_gain
plot_data plot_device=postscript
temperature temp=  0.3000E+03
include file=test1.mater
gain_wavel wavel_range=[  0.8000E+00  0.8400E+00] &&
  conc_range=[  0.1000E+24  0.5000E+25] &&
  curve_number=20
sp.rate_wavel wavel_range=[  0.8000E+00  0.8400E+00] &&
  conc_range=[  0.1000E+24  0.5000E+25] &&
  curve_number=20
index_wavel wavel_range=[  0.8000E+00  0.8400E+00] &&
  conc_range=[  0.1000E+24  0.5000E+25] &&
  init_conc=1.5e24 &&
  curve_number=20
current_conc  conc_range=[  0.1000E+24  0.5000E+25] &&
  data_point=30  &&
  use_macro=yes fit_outfile=tmp.data
alpha_wavel wavel_range=[  0.8000E+00  0.8400E+00] &&
  conc_range=[  0.1000E+24  0.5000E+25] &&
  curve_number=20
gain_density conc_range=[  0.1000E+24  0.5000E+25] &&
  wavel_range=[  0.8000E+00  0.8400E+00] &&
  data_point=30
end_gain
```

The .gain file includes the material definitions of the .mater file through the **include** statement. It also defines the isothermal temperature used for the following calculations. A series of statements then produces a set of curves to preview the physical properties of the first active region found in the .mater file.

To plot these curves, right-click on the .gain file in SimuCenter or use the command line:

```
C:\Work>c:\crosslig\lastip\lastip.exe test1.gain
C:\Work>c:\crosslig\lastip\psplot.bat test1.ps
```

If there is more then one active region defined in .mater that you wish to preview, replace the .mater include statement with the desired active region definition.

3.7 Setting up a simulation

The .sol file is the main simulation input used in Crosslight. It consolidates all the device structural information as well as the material and simulation parameters.

Once again, a basic input file can be created with the command-line setup tools or by right-clicking on an empty .sol file in SimuCenter:

```
C:\Work>c:\crosslig\lastip\setuplastip -sol test1
 Setting up .sol file for filebase=test1

 Please enter the cavity length (um):
200
 Please enter the expected emission wavelength (um):
0.83
 Please enter 1 for 1D laser optical wave boundary
 Please enter 2 for other 2D laser boundary

1
 Please enter the current bias you wish to apply (mA)
100
 Please enter the contact number for the n-type laser contact
 (look up .layer or .geo file if necessary)
1
 Generating .sol file
```

The exact dialogue that will be shown is different for APSYS, LASTIP and PICS3D but the basic principles stay the same. Since we are using LASTIP in this example, we get this .sol file:

```
$file:test1.sol
begin
load_mesh mesh_inf=test1.msh
include file=test1.mater
include file=test1.doping
output sol_outf=test1.out
newton_par damping_step=5. max_iter=100 print_flag=3
use_sor  max_iter=3000 print_sor=noprint
init_wave &&
  length=  0.2000E+03 backg_loss=500. &&
  boundary_type=[2 2 1 1] init_wavel=  0.8300E+00 mirror_ref=0.32 &&
  wavel_range=[  0.8100E+00  0.8500E+00]
equilibrium
```

```
newton_par damping_step=1. print_flag=3
scan var=voltage_1 value_to= -0.1345E+01 print_step=  0.1345E+01 &&
  init_step=  0.2689E+00 min_step=1.e-5 max_step=0.5
scan var=current_1 value_to=  0.2500E+03 print_step=  0.2500E+03 &&
  init_step=  0.2500E+01 min_step=1.e-5 max_step=  0.2500E+02
end
```

There are a few categories of statements that are commonly used in .sol files.

3.7.1 Input/output

In this category, we include all the statements that are used to load previously generated data such as the mesh (**load_mesh**) or merge other input files (**include**) that define the material parameters and doping profiles.

The **output** statement defines a base name that will be used for the output data files. It is generally considered a good idea to used the same base name for input and output files: this is what the command-line setup program does by default.

Using the same base file name ensures that the GUI programs can detect and handle the files correctly.

3.7.2 Physical models

This next set of statements should be located after the input/output statements and varies a lot for each software and indeed, for any given simulation. These statements turn on certain physical models that needed for that particular simulation.

In this example, the **use_sor** statement is used to define a basic 1D mode solver. The **init_wave** statement defines the boundary conditions for the waveguide as well as other critical parameters for the device (length, mirror reflectivity, background losses, etc...).

Depending on the type of simulation, other statements may be used in this section. For example, a solar cell in APSYS would use the **light_power** statement to define the optical pumping conditions.

3.7.3 Bias statements

This category includes the statements that are used to call the Newton solver and find the state of the device under bias. These statements must therefore be preceded by at least one **newton_par** statement to control the solver parameters.

The first of these statements is **equilibrium**. It calls the Newton solver to get the initial solution of the device under thermal equilibrium conditions: this means only the Poisson equation is solved and the net current is zero for both electrons and holes. This is a required first step before bias can be applied.

Subsequent bias is applied with the **scan** statement. The variable controls which electrode is biased as well as the type of bias which is applied: for example, `voltage\_1` is the voltage on electrode #1. Multiple bias variables can be applied simultaneously.

Time is also available as a variable for transient simulations. Note that if you start a transient simulation, it is a good idea to consistently continue to use the time variable as you may otherwise encounter convergence difficulties.

3.8 Running a simulation

To run a simulation, you may right-click the .sol file from SimuCenter or call the main solver from the command line:

```
C:\Work>c:\crosslig\lastip\lastip.exe test1.sol
```

This will produce the following output (truncated for convenience):

```
------------------------------------------------------------

      #         #     #####  #######   ###   ######
      #        # #   #     #    #       #    #     #
      #       #   #  #          #       #    #     #
      #      #     #  #####     #       #    ######
      #      #######       #    #       #    #
      #      #     # #     #    #       #    #
      ####### #     #  #####     #      ###   #

                  Version: 2009.04.01
          Build-date:[y/m/d/h]=2009/ 4/26/14
    Copyright (c) 1995-2008, Crosslight Software Inc.
    Portions of This Software Copyright (c) 1995-2008,
          National Research Council of Canada
                     Licensed to:
                   Evaluation User

               Simulation for Device:
                     test1.sol
```

```
------------------------------------------------------------
Date(d/m/y):          15           5        2009
Time(s/m/h):          11          34          10

======Statement: load_mesh=====

Loading mesh file:
test1.msh
Total mesh points=         58

======Statements of load_macro (band_gap,etc)=====
======Statements of doping=====
======Statements of contact=====

Initializing  parameters.
(Please wait)
--------------Information------------------
Older solver of sor_par has been replaced
  by direct_eigen with default settings.
If you really need to go back to older solver
  please use use_sor instead.
-----------------------------------------

======Statement: output=====


======Statement: equilibrium=====
scan number->           0


Solving equations at equilibrium

Error report for equations and variables:
 it#   eqns       potential
    1  0.440E+04  0.747E+04
    2  0.414E+04  0.326E+04
    3  0.849E+03  0.435E+03
    4  0.112E+03  0.921E+02
    5  0.549E+02  0.217E+02
    6  0.423E+02  0.127E+02
    7  0.256E+02  0.704E+01
    8  0.126E+02  0.318E+01
```

```
     9  0.129E+02  0.365E+01
    10  0.114E+02  0.422E+01
    11  0.301E+01  0.112E+01
    12  0.147E+01  0.459E+00
    13  0.299E+00  0.929E-01
    14  0.107E-01  0.326E-02
    15  0.131E-04  0.388E-05
    16  0.487E-09  0.424E-09
 Calculate optical index change

 Direct eigen solver at lambda=  0.830000000000000
 Select modes with max index:
   3.64984451858962
 Start Arnoldi eigen solver
 sigma=   (763.400219073982,0.000000000000000E+000)
 End of Arnoldi eigen solver
 Eigen Value=[  0.694621E+003  0.140020E+002] Eqn. Error=  0.2731E-09
  Cmplx lateral modal index for z-segment:           1
          1 (3.48172365230206,3.508816922721083E-002)

 Searching for modal gain peak.
 Modal gain (1/m):  -359812.782918322
 At lambda=  0.850000000000000
Active Reg.# 1 Average Conc. (n&p)=  0.7683E+23  0.1429E+24
   Save bias data and continue.
 Lambda(um)=  0.850000000000000
 Lateral mode(s)=           1
 Emitted Power(mW):
 (Mode)      (Total)     (Front)     (Back)
 1            0.4674E-32  0.2337E-32  0.2337E-32

  Solver converged at
Voltage: (Volt)  0.0000E+00  0.0000E+00
Current: (A/m)   0.0000E+00  0.0000E+00


 Data set #           1  printed at
Voltage: (Volt)  0.0000E+00  0.0000E+00
Current: (A/m)   0.0000E+00  0.0000E+00
light:    0.4674E-32

 Printing 2D/3D data to output file
     Data file:test1.out_0001
```

```
 Generating .std file
 Completed .std file

 ======Statement: scan=====
 scan number->           1


 Solving equations with bias.

Changing: voltage_1 with step:   0.2689E+00
 Error report for equations and variables:
  it#   eqns       potential  elec       hole       other
     1  0.182E+02  0.273E+03  0.273E+03  0.272E+03  0.913E-25
     2  0.181E+02  0.164E+03  0.164E+03  0.164E+03  0.588E-25
     3  0.180E+02  0.937E+02  0.937E+02  0.936E+02  0.841E-25
     4  0.178E+02  0.502E+02  0.502E+02  0.501E+02  0.110E-24
     5  0.175E+02  0.249E+02  0.249E+02  0.250E+02  0.129E-24
     6  0.168E+02  0.113E+02  0.113E+02  0.130E+02  0.136E-24
     7  0.153E+02  0.469E+01  0.468E+01  0.634E+01  0.136E-24
     8  0.120E+02  0.187E+01  0.187E+01  0.329E+01  0.250E-24
     9  0.559E+01  0.826E+00  0.822E+00  0.235E+01  0.711E-24
    10  0.510E-02  0.505E+00  0.506E+00  0.195E+01  0.153E-23
    11  0.393E-02  0.428E+00  0.428E+00  0.111E+01  0.197E-23
    12  0.624E-03  0.414E+00  0.414E+00  0.480E+00  0.298E-23
    13  0.116E-03  0.224E+00  0.224E+00  0.331E+00  0.251E-23
    14  0.223E-04  0.699E-02  0.700E-02  0.577E-02  0.315E-24
    15  0.224E-07  0.469E-08  0.398E-07  0.540E-04  0.302E-27
    16  0.953E-11  0.102E-13  0.422E-12  0.127E-08  0.399E-36
  Modal Gain & Mirror Loss [1/m] -0.359304E+006  0.569717E+004
 Calculate optical index change

 Direct eigen solver at lambda=  0.850000000000000
 Select modes with max index:
   3.64995114629961
 Start Arnoldi eigen solver
 sigma=   (727.940677315902,0.000000000000000E+000)
 End of Arnoldi eigen solver
 Eigen Value=[  0.661040E+003  0.955849E+001] Eqn. Error=  0.2910E-09
  Cmplx lateral modal index for z-segment:           1
           1 (3.47827733528838,2.514625154027444E-002)

 Searching for modal gain peak.
 Modal gain (1/m):  -354284.287056734
```

```
 At lambda=  0.850000000000000
Active Reg.# 1 Average Conc. (n&p)=  0.8041E+23  0.1469E+24
   Save bias data and continue.
 Lambda(um)=  0.850000000000000
 Lateral mode(s)=           1
 Emitted Power(mW):
 (Mode)      (Total)     (Front)     (Back)
 1            0.1829E-22  0.9147E-23  0.9147E-23

  Solver converged at
Voltage: (Volt) -0.2689E+00  0.0000E+00
Current: (A/m)   0.2316E-10 -0.2289E-10

...etc....
...etc....
...etc....
...etc....


Changing: current_1 with step:   0.9872E+01
 Error report for equations and variables:
  it#   eqns       potential  elec       hole       other
     1  0.642E+02  0.119E-01  0.735E-02  0.534E-02  0.132E-01
     2  0.387E+01  0.344E-04  0.222E-04  0.263E-04  0.622E-04
     3  0.125E-01  0.359E-08  0.307E-09  0.337E-08  0.201E-06
     4  0.813E-04  0.123E-11  0.413E-12  0.145E-11  0.131E-08
     5  0.460E-06  0.131E-12  0.262E-13  0.155E-13  0.141E-10
  Modal Gain & Mirror Loss [1/m]  0.569680E+004  0.569717E+004
 Calculate optical index change

 Direct eigen solver at lambda=  0.850000000000000
 Select modes with max index:
   3.65739599522727
 Start Arnoldi eigen solver
 sigma=   (730.913284438270,0.000000000000000E+000)
 End of Arnoldi eigen solver
 Eigen Value=[  0.663343E+003 -0.154368E+000] Eqn. Error=  0.5271E-09
  Cmplx lateral modal index for z-segment:          1
          1 (3.48423999577868,-4.054128695061282E-004)

 Searching for modal gain peak.
 Modal gain (1/m):   5696.79608291268
 At lambda=  0.850000000000000
```

```
Active Reg.# 1 Average Conc. (n&p)=  0.1743E+25  0.1813E+25
   Save bias data and continue.
 Lambda(um)=  0.850000000000000
 Lateral mode(s)=           1
 Emitted Power(mW):
 (Mode)      (Total)     (Front)     (Back)
 1            0.6664E+02  0.3332E+02  0.3332E+02

  Solver converged at
Voltage: (Volt) -0.1978E+01  0.0000E+00
Current: (A/m)   0.2500E+03 -0.2500E+03


 Data set #           3  printed at
Voltage: (Volt) -0.1978E+01  0.0000E+00
Current: (A/m)   0.2500E+03 -0.2500E+03
light:    0.6664E+02

 Printing 2D/3D data to output file
     Data file:test1.out_0003
 Generating .std file
 Completed .std file
 Date(d/m/y):          15           5        2009
 Time(s/m/h):          16          34          10
```

Every time a bias step is attempted, the previously converged solution will serve as the initial guess to the Newton solver. The type of bias step as well as its value is also shown in the log.

Once the Newton solver is called, it will iteratively attempt to reduce the equation error, which indicates how well the discretized equations are being satisfied. If the equations are exactly solved at the end of an iteration, then this value will be zero.

It also tries to minimize the variable error in order to get a stable solution. This is done by comparing the solution between successive iterations: the difference should go to zero once the solution stabilizes.

This is shown in the log in tabular form with each column having its own meaning:

it# : iteration number

eqns : equation error

potential : error on the potential variables

elec : error on the electron variables (electron concentration or quasi-fermi level)

hole : error on the hole variables (hole concentration or quasi-fermi level)

other : error on other variables not listed above (usually, photon density)

If a bias step should fail to converge, it is usually because the previously converged solution is a poor initial guess for this bias step. In that case, there is a built-in algorithm to retry with a smaller bias step which may have a better chance of converging.

If this procedure should also fail, then examining the evolution of the equation and variable errors may provide clues on how to improve convergence.

3.9 Output Data Organization

For performance reasons, we do not print all output data at every bias step taken. Instead, we divide the output data into two categories: bias-dependent data (scan_data) and structural/spectral data (xy_data).

The former includes bias current, voltage, laser power, etc... and is accumulated at every bias step. The latter includes position-dependent data like the carrier densities and certain spectral values like the modal gain which is only printed at specified intervals. This printing occurs either when a scan statement is finished, as specified by the print_step parameter of the **scan** statement. Whenever structural data is printed, any accumulated scan-dependent data is also printed.

By convention, the output file names are defined by the **output** statement in "*.sol" and by adding a numbered extension "_####". The extension is always _0001 for the equilibrium calculations and increases by one every time printing of the data is requested.

This means that all output data is assigned a "data set number" for later use. For example, let us supposed that there is a scan from 0V to 1V with max_step=0.3. Then the following data sets would be produced: #1 (equilibrium), #2 (0.3 V), #3 (0.6 V), #4 (0.9V) and #5 (end of scan at 1V). This can be confirmed by examining the ".sol.msg" file which contains a list of all data sets printed in a simulation and the associated bias value.

3.10 Analyzing the Results

To plot data, you can either use the CrosslightView GUI or use a .plt file. This input file is responsible for post-processing and can created in the same way as the other input files: by right-clicking in SimuCenter or using the command line:

```
C:\Work>c:\crosslig\lastip\setuplastip -plt test1
 Setting up .plt file for filebase=test1
 Please enter the graphics device
   (1) postscript (2) window (3) x11
 Your choice?
1
 Generating .plt file

 Adding a new plot to .plt file
 Please select the type of plot:
   (1) 1d plot along y
   (2) 1d plot along x
   (3) 2d contour plot
   (4) 2d vector plot
   (5) 3d surface plot
   (6) I-V characteristics
   (7) I-t characteristics
   (8) L-I characteristics
   (9) finish
 plot_type?
8

 Please enter the electrode number for I
 I_electrode_num?
1

 Adding a new plot to .plt file
 Please select the type of plot:
   (1) 1d plot along y
   (2) 1d plot along x
   (3) 2d contour plot
   (4) 2d vector plot
   (5) 3d surface plot
   (6) I-V characteristics
   (7) I-t characteristics
   (8) L-I characteristics
   (9) finish
 plot_type?
1

 Please enter the x-position of the 1d plot
   (1) at 0
   (2) at 1/10 xsize
```

```
   (3) at 1/4 xsize
   (4) at 1/2 xsize
   (5) at 3/4 xsize
   (6) at xsize
 x-position?
4
 Please enter the variable you wish to plot
 (1) electron concentration   (2) hole concentration
 (3) trap concentration   (4) potential
 (5) both electron & hole conc. (6) energy bands
 (7) elec current [x-component] (8) elec current [y-component]
 (9) hole current [x-comp]  (10) hole current [y-comp]
 (11) displac. curr. [x-comp] (12) displac. curr. [y-comp]
 (13) total current [x-comp] (14) total current [y-comp]
 (15) x-comp. electric field (16) y-comp. electric field
 (17) trap occupancy  (18) optical generation rate
 (19) optical field (20) hot electron energy
 (21) donor concentration (22) acceptor concentration
 (23) radiative recombination (24) Auger recombination
 (25) SRH recombination  (26) lattice temperature
 (27) electron conc difference between nodes
 (28) wave intensity (29) local gain
 (30) stimulated recombination
 variable?
28

 Adding a new plot to .plt file
 Please select the type of plot:
   (1) 1d plot along y
   (2) 1d plot along x
   (3) 2d contour plot
   (4) 2d vector plot
   (5) 3d surface plot
   (6) I-V characteristics
   (7) I-t characteristics
   (8) L-I characteristics
   (9) finish
 plot_type?
9
```

This creates the following:

```
$file:test1.plt
```

```
begin_pstprc
plot_data plot_device=postscript
get_data  main_input=test1.sol &&
  sol_inf=test1.out &&
  xy_data=[  3  3] scan_data=[1   3]
plot_scan scan_var=current_1 &&
  variable=laser_power
plot_1d variable=wave_intensity &&
  from=[  0.7500E+00  0.0000E+00] &&
  to=[  0.7500E+00  0.2200E+01]
end_pstprc
```

To run this .plt file, you can right click on the file in SimuCenter or use the command line again:

```
C:\Work>c:\crosslig\lastip\lastip test1.plt
C:\Work>c:\crosslig\lastip\psplot test1.ps
```

This will produce two plots. The first is the LI curve for the whole simulation: the specified **scan_data** range in **get_data** covers the whole simulation. The second plot will be the wave intensity profile at the last bias point (#3).

Note that certain seldom-used variables are not (by default) available for plotting. To make them available, you may need to use the **more_output** statement in the .sol file and re-run the simulation.

PARS OPTICS GROUP

Chapter 4

CONVERGENCE ISSUES

This chapter deals with convergence of the sparse matrix solver. In most cases, the simulator will attempt to correct the problem by using a smaller bias step but there are cases where the equation and variable error terms cannot be reduced properly. This will stall the progress of the simulation and cause the simulator to print an error message.

There are several possible causes of convergence difficulties: we have tried to discuss the most common problems and their solution in the following sections. Note that there may be several different solutions to the same problem.

Please report all other cases of non-convergence to Crosslight for technical assistance. However, please keep in mind that the the simpler the structure, the easier it is to debug. If possible, start from a simplified 1D device that works and progressively iterate towards your final design until the convergence problem appears.

4.1 Choice of voltage or current bias

The choice of voltage or current bias affects the convergence and stability of the Newton solver. In order to guarantee convergence, small changes in the applied bias should always result in small changes in the overall solution.

One particular situation where current bias is not appropriate is when the total amount of current flowing in the device is very small. Under these conditions, the actual current amount may fluctuate due to lack of numerical precision, making it difficult to use current bias. This situation can be detected by observing the net current over all the electrodes: if the sum is not zero, then Kirchoff's Current Law is violated and the current is too low to use as a control variable.

The solver can also enter a non-convergent state if the applied voltage bias is much higher than the turn-on voltage. Since the conductivity increases exponentially with

bias in a typical diode, seemingly small changes in voltage can result in very large changes of the solution.

This leads us to a simple general rule:

- Use voltage bias for devices with high resistance
- Use current bias for devices with low resistance

For example, a typical diode under forward bias has low resistance past its turn-on point but high resistance at lower bias or under reverse bias conditions. With these two extremes in mind, the following general strategy is recommended when setting up a simulation under forward bias:

1. Solve for equilibrium solution
2. Apply voltage until 80-90% of the built-in bias value is reached. It is also possible to automatically terminate a voltage scan when a certain current is reached by using the auto_finish parameter in the **scan** statement.
3. Verify that Kirchoff's Current Law is satisfied at this bias point
4. Apply current bias until desired value is reached

For laser devices, the photon coupling adds some complications that need to be considered. Once threshold is reached, the carrier concentration and Fermi-level splitting at the junction are pinned by the large stimulated recombination term. It is almost impossible to apply any voltage bias (which is roughly equal to the Fermi level splitting at the junction) without disturbing the solution.

Therefore, under lasing conditions, the only way to perform the simulation properly is to use current controlled bias. The general strategy outlined above should be slightly revised to ensure that the current scan starts below threshold.

For non-laser devices, there are a few typical applications where, based on the experience of the Crosslight team, the consistent use of voltage bias is strongly recommended:

- Devices based on organic semiconductors (OLED) and other materials with wide bandgap
- Devices with a large contact resistance
- Solar cells and photodectors

4.2 Special bias considerations for PICS3D

In PISC3D, the coupling of the photon density is more complicated than in LASTIP so it is not included by default. This introduces some extra steps that must be taken when defining the applied bias in a simulation.

In order to turn on the photon coupling, we need to know how many photons are in each longitudinal modes. This can be obtained by evaluating the round-trip gain equation (RTG) which is a function of the longitudinal index and gain profiles. However, these profiles are also affected by the local photon density through longitudinal spatial hole burning (LSHB) so it is difficult to set the initial guess.

To solve this conundrum, we consider the case where the laser cavity is below threshold. It is then safe to assume that the photon density is close enough to zero that the index profile is unaffected by LSHB. We can then use the unperturbed index profile to get an initial estimate of the photon density.

It is therefore required that the scan preceding the introduction of the photon coupling use the auto_finish=rtgain condition to terminate. This will calculate the positions of the longitudinal modes as well as provide an initial guess of the photon density in each mode.

However, the choice of the RTG value on which to terminate this scan can have a strong influence on the simulation and its convergence. If it is too high, then the photon density may not be as close to zero as originally thought and the initial guess may be inaccurate when the photon coupling is turned on. On the other hand, if the ending value is too low, then some modes critical to the simulation may be missing from the initial mode search. As a compromise, we suggest that the RTG ending value be set just above the transparency density of the gain material.

With this in mind, let us revise the general strategy from above for PICS3D use:

1. Solve for equilibrium solution
2. Apply voltage until 80-90% of the built-in bias value is reached. It is also possible to automatically terminate a voltage scan when a certain current is reached by using the **auto_finish** parameter in the **scan** statement.
3. Verify that Kirchoff's Current Law is satisfied at this bias point
4. Apply current bias with auto_finish=rtgain. The peak value of the RTG should be high enough to offset the mirror losses while still being less than 1.0 (in order to avoid the lasing threshold).
5. Double-check the modes found in the mode search to make sure all relevant modes are included.
6. Apply current bias with **solve_rtg=yes** until desired value is reached

For highly resistive devices or VCSELs with very low threshold currents, it may be advisable to merge steps 2 & 4 and use the **auto_finish=rtgain** condition during the voltage scan. A second termination condition using **auto2_finish** can be used to ensure the current is large enough for the final step.

4.3 Unphysical boundary

When the boundary condition is not reasonable, it may cause convergence problems. One such example occurs when an Ohmic contact is too close to an active region or an area with large splitting of IMREF's.

The Ohmic contact is an ideal boundary that forces the IMREF's to the same equilibrium level. Such an abrupt change may be hard for the continuity equation to follow and therefore, lead to a loss of convergence.

Another common case of non-convergence is in thermal simulations where the an external heat resistor is attached to a contact which is too close to the self-heating region.

4.4 Thermal runaway

Since material properties are affected by temperature effects, convergence may be adversely affected if the temperature increases too quickly in the device. In that case, the solution from a previous bias step may be too poor to use as the initial guess of the next bias step.

In that case, the only remedy is to use smaller bias steps to ensure a smooth increase in temperature. The max_step parameter of the **scan** statement may be used to control the step size.

The **heat_flow** statement also supports a max_temp_incr parameter that will force the use of a smaller step if the temperature increase is too high. However, this can cause a simulation to fail if the smaller step falls below the minimum allowed by the min_step parameter in **scan**.

4.5 Coarse mesh

A coarse mesh is a common cause of non-convergence. It is wise to stop the simulation and plot the bands, the distribution of potential and the carrier concentrations at the point of failure or at a previous data set. Inspection of the plots may enable you to locate rough mesh points in regions where solutions vary rapidly with distance or

are poorly sampled.

In particular, you should make sure to have sufficient mesh points near these regions of interest:

- Barriers formed by Schottky contacts, heterojunctions or boundaries between different doping levels
- Interband/intraband tunnel junctions
- Regions with strong current crowding
- Quantum well regions where there are many energy levels and the wave functions must be sampled correctly
- Regions where the optical mode peaks

For a given number of mesh points, their distribution within a given layer can also have an influence on convergence. If one of the above situations occurs within a layer, mesh point ratios should be used to put denser mesh near the region of interest. If there is a region of interest on both sides of a layer, symmetric mesh point distribution should be considered. For more details on mesh point ratios, consult the reference manual for the **put_mesh**, **layer** and **column** statements.

If an increased number of mesh points is still called for, then you may wish to try automatic mesh refinement with the **refine_mesh** command.

If a refined mesh does not help, try to reduce the complexity of the device (*e.g.* try a 1D device with similar structure). Testing a simple device may help you to find out the cause of the problem in the more complex device. Sometimes you may find that the solution has difficulty converging for a particular type of junction. This can be caused by approximations made to boundary conditions, or by unintentionally imposing unrealistic boundary conditions.

4.6 Fine mesh

Just as coarse mesh can lead to lack of convergence, too fine a mesh can sometimes also cause problems. In particular, regions with very low resistivity (metals, highly-doped contact regions, etc...) are at risk. The reason for this is simple: if the whole layer has a very small voltage drop, then the ΔV between closely-spaced mesh points can become negligible.

In the limiting case of a material with low resistivity, applying Ohm's law in this situation will result in a situation where $\Delta I = \frac{\Delta V}{R} \rightarrow \frac{0}{0}$ and the current between

closely-spaced mesh points becomes numerically unstable. The simulator will usually print a message about its inability to accurately control the current in this situation.

We note that this is a very rare situation that can usually be remedied by simply altering the distribution of mesh points in the low-resistance region.

4.7 Use of basic variables

The default basic variables in our simulator are the potential and quasi-Fermi levels. In some structures with many current blocking or high resistance regions, the solution at low bias is difficult to obtain with high accuracy. Devices of this type includes laser diodes (LD) with highly doped current blocking layers, junction field effect transistors (JFET) and charge coupled devices (CCD).

The reason is that under high resistance, the current is so small that to satisfy the current continuity equation within a small but finite tolerance, there are actually many possible solutions for the quasi-Fermi levels. As a consequence, the variables tend to vary between these many solutions and refuse to converge to a unique solution. Since the value itself of the quasi-Fermi levels is not small under these conditions, this fluctuation cannot simply be neglected.

The choice of electron and hole concentrations as the basic solution variable avoids the fluctuation problem mentioned above. The minority carrier concentration density is too low to show up in the error vector and we do not care about the accuracy of the minority carriers in those regions with low current densities.

The **newton_par** statement provides the parameter **change_variable** to allow the user to switch from Fermi levels to carrier concentrations. However, use of carrier concentration also has drawbacks.

The accuracy for minority carriers may not be accurate. The variables may also vary over 20 orders of magnitudes and as a result, the inversion of the Jacobian matrix may cause numerical overflow problems. Therefore, some of the off-diagonal matrix elements must be reduced artificially which may affect the overall solution.

4.8 Slow transient technique

For structures with high resistivity or with complicated high doping, convergence may be difficult at low bias before current is turned on. For example, nitride-based MQW structures with strong polarization interface charges are known to be difficult to converge.

One useful and surprisingly powerful technique is the Ťslow transientŤ method. For example, we may have an original difficult step:

```
scan var=voltage_1 value=-3.5
```

We can replace this with:

```
scan var=voltage_1 value=-3.5 var2=time value2_to=1.0
```

This means that the voltage will be increased over the course of 1 second. This is slow enough that transient effects will be minimal (since carrier lifetimes are in the ns range). However, the time step is also small enough that the displacement current makes a numerically significant contribution to the current continuity equations.

There is physical and numerical basis for such a method: highly insulating regions inherently cause instabilities with the current continuity equation simply because the system may flip between different states that have similar steady state currents.

In reality, this flip is facilitated by a transient charging/discharging current which drives the system into the correct physical steady state. As a matter of fact, one can make the point that there is no such thing as steady state: just the limiting case where $dt \to \infty$. Adding a transient component merely restores some of the original physics to the equations.

Such a technique has been used to tackle some conventionally difficult cases such as floating poly gate in MOSFET and highly insulating organic/insulator structures.

4.9 Poor initial guess solution

As is well known for Newton's method, a poor initial guess solution may cause the solution to diverge. Such a situation is normally not a concern because the simulator has automated the procedure of finding the best guess solution. If a solution is not found with an initial guess solution from a previous bias, then the simulator automatically reduces the bias step and tries a new Newton iteration.

As mentioned previously, simulations of some structures with highly doped current blocking layers have difficulty converging at low bias because of high resistance. A poor initial guess may cause problems in this situation. One may wish to avoid the solution at low bias by applying a large initial bias step. If the device structure is relatively simple this trick often works.

In a case where the device structure is complicated and the doping of the current blocking layer is high, this trick may fail. One of the reasons for such a failure is that the guess solution (equilibrium solution) is too far away from the solution at high bias. Such a condition is worsened by high doping at the current blocking layer.

A trick to overcome the convergence difficulty is to solve for an " easy device" at high bias first. Then the solution from the " easy device" is used to initialize the

solution for the " target device" at high bias. Here the " easy device" has the same mesh and geometric structure as the " target device" except the current blocking layer is lightly doped. For the simulator, a structure with a lightly doped blocking layer is an easy structure.

4.10 Current-blocking structures

4.10.1 n-p-n & p-n-p junctions

Convergence may be difficult for structures which use n-p-n or p-n-p junction to block the current flow for a certain region. This kind of design is commonly used in buried heterostructure (BH) lasers.

This type of structure requires special attention since the use of quasi-Fermi levels as the default variables often causes convergence difficulties. As mentioned in a previous section, the change of variable from Fermi levels to carrier concentrations may avoid such convergence problems. However, if the Fermi levels solution is desired for any reason, or if the use of new variables fails, the following techniques may be used to force the convergence of the solver using quasi-Fermi levels.

Current-blocking structures have difficulty converging because more material regions are involved. The heavy doping in the current blocking layer also tends to cause trouble, because the current flow condition there is very different from that in the other regions.

To guarantee good convergence, we must supply the simulator with a good guess solution at a reasonably high bias voltage. Slowly ramping up the bias voltage does not always help because low bias itself is a problem, as was discussed in previous sections.

We recommend the following trick whenever a heavily doped current-blocking structure shows non-convergence:

1. Create an identical structure with lightly doped current blocking layer. Declare this lightly doped current blocking layer as having a "new_doping" instead of the usual "doping".

2. Run the simulator for this structure and ramp up the voltage to 80-90 % of the bandgap of the active region. This should be much easier than before since we are dealing with a structure where the doping has been artificially reduced.

3. Ramp up the doping density in the blocking layer (marked as "new_doping") to a realistic doping level.

4. Now that the device structure/doping is what we intend to simulate, continue the simulation with current bias and other standard simulation tasks.

4.10.2 Barriers/Superlattices

High barriers ($\gg kT$) may block current injection if only thermionic emission is considered. They are commonly found at Schottky contacts or at material/doping interfaces. Quantum tunneling may be required to overcome this and allow the current to flow properly. Periodic structures may also support miniband tunneling at some carrier energies (Esaki-Tsu model).

The user is advised to consult Chapter 9 for a more complete discussion of tunneling models.

4.10.3 Insulating layers

In our software, insulating materials such as SiO_2 are often described by a special kind of macro (material type=insulator). Regions using these macros are special in that the current continuity equation is not solved: instead, the current is explicitly set to zero. This means that features that enhance the current (tunneling, impact ionization, etc...) will not function since they would multiply a current value that is exactly zero.

In order for these features to function, the zero current must be replaced with a numerically small value that better represents reality. This can be done by considering the insulator material as a wide bandgap semiconductor. For example, users may substitute the sio2 macro for s-sio2 in their design.

4.10.4 Deep MQW regions

Very deep and shallow QWs such as those found in nitride-based devices may block current if we assume that all carriers are thermalized (i.e. Drift-Diffusion model). This can manifest itself as an unrealistically high turn-on voltage since a stronger field is required to get the current flowing.

A non-local/quantum transport correction term can facilitate the current injection and solve the related convergence issues. This can be turned on by using the **q_transport** statement.

4.11 Use of Auxiliary Ohmic Contacts

A powerful technique to force the simulation software to converge is the use of auxiliary ohmic contacts. For some complex devices, certain high resistance areas are far away from any boundaries. For example, a p-n-i-p-n structure under reverse bias condition has its i-region well isolated from the ohmic contacts. As a result, the variables may fluctuate in the i-region and we may have a convergence problem no matter how fine the mesh spacing is.

In such a case, we may add an auxiliary ohmic contact to be attached to a single mesh point in the isolated high resistant region. When we use voltage control on the auxiliary contact, the solver should be easier to converge because the new contact prevents the variables from fluctuating in the high resistance region.

However, the voltage control on the contact may cause current to flow in or out of it while in reality, no current should exist in such a contact. To eliminate the effects of the auxiliary contact, we can use current control to reduce its current to zero after the desired bias on the other electrodes has been obtained.

For example, in the above p-n-i-p-n structure (see Fig. 4.1), we wish to reverse bias to breakdown at about 10 volts. We can simultaneously ramp up the voltages at the auxiliary contact to about 5 volts to ensure convergence. After the device breaks down and current starts to flow at around 10 volts of total bias, we can use current control to force the current at the auxiliary contact to be zero, thus eliminate its effects on the device. Without the the auxiliary contact, the above device would be difficult to bias to breakdown.

4.12 Bandgap reduction technique

Wide bandgap materials are hard to converge since their intrinsic carrier densities are low. The reverse junction current is orders of magnitude lower than their narrow bandgap counterparts. The problem is more serious if impact ionization is involved because the surge of ionization current in reverse junction may never be detected by the numerical solver since the reverse current is below the numerical floating point accuracy.

One technique is to artificially reduce the bandgap first, achieve the desired bias current and finally increase the semiconductor bandgap back to its original value.

For example:

```
equilibrium bandgap_reduction = 0.2
scan var=voltage_1 value_to = -0.5
scan var=current_1 value_to = 10e-3
```

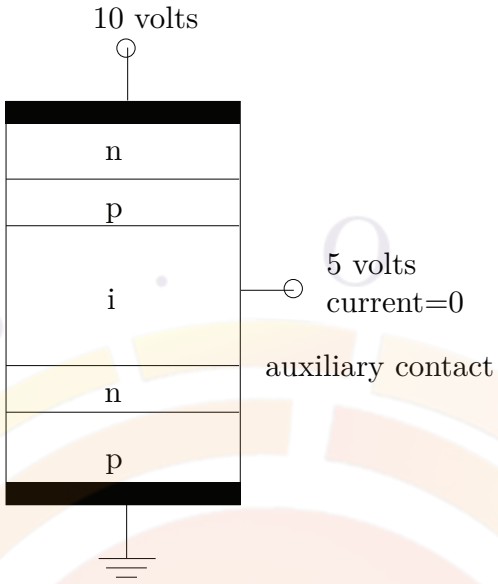


Figure 4.1: Schematic for the use of auxiliary contact for a p-n-i-p-n reverse biased structure.

```
scan var=bandgap_reduction value_to=0.0 var2=current_1 value2_to=10e-3
scan var=voltage_1 value_to = 0.0
```

This reduces the original bandgap value by 20% (e.g. 1 eV $\rightarrow$ 0.8 eV) and increases the bias to the desired value. Afterwards, the bandgap is restored to its original value while keeping the desired bias current fixed; the voltage will change to match.

As a final step, we get the IV curve by ramping down the bias: since the bandgap was altered during when the bias was ramped up, the IV curve is incorrect for this region. In this situation, care should be taken to only plot data from the last scan statement.

4.13 Negative differential mobility issues

For materials with negative differential mobility due to transition of electrons from the Γ band to L and/or X bands, convergence may become a problem when the applied voltage is high. Examples of such problems are a HEMT under high voltage bias and a laser diode under high current injection condition.

The steady state solution can be driven into non-convergence because of the negative differential resistance due to the special field dependence mobility as shown in Fig. 4.2 (as defined by the "n.gaas" velocity model in the gaas macro).

In reality, the device may be driven into unstable or oscillation states (Gunn effect) at some bias points. We rarely see such kind of negative I-V characteristics in steady

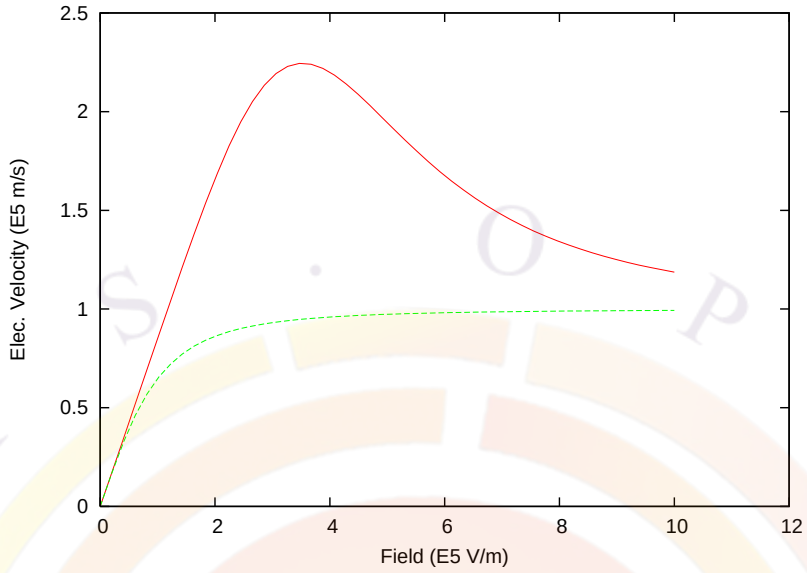


Figure 4.2: Field dependence electron velocity of undoped GaAs. The solid curve is the "n.gaas" mobility model while the dash curve is the "beta" model.

state measurements. A possible explanation is that the field/current adjusts itself in the region of negative resistance in the form of transient redistribution and the steady state terminal I-V curve bypasses the peak of negative resistance.

Thus, to overcome the non-convergence in steady state simulation in such a situation, we suggest using a different form of mobility model (such as the "beta" model, as illustrated by dash line in Fig. 4.2).

An alternative method to overcome the non-convergence is to use transient simulation while biasing the FET. This method means the simulator and the computer hardware must work harder.

Part II

PHYSICAL AND NUMERICAL MODELS

PARS OPTICS GROUP

Chapter 5

DRIFT-DIFFUSION MODEL

This chapter provides a description of the basic theories used in all of Crosslight's simulation software packages. The variety of physical phenomena in a semiconductor require many different physical models. However, the drift-diffusion (DD) model is the most basic since many of the well known characteristics of a semiconductor device such as the capability to amplify electrical signal may be explained using the DD theory.

This chapter will also explain the boundary conditions and basic properties such as mobility and pure resistor behavior.

5.1 Basic equations

The basic equations[1] used to describe the semiconductor device behavior are Poisson's equation:

$$-\nabla\cdot\left(\frac{\epsilon_0\epsilon_{dc}}{q}\nabla V\right)=-n+p+N_D(1-f_D)-N_Af_A+\sum_j N_{tj}(\delta_j-f_{tj}), \qquad (5.1)$$

and the current continuity equations for electrons and holes:

$$\nabla\cdot J_n-\sum_j R_n^{tj}-R_{sp}-R_{st}-R_{au}+G_{opt}(t)=\frac{\partial n}{\partial t}+N_D\frac{\partial f_D}{\partial t}, \qquad (5.2)$$

$$\nabla\cdot J_p+\sum_j R_p^{tj}+R_{sp}+R_{st}+R_{au}-G_{opt}(t)=-\frac{\partial p}{\partial t}+N_A\frac{\partial f_A}{\partial t}. \qquad (5.3)$$

These equations govern the electrical behavior (*e.g.*, I-V characteristics) of a semiconductor device.

Optionally, two more equations are used to described the carrier energy (w) or the carrier temperature distribution. This is not to be confused with the lattice temperature; the former is a description of how the carrier distribution deviates from Fermi-Dirac distribution. Such a model is also referred to as the hydrodynamic model.

Our theory here is based on the formulation derived by Azoff [2][3]. For simplicity, we only present the equations for hot electrons. The corresponding equations for hot holes are completely analogous:

$$\nabla \cdot S + R_n w - \nabla E_c \cdot J_n + \frac{n(w - w_0)}{\tau_w} + \frac{\partial (nw)}{\partial t} = 0 \tag{5.4}$$

$$S = -\frac{5}{3} J_n w - \frac{10}{9} \mu_n n w \nabla w. \tag{5.5}$$

where w is the total energy of an electron and $w_0 = 3kT/2$ is the electron energy at equilibrium. S is the electron energy flux intensity and τ_w is the energy relaxation time.

Definitions of symbols are listed in the nomenclature at the end of this chapter. More details about the derivation and discretization of the hydrodynamic model is given in the Appendix.

The primary function of our simulator is to solve these equations self-consistently for the electrostatic potential, V, the electron and hole concentrations (n, p) and the electron and hole energies (W_n, W_p).

From theories of semiconductor device physics [1] the carrier flux densities J_n and J_p in Equations (5.2) and (5.3) can be written as functions of the carrier concentration and the quasi-Fermi levels:

$$J_n = n \mu_n \nabla E_{fn} \tag{5.6}$$

$$J_p = p \mu_p \nabla E_{fp} \tag{5.7}$$

where μ_n and μ_p are mobilities of electrons and holes. For convenience, we use carrier flux density and current density interchangeably even though they differ by a factor of electron charge.

For the hydrodynamic model, the expression for the electron (or hole) current is modified:

$$J_n = \mu_n \left\{ -n \nabla [\psi + \chi + \gamma_n] + \frac{2}{3} \nabla (nw) - nw \nabla ln(m_n) \right\} \tag{5.8}$$

More details about the current flow expression in the hydrodynamic model is given in the Appendix.

5.1.1 SRH and Auger recombination

The software treats carrier recombination due to deep level traps [Shockley-Read-Hall (SRH) recombination] using:

$$R_n^{tj} = c_{nj} n N_{tj}(1 - f_{tj}) - c_{nj} n_{1j} N_{tj} f_{tj} \tag{5.9}$$

$$R_p^{tj} = c_{pj} p N_{tj} f_{tj} - c_{pj} p_{1j} N_{tj}(1 - f_{tj}) \tag{5.10}$$

where n_{1j} is the electron concentration when the electron quasi-Fermi level coincides with the energy level E_{tj} of the jth trap. A similar definition applies to p_{1j}.

Under transient conditions, the following trap dynamic equation is valid [4]:

$$N_{tj} \frac{\partial f_{tj}}{\partial t} = R_n^{tj} - R_p^{tj} \tag{5.11}$$

The capture coefficients c_{nj} and c_{pj} for electrons and holes relate to the lifetime of the carrier due the jth recombination center by the following relation:

$$\frac{1}{\tau_{nj}} = c_{nj} N_{tj} \tag{5.12}$$

$$\frac{1}{\tau_{pj}} = c_{pj} N_{tj} \tag{5.13}$$

One can show that Equations (5.9) to (5.13) are equivalent to the standard expression of the SRH model [1] under steady state conditions.

The capture coefficients can be further expressed as:

$$c_{nj} = \sigma_{nj} \overline{v}_n \tag{5.14}$$

$$\overline{v}_n = \sqrt{\frac{8kT}{\pi m_n}} \tag{5.15}$$

$$c_{pj} = \sigma_{pj} \overline{v}_p \tag{5.16}$$

$$\overline{v}_p = \sqrt{\frac{8kT}{\pi m_p}} \tag{5.17}$$

A trap (or recombination center) is completely specified by its density N_{tj}, capture cross sections σ_{nj} and σ_{pj}, and energy level E_{tj}.

The Auger recombination rate is given by

$$R_{au} = (C_n n + C_p p)(np - n_i^2), \tag{5.18}$$

where the Auger coefficients C_n and C_p depends on the type of material simulated.

5.1.2 Carrier statistics

The electron and hole concentrations in semiconductors are defined by Fermi-Dirac distributions and a parabolic density of states which, when integrated, yield [1]:

$$n = N_c F_{1/2}\left(\frac{E_{fn} - E_c}{kT}\right) \tag{5.19}$$

$$p = N_v F_{1/2}\left(\frac{E_v - E_{fp}}{kT}\right) \tag{5.20}$$

where $F_{1/2}$ is the Fermi integral of order one-half.

For the convenience of numerical evaluation, the approximation proposed by Bednarczyk and Bednarczyk is used [5]:

$$F_{1/2}(x) \approx \left(e^{-x} + \xi(x)\right)^{-1} \tag{5.21}$$

$$\xi(x) = \frac{3}{4}\sqrt{\pi}\left[\nu(x)\right]^{3/8} \tag{5.22}$$

$$\nu(x) = x^4 + 50 + 33.6x\left\{1 - 0.68exp[-0.17(x+1)^2]\right\} \tag{5.23}$$

This expression is accurate to within 0.4% of error in all ranges.

In the limit of low carrier concentration Equations (5.19) and (5.20) reduce to the familiar Boltzmann statistics:

$$n = N_c exp\left(\frac{E_{fn} - E_c}{kT}\right) \tag{5.24}$$

$$p = N_v exp\left(\frac{E_v - E_{fp}}{kT}\right) \tag{5.25}$$

The program uses the more general Fermi-Dirac statistics of Equations (5.19), (5.20) and (5.21) by default.

5.1.3 Incomplete ionization of impurities

The simulation program can accurately account for the incomplete ionization of shallow impurities in semiconductors. The occupancies f_D and f_A are used to describe the degree of ionization. It is assumed that the shallow impurities are in equilibrium with the local carriers and therefore the occupancy of the shallow impurities can be described by:

$$f_D = \frac{1}{1 + g_d^{-1} exp[(E_D - E_{fn})/kT]} \tag{5.26}$$

$$f_A = \frac{1}{1 + g_a^{-1} exp[(E_A - E_{fp})/kT]} \tag{5.27}$$

where the subscripts D and A are used to denote shallow donors and acceptors, respectively. The degeneracy levels are automatically set to $g_d = 2$ and $g_a = 4$.

From the discussions of Section 5.1.1, the occupancy of a deep level trap can be determined through the trap dynamic equation, Equation (5.11). In general the deep trap is not in equilibrium with the carriers (*i.e.*, the trap does not share the same quasi-Fermi level as the carriers).

From Equations (5.9), (5.10) and (5.11), one obtains the following expression for the trap occupancy under steady state conditions:

$$f_{tj} = \frac{c_{nj} n + c_{pj} p_{1j}}{c_{nj}(n + n_{1j}) + c_{pj}(p + p_{1j})} \tag{5.28}$$

In the case of surface states or surface recombination centers, the software allows for the distribution of dense traps near the surface region. This provides a mechanism for the surface charge states as well as for surface recombination. Fermi level pinning effects on a semiconductor surface can be modeled using this approach.

In a transient simulation the trap occupancy is a function of time, depending on the trap capture rates as well as on the local carrier concentrations. The program uses Equation (5.11) to determine the trap states at each transient time step.

By default the simulation software assumes incomplete ionization of dopants with energy level defined by the **level** parameter in the **doping** statement. The dopants of a specific material may be forced to be fully ionized with the command **full_ionization**.

For a more accurate model of incomplete ionization in heavily-doped semiconductors, the Mott transition model can also be used.

5.1.4 Poole-Frenkel model of incomplete ionization

The Poole-Frenkel effect (also Frenkel-Poole effect or field induced emission) was originally used to describe field dependent thermionic emission from traps in the bulk of an insulator [1]. This mechanism is, however, equally applicable to incomplete ionization of impurities in semiconductors.

Under an electric field F, the electrostatic potential near an impurity center is modified and the effective work function or the ionization energy is reduced by:

$$\Delta E_{PF} = \sqrt{\frac{qF}{\pi \epsilon_0 \epsilon}} \tag{5.29}$$

The Poole-Frenkel model is implemented by shifting the ionization energy by ΔE_{PF}. The field dependence of this shift introduces some complication into the discretized Poisson's equation. The reason is that at each node, additional work must be done to search for the maximum field component surrounding the node.

Another complication associated with the Poole-Frenkel model occurs at the ohmic contacts, because the simulator solver is set up such that the built-in potential at an ohmic contact is always fixed, or in other words, the dopant energy level or the Imref at the contact is fixed relative to the band edges. Such an assumption contradicts with the Poole-Frenkel model which changes the ionization energy and the Imrefs as the applied field is varied.

To overcome this difficulty, the ionization energy at the contact is allowed to be shifted to a constant value before the simulation. Without such a shift in the ionization energy at the contact, the ohmic contact may behave like a Schottky contact because of a large fixed built-in potential.

The Poole-Frenkel model is important when the ionization energy is large and the temperature is low (*e.g.*, 100 meV at 100K). Without such a model, the semiconductor may have an unrealistic high resistance.

By default, this model is turned off and can be enabled using the parameter **pf_model** of the **doping** statement.

5.1.5 Heavy doping effects

The Mott transition occurs at high doping concentrations, when the distance between impurities/dopants becomes comparable to the Bohr radius. It has been shown[6] that the effective Bohr radius a_B^* and critical impurity concentration N_{crit} are determined by:

$$a_B^* = \frac{\epsilon_r}{m^*/m_0} 0.53\text{Å} \tag{5.30}$$

$$N_{crit}^{1/3} = \frac{1}{4a_B^*}\left(\frac{\pi}{3}\right)^{\frac{1}{3}} \tag{5.31}$$

where ϵ_r is the relative dielectric constant and m^* is the carrier effective mass. The ionization energy of the impurity as a function of the concentration can then be obtained by:

$$E_D = E_{D0}[1 - (N_D/N_{crit})^{1/3}] \tag{5.32}$$

A similar expression exists for acceptors. This model is activated by default for all shallow dopants in Crosslight Software.

The applicability of this model to deeper dopants such as GaN:Mg is not as well understood so a maximum ionization energy of 100 meV is used to switch the Mott transition on/off. This behavior can be controlled on a per-material basis using the **dopant_ionization_model** command.

The ionization energy can also be manually controlled by the user to match measured values. There are multiple ways of doing so but the most convenient methods are:

- The **adjust_doping** statement in the layer file. This overrides the **level** parameter of the **doping** statement generated by layer.exe.
- The **shal_acpt_level** and **shal_dnr_level** commands can be implemented as parameter overrides in the input files or as part of a material macro. These commands are flexible and can even be used to create custom formulas for the activation energy dependence on the dopant concentration.

5.1.6 Carrier mobility

The carrier mobilities μ_n and μ_p account for the scattering mechanism in electrical transport.

One of the main effects on the mobility is the local electrical field [1]. The software provides several analytical formulas of field dependent mobility:

- 1. The simplest mobility model uses constant mobilities μ_{0n} and μ_{0p} for electrons and holes, respectively, throughout each material region in the device. This model should not be confused with constant velocity: constant mobility means *linear* velocity.
- 2. Another simplified field dependent mobility model is the two-piece mobility model:

$$\mu_n = \begin{cases} \mu_{0n} & \text{for } F < F_{0n}, \\ v_{sn}/F & \text{for } F \geq F_{0n}. \end{cases} \tag{5.33}$$

$$v_{sn} = \mu_{0n} F_{0n} \tag{5.34}$$

for the electron mobility. F_{0n} is a threshold field beyond which the electron velocity saturates to a constant.

Similar expressions can be defined for holes:

$$\mu_p = \begin{cases} \mu_{0p} & \text{for } F < F_{0p}, \\ v_{sp}/F & \text{for } F \geq F_{0p}. \end{cases} \tag{5.35}$$

$$v_{sp} = \mu_{0p} F_{0p} \tag{5.36}$$

We note that this model may be used to approximate a constant velocity model by setting a very low of F_0 so that the velocity is saturated throughout the useful simulation range.

- 3. The Canali or *beta* model[7] is commonly used in the literature and has the following form for electrons and holes, respectively:

$$\mu_n = \frac{\mu_{0n}}{(1 + (\mu_{0n}F/v_{sn})^{\beta_n})^{1/\beta_n}} \tag{5.37}$$

$$\mu_p = \frac{\mu_{0p}}{(1 + (\mu_{0p}F/v_{sp})^{\beta_p})^{1/\beta_p}} \tag{5.38}$$

- 4. The transferred electron model[7] or *n.gaas* model is used in many III-V compound semiconductors which exhibit negative differential resistance due to the transition of carriers into band valleys with lower mobility [1]. The software has implemented the following field dependent model for this case:

$$\mu_n = \frac{\mu_{0n} + (v_{sn}/F_{0n})(F/F_{0n})^3}{1 + (F/F_{0n})^4} \tag{5.39}$$

- 5.Poole-Frenkel is the name given by Crosslight to the hopping mobility model commonly used for organic semiconductors; this movement is very similar to the Poole-Frenkel effect (Frenkel-Poole emission) in insulators[1] with an exponential field-dependent term determining the probability of a highly localized carrier moving from one site to another.

  The mobility from this model may be expressed using the following formula:

$$\mu = \mu_0 exp[(F/F_{cr})^{px}] \tag{5.40}$$

- 6. A modified transferred-electron mobility model designed for GaN devices[7]. See **beta_mte** for details.

- 7. User-defined mobility model. More details can be found in the reference section under **user_defined_mobility**.

Besides the field dependence of the mobility, another important effect is the impurity dependence of the low field mobility [1]. The program by uses the following formulas by default:

$$\mu_{0n} = \mu_{1n} + \frac{(\mu_{2n} - \mu_{1n})}{1 + \left(\frac{N_D + N_A + \sum_j N_{tj}}{N_{rn}}\right)^{\alpha_n}} \tag{5.41}$$

$$\mu_{0p} = \mu_{1p} + \frac{(\mu_{2p} - \mu_{1p})}{1 + \left(\frac{N_D + N_A + \sum_j N_{tj}}{N_{rp}}\right)^{\alpha_p}} \tag{5.42}$$

where the various parameters are fitting parameters from experimental data which may also be temperature-dependent. Additional impurity models may be invoked through **low_field_mobility_model**.

In many applications, the mobility is anisotropic and also depends on the vertical field. A number of vertical field dependent model (such as the Lombardi model) are implemented within the statement of **mobility_xy**.

5.2 Boundary Conditions

The boundary conditions for the electrical part [Equations (5.1) to (5.3)] include ohmic contacts, Schottky contacts, Neumann (reflective) boundaries, lumped elements and current controlled contacts. For the hot carrier equations, the boundary condition here is the contact carrier temperature.

5.2.1 Ohmic contact

Ohmic contacts are implemented as simple Dirichlet boundary conditions, where the surface potential and electron and hole quasi-Fermi levels (V_s, E^s_{fn},E^s_{fp}) are fixed. The minority and majority carrier quasi-Fermi potentials are equal and set to the applied bias of the electrode:

$$\phi^s_n = \phi^s_p = -E^s_{fn} = -E^s_{fp} = V_{applied}. \tag{5.43}$$

The potential V_s at the boundary is fixed at a value consistent with zero space charge:

$$-n + p + N_D(1 - f_D) - N_A f_A + \sum_j N_{tj}(\delta_j - f_{tj}) = 0. \tag{5.44}$$

With the solution of V_s and Equation (5.43) one can use Equations (5.19) and (5.20) to calculate n_s and p_s.

It should be pointed out that the purpose of creating an Ohmic contact in a simulation is to have a boundary condition which does not disturb the area of simulation but provides a path for current flow. This is in contrast to the Schottky contact which can be either injecting or depleting carriers, depending on the polarity of the bias.

In most device simulations, the ideal Ohmic contact with a charge neutral assumption is sufficient as a good carrier injector if the contact area is highly doped or if the carrier injection requirement is not too high. If the vicinity of the contact has low doping or if a high level of carrier injection is required, the ideal Ohmic contact described above ceases to be a valid model for a realistic Ohmic metal contact. In such a case, the simulator refuses to inject carriers into the device, no matter how large the bias current is.

To avoid this problem, remember to use high doping in the vicinity of an ideal Ohmic contact region. Otherwise, a low barrier Schottky contact may also be used.

5.2.2 Schottky contacts

Schottky contacts to the semiconductor are defined by the barrier height of the electrode metal and a surface recombination velocity.

The surface potential at a Schottky contact is defined by:

$$V_s = \chi - \chi_{ref} - \phi_b + V_{applied}. \tag{5.45}$$

where χ and χ_{ref} are the electron affinities of the semiconductor and a reference material, respectively, and ϕ_b is the Schottky barrier height.

Note that this applies only for n-contacts. For p-contacts, this equation needs to be modified by the semiconductor bandgap to calculate the proper hole barrier. The sum of the barrier heights for electron and holes is expected to be equal to the bandgap [1].

In general, the quasi-Fermi levels E^s_{fn} and E^s_{fp} are no longer equal to $V_{applied}$ and are defined by a current boundary condition at the surface instead:

$$J_{sn} = \gamma_n \overline{v}_n^{therm}(n_s - n_{eq}), \tag{5.46}$$
$$J_{sp} = \gamma_p \overline{v}_p^{therm}(p_s - p_{eq}), \tag{5.47}$$

where J_{sn} and J_{sp} are the electron and hole currents at the contact, n_s and p_s are the actual surface electron and hole concentrations, and n_{eq} and p_{eq} are the equilibrium electron and hole concentrations if infinite surface recombination velocities are assumed (*i.e.* $\phi_n = \phi_p = V_{applied}$).

The thermal recombination velocities are given by:

$$\overline{v}_n^{therm} = \sqrt{\frac{kT}{2m_n\pi}}, \tag{5.48}$$
$$\overline{v}_p^{therm} = \sqrt{\frac{kT}{2m_p\pi}}, \tag{5.49}$$

The constants γ_n and γ_p provide a mechanism to include any correction (*e.g.* due to tunneling) to the standard theory.

5.2.3 Abrupt heterojunctions

An abrupt junction is also considered a boundary condition in the program because most physical quantities are discontinuous at the interface. Similar to a Schottky contact, an abrupt heterojunction exhibits non-ohmic behavior, such as rectification effects.

The physical model used by the software for current transport across the junction is the thermionic emission model [1]. A similar expression for the current flux across the junction can be defined as:

$$J_{hn} = \gamma_{hn} \overline{v}_{bn}^{therm}(n_b - n_{b0}), \tag{5.50}$$
$$J_{hp} = \gamma_{hp} \overline{v}_{bp}^{therm}(p_b - p_{b0}), \tag{5.51}$$

where n_b and p_b denote the electron and hole concentrations, respectively, on the barrier side of the junction. n_{b0} and p_{b0} are the corresponding concentrations when the quasi-Fermi levels are the same as those on the opposite side. These equations ensure that, when the quasi-Fermi levels on both sides of the barriers are the same, the net current is zero. The derivation of the boundary conditions of the above equations can be found in Refs. [8] and [9].

The thermal recombination velocities are given by:

$$\overline{v}_{bn}^{therm} = \sqrt{\frac{kT}{2m_{bn}\pi}}, \tag{5.52}$$

$$\overline{v}_{bp}^{therm} = \sqrt{\frac{kT}{2m_{bp}\pi}}. \tag{5.53}$$

Other symbols have similar meanings as those in Section 5.2.2.

The abrupt heterojunction model described here is important to accurately model the thermionic emission behavior (*e.g.* temperature dependence) of a device using an abrupt heterojunction as a means of carrier confinement.

The abrupt junction model has its limitations, however, especially when both sides of the junction are heavily doped with the same type of dopant (*e.g.* donors). In such a case, the Fermi level is strongly pinned at the band edges on both sides, and the potential barrier is forced to form a sharp peak above the Fermi level. Since the peak region in the barrier is strongly depleted of carriers, the resistance is very high according to Equation (5.6). If this junction happens to be reverse biased (*i.e.* the carriers on the barrier side tend to be more depleted when the bias is applied), the high resistance can cause an unrealistically large voltage drop there.

Fortunately, the difficulty associated with a highly doped abrupt heterojunction does not occur very often near the active regions, because junctions there are rarely heavily doped with the same dopant on both sides. If the heavily doped contact region has an abrupt heterojunction, problems may arise. An example of this is GaAs/AlGaAs, which may have a heavily doped GaAs substrate under the heavily doped cladding region.

In most cases the abrupt junction model implemented in the program is adequate. In setting up a new device structure one may ignore the problem mentioned above unless the simulation result shows an unrealistically large voltage drop. The user may be able to spot such a problem just from simulator's run-time printout. For example, if the printout shows shows that 10 Volts are needed to pass a few mA of current, it usually indicates there is a barrier of some kind.

By plotting the band diagram, it should be possible to locate if an abrupt junction is causing this problem. If so, we can replace the abrupt junction with a graded barrier without affecting any active regions. Usually a graded barrier with grading length of 100 Åor so is adequate to replace the heavily doped junction that is causing the

trouble.

This limitation of the program arises from the assumption of drift-diffusion as the key transport mechanism. In reality the main mechanism of carrier transport across the thin and sharp barrier is quantum tunneling.

5.2.4 Neumann boundaries

Along the outer non-contacted edges of simulated devices, homogeneous reflecting Neumann boundary conditions are imposed so that current only flows out of the device through the contacts. Additionally, in the absence of surface charge along such edges, the normal electric field component goes to zero. In a similar fashion, current is not permitted to flow from the semiconductor into an insulating region; further, at the interface between two different materials, the difference between the normal components of the respective electric displacements must be equal to any surface charge present along the interface.

5.2.5 Lumped elements

Lumped elements have been created to cut down the number of grid points to discretize some device structures, thereby saving CPU time. Lumped resistance might be useful in a simulation of a semiconductor device structure with a substrate contact located far away from the active region. If the whole structure were to be simulated, a tremendous number of grid points, probably more than half, would be wasted to account for a purely resistive region of the device. Because CPU time is generally a superlinear function of the number of grid points, including such regions can be quite expensive.

Applying this boundary condition creates an extra unknown (V_s), the voltage on the semiconductor contact, which if defined by Kirchoff's equation:

$$\frac{V_{applied} - V_s}{R_s} - \sum_{i=1}^{N_b} (J_n + J_p) = 0, \tag{5.54}$$

where N_b is the number of boundary grid points associated with the electrode of interest.

Note that this auxiliary equation, due to the currents inside the summation, has dependencies on the values of potential and carrier concentrations at the nodes on the electrode as well as all nodes directly adjacent to the electrode. It is important to remember that a lumped element contact will have a single voltage (or potential-adjusted for possible doping non-uniformity) associated with the entire electrode.

The simulator should be used as much as possible to calculate any resistance components that might be included as lumped elements. For instance, in the case of sub-

strate resistance of a semiconductor device, one could just simulate the n-substrate with ohmic contacts at both ends. From the plot of the contact current versus voltage, the resistance can be directly extracted from the slope.

5.2.6 Current boundary condition

The current boundary condition is a very important option in the simulation of device. For example, when a laser diode is biased beyond threshold, the gain and carrier concentration essentially saturate. This also causes the bias voltage to saturate, while the current continues to rise because of stimulated emission. Under such conditions, operating the simulator at a voltage-controlled mode would cause the current to jump to huge values for a small increase in voltage.

In some devices (not necessarily lasers), the terminal current is a multi-valued function of the applied voltage. This condition implies that for some voltage boundary conditions, a numerical procedure may end up, depending on the initial guess solution, with a solution in either of two distinct and stable states. Furthermore, a condition of primary interest is at the trigger point, where $dV/dI = 0$, which is difficult to obtain with a simple voltage boundary condition. Additionally, it is nearly impossible to compute any solutions in the negative resistance regime with voltage inputs.

A possible alternative to the difficult situation mentioned above would be to define a current controlled boundary condition, since voltage can be thought of as a single-valued function of the terminal current. Such a boundary condition has been implemented in the software, as an auxiliary equation with an additional unknown boundary potential. Like the lumped element case, a Kirchoff equation is written at the contact points:

$$J_{source} - \sum_{i=1}^{N_b} (J_n + J_p) = 0. \tag{5.55}$$

Note that unlike the lumped element case, J_{source} is a constant specified by the user and has no dependence on the boundary potential V_s (the V_s dependence is buried in the summation). Since the full Newton's method is used for the solution of all equations in the program, no degradation of convergence has been observed for current controlled simulations.

The choice of a voltage or current bias is mostly a function of the dI/dV slope; this is explained in more detail in Sec. 4.1. For regions where dI/dV is small, the voltage boundary condition is definitely preferable; this occurs frequently below the turn-on voltage or under reverse bias conditions. If the opposite is true, then the current bias is preferable.

It is not uncommon for the negative resistance regime of certain devices to have a slope dI/dV very close to zero. Such behavior should be considered when using a

current source to trace out an entire I-V curve. It might be preferable to switch back to a voltage source after passing the trigger point.

5.3 Pure Resistor Regions and Metal Macros

For resistors and metals, we assume the following conditions are true:

- Charge neutrality: we assume that the space charge at any point is always zero even when current flows.
- Zero diffusion current: the distribution of carriers is uniform enough that only drift current exists.
- Equilibrium carrier density: we assume that carrier density stays constant even when current flows so that no recombination occurs.
- Uniform material properties throughout the layer.

Under such assumptions, the Poisson and current continuity equations reduce to:

$$\nabla \cdot \left(\frac{\epsilon_0 \epsilon_{dc}}{q} \nabla V \right) = 0 \tag{5.56}$$

$$\nabla \cdot J = 0 \tag{5.57}$$

These can be combined into a single equation by writing the drift current as:

$$J = \sigma \nabla V \tag{5.58}$$

Therefore, only the Poisson equation needs to be solved in metal/resistor regions.

Model History

Crosslight's metal models have undergone a number of major changes over the years so it is important to document the history of these changes for existing users.

Pseudo-metal semiconductor

Prior to the 2007.3 version, the standard metal model described above was used to represent resistors. However, this model had some limitations. For semiconductor regions, the drift-diffusion model solves three equations so solving a single equation

for certain points did not integrate very well other models such as our small-signal AC solver or the quantum tunneling code.

To simplify integration of metals with these models, starting from version 2007.3, Crosslight's device simulators approximated a resistor region as a special kind of semiconductor with a heavy carrier mass (∼10) and very small bandgap (0.1 eV). This approach allowed the semiconductor to duplicate many properties of metals including Fermi level pinning. However, the need to define an equivalent carrier mobility (to match the desired metal resistivity) introduced a dependency on the carrier density. As this was controlled by the increased DOS mass, there was a mismatch with the conduction/transport mass used in the quantum tunneling formulation which required further corrections.

Feedback from users has ultimately demonstrated that this pseudo-metal semiconductor approach has its own set of limitations, especially with thicker layers. As of the October 2014 version, this approach is now considered obsolete and the pre-2007 standard model has been restored. However, our AC and quantum tunneling models have now been improved and may be used in conjunction with resistors.

Band alignment

Starting with version 2013, a subroutine was added to automatically adjust the work function of resistors and align it with the affinity of the neighboring semiconductor. This was done to facilitate the use of metals which, in most applications, are used for ohmic contacts. However, since thermionic emission is used at all internal boundaries, Schottky contacts of poor quality may be formed if the affinity of the resistor macro is not adjusted properly. An example of this is ITO which can be either a p (for GaN) or n (for OLED) contact metal depending on the application: the affinity needed to successfully inject the correct carriers changes greatly in these two cases.

The pre-2013 Schottky-like behavior can be recovered using the **no_auto_workfunction** statement. This may be necessary in cases where the Schottky behavior is expected, when the effect of the metal's affinity must be studied, or in thermal simulations where affinity shifts due to temperature may ruin the automatic alignment process.

Interaction with quantum tunneling models

One should exercise caution when setting up quantum tunneling between a metal macro and a semiconductor barrier. When using the propagation matrix approach, the program uses the default density of state (DOS) mass from the macro to define the tunneling probability; put differently, tunneling depends on the kinetic energy of an individual particle. However, the DOS mass is a measure of the shape of the $E(k)$ dispersion relation and describes "how many" carriers there are in a band rather than

"how fast" those carriers are traveling. These two mass values may not always be the same.

Luckily, the DOS mass is not needed in the current version of the metal/resistor macros and unlike previous the pseudo-metal approach, we no longer need to calculate an equivalent carrier mobility. Instead, we use experimental data and the mobility-carrier density relationship is completely abstracted away:

$$\sigma = \frac{1}{\rho} = q(\mu_n n + \mu_p p) \tag{5.59}$$

The tunneling transport mass in our model is thus defined in the following way:

- 1 Internally, the DOS mass of metal/resistor macros is set at a default value of 0.1 for n and p. This can be overridden by explicitly declaring the DOS mass in the resistor macro.
- 2 The transport mass is calculated using a ratio of 1:1 with the DOS mass. This ratio may be overridden using the commands **cond_dos_mass_ratio_n** and **cond_dos_mass_ratio_p**.

5.4 Bandgap Narrowing Effect

It is observed experimentally that the shrinkage of bandgap occurs when impurity concentration is particulary high, eg. $n_{imp} > 10^{23} m^{-3}$. This effect is so called bandgap narrowing effect which is ascribed to the emerging of the impurity band formed by the overlaped impurity states. The bandgap narrowing model proposed by Slotboom is given by

$$\Delta E_g = E_{ref} \left\{ \ln \frac{n_{imp}}{n_{ref}} + \sqrt{\ln^2 \frac{n_{imp}}{n_{ref}} + 0.5} \right\} \tag{5.60}$$

where E_{ref}, n_{ref}andn_{imp} represent energy parameter, density parameter and impurity concentration, respectively.

In case of silicon, E_{ref} and n_{ref} were obtained by fitting experiment, which yeilds

$$E_{ref} = 0.009 \ [V] \tag{5.61}$$
$$n_{ref} = 10^{23} \ [m^{23}] \tag{5.62}$$

It should be noted that the bandgap narrowing effect is doping dependent effect, whereas the many body effect, which reduces band gap as well, is carrier dependent effect.

It should also be pointed out that the macro system in Crosslight software provides an easy way for user to define any kind of bandgap narrowing model since dopant density is an internal variable automatically passed to any macro functions, including the bandgap.

5.5 Trap Models

A deep level trap located with the semiconductor bandgap is described by both the charge property (deep donor or deep acceptor) as it appears in Poisson's equation and by a rate equation with capture and escape terms:

$$R_n^{tj} = c_{nj} n N_{tj}(1 - f_{tj}) - c_{nj} n_{1j} N_{tj} f_{tj}, \quad (5.63)$$

$$R_p^{tj} = c_{pj} p N_{tj} f_{tj} - c_{pj} p_{1j} N_{tj}(1 - f_{tj}) \quad (5.64)$$

Traps can be defined as either localized defects using the **doping** statement or as an integral part of a material using the **material** statement. The latter method is especially convenient when dealing with amorphous materials like a-Si:H. When combining the two methods, the following rules are used:

- 1. Load all macros and initialize trap properties and trap level models if any **material** statements define traps.
- 2. Get trap information from **doping** statement regarding trap concentration and level. This will affect all materials in a given area since doping is done at the mesh level and does not distinguish by material.
- 3.
  - If (2) exists for traps, we should now have density, level and charge-types. Calculate capture cross section according to material lifetime for trap_1 only.
  - If (2) does NOT exist, assume a low mid-gap donor of 1.e6 1/m3. Then, calculate capture cross section according to material lifetime for trap_1 only.
- 4. Check **trap_conc_i** commands. If they exist, we ADD that concentration to that obtained in (2).
- 5. Check for **trap_level_i** statements which override values obtained from (2) and (3).
- 6. Check for **trap_ncap_i** and **trap_ncap_i** statements that override values obtained from (3).

The software is limited to only a few species of traps (defined by numeric labels) but a single species of trap can represent many different trap levels within the bandgap. In the extreme of densely populated traps such as those appearing in amorphous materials, it is more convenient to use a continuous function to describe the energy distribution of the trap levels.

As of the 2015 version, the software supports Gaussian, exponential tail and uniform trap distributions.

Trap Photo-Excitation

When the photon energy of light is less than the semiconductor bandgap, it is still possible to generate photo-carriers if deep level traps are sensitive to light. Electrons and holes being trapped in the deep level trap centers require less than bandgap energy to be excited to the conduction or valence bands. We shall refer to this process as trap photo-excitation.

In such a case, the Schokley-Read-Hall recombination terms of the drift-diffusion must be revised to include photo-carrier generation terms due to emission from traps (G_{tn} and G_{tp}):

$$R_n^{tj} = c_{nj} n N_{tj}(1 - f_{tj}) - c_{nj} n_{1j} N_{tj} f_{tj} - G_{tn}, \tag{5.65}$$

$$R_p^{tj} = c_{pj} p N_{tj} f_{tj} - c_{pj} p_{1j} N_{tj}(1 - f_{tj}) - G_{tp} \tag{5.66}$$

We will derive the generation term G_{tn} from the consideration that it must be proportional to trap density and electron occupancy. It must also be proportional to light intensity. Let us recall the photon generation term of direct electron-hole pairs:

$$G_{eh} = v_g S \alpha \tag{5.67}$$

where v_g is the group velocity of light, S the photon density and α the absorption coefficient.

In analogy, we can write down the trap excitation term as:

$$G_{tn} = v_g S N_{tj} f_{tj} \sigma_{xn} \tag{5.68}$$

where σ_{xn} has the unit of area and shall be defined as the trap photo-excitation cross section. Please note that this cross section times the amount of trapped electrons (i.e., $N_{tj} f_{tj}$) may be compared with usual bulk material absorption coefficient in units of 1/m.

Similarly, trap excitation for holes is given by:

$$G_{tp} = v_g S N_{tj}(1 - f_{tj}) \sigma_{xp} \tag{5.69}$$

To activate this model, the user should use the **trap_excitation** statement.

Chapter 6

NUMERICAL TECHNIQUES AND 3D SIMULATION

This chapter discusses numerical issues related to solving the drift-diffusion equations on a discretized 2/3 dimensional finite element mesh. The 3D mesh may be regarded as a collection of 2D planes with carefully constructed plane-to-plane finite element connectivity. The reduction of 3D into 2D in case of cylindrical symmetry is also discussed.

6.1 Numerical Techniques

6.1.1 Introduction

All Crosslight software packages involve solving the drift-diffusion equations along with a few others, mostly related to the optical part of the device simulation. Since a substantial amount of computation time is spent on the DD equations, we shall concentrate mainly on the numerical techniques related to them here.

Equations (5.1) to (5.3) describe the electrical behavior of the bulk or quantum well semiconductor devices. Poisson's equation (Equation 5.1) governs the electrostatic potential and the continuity equations (Equations (5.2)-(5.3)), govern the carrier concentrations.

These differential equations are discretized as described in the next section. The resulting set of equations is coupled and nonlinear. Consequently, there is no method to solve the equations in one direct step. Instead, solutions must be obtained by a nonlinear iteration method, starting from some initial guess. The solution methods are detailed in subsection 6.1.3, and the choice of initial guess is explained in subsection 6.1.4.

6.1.2 Discretization

To solve the device equations on a computer, they must be discretized on a simulation grid. That is, the continuous functions of the PDE's are represented by vectors of function values at the nodes, and the differential operators are replaced by suitable difference operators. Instead of solving for four unknown functions (potential, electron and hole concentrations, and the electron or hole energy), the simulator solves for $3N$ or $4N$ unknown numbers, depending on the model choice, where N is the number of grid points.

The key to discretizing the differential operators on a general triangular grid is the box method [10]. Each equation is integrated over a small polygon enclosing each node, yielding 4N nonlinear algebraic equations for the unknown potential, concentrations and the wave amplitude. The integration equates the flux into the polygon with the sources and sinks inside it, so that conservations of current and electric flux are built into the solution.

The integrals involved are performed on a triangle by triangle basis, leading to a simple and elegant way of handling general surfaces and boundary conditions. In this case the integral is simply replaced by a summation of the integrand evaluated at the node multiplied by the area surrounding it.

In the case of the continuity equations, the carrier fluxes must be evaluated with care; the classic finite difference formulas are modified as first demonstrated by Scharfetter and Gummel (SG-formula) in 1969 [11]. From the user's point of view, the discretization is completely automatic and no intervention is required.

For the hydrodynamic model, the SG-formula for discretization must be modified. The details are worked out in the Appendix.

6.1.3 Newton's method

We discuss the numerical solution of drift-diffusion equations. In Newton's method, all of the variables in the problem are allowed to change during each iteration, and all of the coupling between variables is taken into account. Due to this, the Newton algorithm is very stable, and the solution time is nearly independent of bias conditions.

The basic algorithm is a generalization of the Newton-Raphson method for the root of a single equation. Equations (5.1)–(5.3) can be written as:

$$F_v^j(V^{j1}, E_{fn}^{j1}, E_{fp}^{j1}) = 0, \tag{6.1}$$

$$F_n^j(V^{j1}, E_{fn}^{j1}, E_{fp}^{j1}) = 0, \tag{6.2}$$

$$F_p^j(V^{j1}, E_{fn}^{j1}, E_{fp}^{j1}) = 0, \tag{6.3}$$

where j runs from 1 to N, and $j1$ includes j itself plus its surrounding mesh points.

Equations (6.1) to (6.3) represent a total number of $3N$ equations, which is sufficient to solve for $3N$ variables: (V^1, E^1_{fn}, E^1_{fp}, V^2, E^2_{fn}, E^2_{fp}, ..., V^N, E^N_{fn}, E^N_{fp}).

Once the equations are discretized in the above form, standard Newton techniques can be used to solve for them. These involve the evaluation of the Jacobian matrix to linearize Equations (6.1) to (6.3), followed by a linear solver (involving LU factorization of the matrix) and finally, nonlinear iteration to get the final solution. Since the Jacobian matrix is sparse, sparse matrix techniques are used to improve the computation speed.

The major acceleration of a Newton iteration is the Newton-Richardson method, whereby the Jacobian matrix is only refactored when necessary. When it is not necessary to factorize, the iterative method using the previous factorization is employed. The iterative method is extremely fast provided the previous factorization is reasonable. Frequently the Jacobian need only be factorized only once or twice per bias point using the Newton-Richardson method, as opposed to the twenty to thirty times required in the conventional Newton method. The decision to refactor is made on the basis of the decrease per step of the maximum error of both the equation residuals and the variable differences. This mechanism has been automated within the program and no user intervention is needed.

6.1.4 Initial guess and adaptive bias stepping

Several types of initial guess solutions are used in the simulator. The first is the simple charge neutral assumption used to obtain the first (equilibrium) bias point. For the wave equation, the guess solution is a delta function at the center of the active region. This is the starting point of any device simulation.

Any later solution with applied bias needs an initial guess of some type, obtained by modifying one or two previous solution(s). When only one previous solution is available, the solution currently loaded is used as the initial guess, modified by setting the applied bias at the contact points. The same is true for the wave equation. When two previous solutions with two different biases are available, it is possible to obtain a better initial guess by extrapolating the two previous solutions All of these steps are automated within the program and no user intervention is needed.

A schematic of the biasing strategy is given in Figure 6.1.

As in any Newton method, convergence strongly depends on the wise choice of the initial guess solution. In principle, the Newton nonlinear iteration should always converge as long as the initial guess is close enough to the solution. The closer the initial guess is to the solution, the fewer nonlinear iterations are required to reach convergence. The program takes this fact into account and implements an adaptive method to control the bias step after the successful convergence of the previous solution.

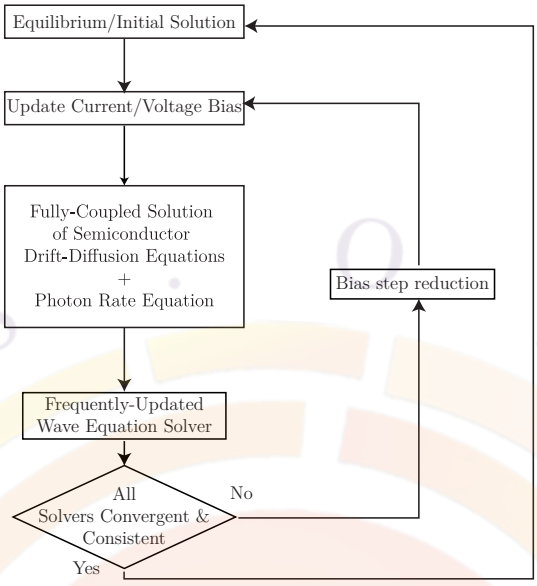


Figure 6.1: Flow diagram of how the simulator performs a typical simulation.

Assuming that the previous solution takes K_1 number of iterations for a bias step of ΔV_1 to reach convergence and the user believes (by experience) that the optimal number of iterations should be K_0, the current bias step ΔV_2 is adjusted to $\Delta V_2 = \Delta V_1 K_0/K_1$. In this way an optimal step can be determined such that the nonlinear Newton iteration is controlled to converge within K_0 iterations. Again this procedure is completely automated for the user.

One common cause of convergence problems in a Newton method is oscillations in the solution. This usually occurs when the initial guess is poor and the solver overshoots the solution. In order to prevent this, a damping value has been implemented which prevents the solver to change the solution too much between successive iterations. This value is defined in the **newton_par** statement. A small damping value will damp the oscillations effectively but may cause slower convergence. A large value allows faster convergence when the initial guess is poor but carries a larger risk of oscillations.

Another important topic for the Newton solver is the choice of solver variables: please refer to section 4.7 for details.

6.1.5 Transient simulation

The method used to discretize the differential equations in time is the backward Euler method. Its basic approximation is the following equation:

$$\frac{\partial S}{\partial t} = \frac{S(t) - S(t - \Delta t)}{\Delta t}. \tag{6.4}$$

This method has the advantages of being highly stable and recovers the steady state solution in the limit $\Delta t \to \infty$.

Strictly speaking, the simulation system has more than 5 sets of equations when the trap dynamic equation (Equation (5.11)) is also included in a transient simulation. However, this equation is only dependent on local variables and can be solved before the major Newton step involving all mesh points.

Once all the equations are discretized in terms of solution of the previous time step, one can treat the solution of the present time step using the same method as for the steady state solution, *i.e.* one can linearize the discretized equations and use Newton's method to solve them.

6.1.6 Mesh issues

Correct allocation of the grid is a crucial issue in device simulation. The number of nodes in the grid (N) has a direct influence on the simulation time, where the number of arithmetic operations necessary to achieve a solution is proportional to N^p where p usually varies between 1.5 and 2.

Because the different parts of a device have very different electrical and optical behavior, it is usually necessary to allocate a fine grid to some regions and a coarse grid to others. Whenever possible, it is desirable not to allow the fine grid in some regions to spill into regions where it is unnecessary, in order to maintain a reasonable simulation time. For more mesh allocation suggestions, please refer to sections 4.5 and 4.6.

The first step in mesh generation is to specify the device boundaries and the region boundaries for each material. The program uses triangles and trapezoids as the basic building blocks to describe an arbitrary device. Furthermore, the edge of each polygon can be bent to the desired shape.

The basic operation in the mesh generator after the boundaries have been defined is to draw lines parallel to the edges of the polygons (see Figure 6.2). This way one obtains, smaller trapezoids which can be bisected into two triangles (the basic elements in the finite element method).

Another basic operation is doubling the mesh density in a specified region as shown in the example of Figure 6.2. This is accomplished by drawing a new line between the old mesh lines. Repeating this operation, one can manually generate a mesh with any variation of mesh densities.

The simulator has provided another practical approach to automatic mesh generation and refinement. The mesh is refined according to the variation of specified physical quantities. For example, in a case where the potential variation between two adjacent nodes is greater than a value specified by the user, the program will automatically

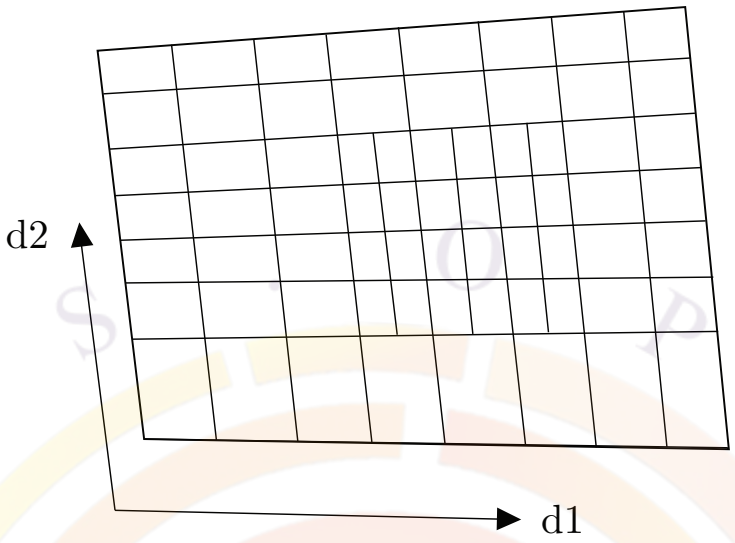


Figure 6.2: Schematic for the basic operations in the mesh generator. Two mesh lines in the d_2 direction, are used to double the mesh near the center of the polygon.

allocate additional mesh points between the two nodes. More detail can be found in the description of the statement **refine_mesh** in the reference section.

6.2 The Cylindrical Coordinate System

6.2.1 Introduction

Since many devices are fabricated with cylindrical symmetry (e.g. VCSELs, photodetectors), it is a good idea to establish a cylindrical coordinate system for device simulation. By making use of the rotational symmetry, we can reduce the three-dimensional coordinate system to a two-dimensional one. To do so, we first consider our differential equation systems under a general coordinate system. Then, we apply the theory to the cylindrical system.

6.2.2 Some general considerations for an arbitrary coordinate system

Our purpose is to re-write the drift-diffusion equation, usually written in cartesian coordinates, (x,y,z), in the new coordinate system (q_1, q_2, q_3) which are orthogonal to each other. We start by considering a small distance along each of the three

coordinates:

$$ds_j = H_j dq_j, H_j = \sqrt{\left(\frac{\partial x}{\partial q_1}\right)^2 + \left(\frac{\partial y}{\partial q_2}\right)^2 + \left(\frac{\partial z}{\partial q_3}\right)^2}. \tag{6.5}$$

Then the gradient of a scalar variable u is given by:

$$\nabla u = \frac{1}{H_1}\frac{\partial u}{\partial q_1}\mathbf{e_1} + \frac{1}{H_2}\frac{\partial u}{\partial q_2}\mathbf{e_2} + \frac{1}{H_3}\frac{\partial u}{\partial q_3}\mathbf{e_3} \tag{6.6}$$

where $\mathbf{e_j}$ is used to denote the unit vector for each of the new coordinates.

Let us derive the expression for the divergence, $\nabla \cdot \mathbf{A}$ of an arbitrary vector $\mathbf{A}$. We consider a small volume defined within from q_1 to $q_1 + dq_1$, q_2 to $q_2 + dq_2$, and q_3 to $q_3 + dq_3$. The out-going flux ($\mathbf{A}$) though the surfaces at q_1 and $q_1 + dq_1$ of this small volume is given by:

$$(A_1 H_2 H_3)|_{q_1+dq_1} dq_2 dq_3 - (A_1 H_2 H_3)|_{q_1} dq_2 dq_3 = \frac{\partial}{\partial q_1}(A_1 H_2 H_3) dq_1 dq_2 dq_3 \tag{6.7}$$

Similarly, the flux through surfaces at q_2 and $q_2 + dq_2$ is given by:

$$\frac{\partial}{\partial q_2}(A_2 H_3 H_1) dq_1 dq_2 dq_3 \tag{6.8}$$

and the flux through surfaces at q_3 and $q_3 + dq_3$ is given by:

$$\frac{\partial}{\partial q_3}(A_3 H_1 H_2) dq_1 dq_2 dq_3 \tag{6.9}$$

The divergence is defined as the the total out-going flux divided by the volume under consideration:

$$\nabla \cdot \mathbf{A} = \frac{1}{H_1 H_2 H_3}\left[\frac{\partial}{\partial q_1}(H_2 H_3 A_1) + \frac{\partial}{\partial q_2}(H_3 H_1 A_2) + \frac{\partial}{\partial q_3}(H_1 H_2 A_3)\right] \tag{6.10}$$

6.2.3 Cylindrical coordinate system

The relation between the old (x_1, y_1, z_1) and the new coordinate system (r, θ, z_r) is given by:

$$\begin{aligned} x_1 &= r cos\theta \\ y_1 &= r sin\theta \\ z_1 &= z_r \end{aligned} \tag{6.11}$$

To avoid confusion with the (x,y,z) system used in our simulation software, we have used a new set of symbols here. In our simulator, x, y are the lateral and vertical

coordinates of a 2D plane and z is a depth coordinate used to stack planes. In the cylindrical system, the z_r axis is usually normal to the layers and corresponds to y so that r corresponds to x.

It is easy to find that:

$$H_r = 1, H_\theta = r, H_z = 1. \tag{6.12}$$

Therefore, the gradient is given by:

$$\nabla u = \frac{\partial u}{\partial r}\mathbf{e_r} + \frac{1}{r}\frac{\partial u}{\partial \theta}\mathbf{e}_\theta + \frac{\partial u}{\partial z_r}\mathbf{e_z} \tag{6.13}$$

and the divergence is given by:

$$\nabla \cdot \mathbf{A} = \frac{1}{r}\left[\frac{\partial}{\partial r}(rA_r) + \frac{\partial}{\partial \theta}(A_\theta) + \frac{\partial}{\partial z_r}(rA_z)\right] \tag{6.14}$$

or

$$\nabla \cdot \mathbf{A} = \frac{1}{r}\frac{\partial}{\partial r}(rA_r) + \frac{1}{r}\frac{\partial}{\partial \theta}A_\theta + \frac{\partial}{\partial z_r}A_z \tag{6.15}$$

6.2.4 Application to devices with cylindrical symmetry

For device with cylindrical symmetry, $\frac{\partial}{\partial \theta}$ yields zero. The operators of gradient and divergence become:

$$\nabla u = \frac{\partial u}{\partial r}\mathbf{e_r} + \frac{\partial u}{\partial z_r}\mathbf{e_z} \tag{6.16}$$

$$\nabla \cdot \mathbf{A} = \frac{1}{r}\left[\frac{\partial}{\partial r}(rA_r) + \frac{\partial}{\partial z_r}(rA_z)\right] \tag{6.17}$$

Therefore it is easy to see that in the cylindrical system, all we need to do is to replace $\mathbf{A}$ by r$\mathbf{A}$ and multiply $\frac{1}{r}$ in front of the differential operators.

Our next task is to convert the new differential equation into a suitable discretization scheme for the cylindrical system. Let us recall how we discretize the drift-diffusion equation in the old coordinate system. We evaluate the gradient (the electric field or the current density vector) for each boundary of the finite box which we create for the neighboring elements. We then sum up the total flux and divide it by the area of the finite box. In the new system, we need to multiply the flux by r which is evaluated at the boundary point. We also divide the total flux by the area of the finite box and by r evaluated at the center of the box (or the center point).

It is obvious that we should keep in mind that the missing dimension is no longer infinite but has rotational symmetry. For example, we need to integrate current density over the 2π radians to get the total current.

6.3 3D Modeling in Semiconductors

6.3.1 Introduction

For historical reasons and previous limitations on available computer power, most of the Crosslight Software packages are written for a two-dimensional cross-section of a semiconductor device. Recent progress has allowed computer power to increase significantly so we can now consider doing 3D simulations on a desktop PC.

Some outstanding issues must be resolved to extend the solver from 2D to 3D:

- The user input/interface should be a natural extension of a 2D simulation. All material macros in 2D should be usable in 3D. All physical models and boundary conditions must be extendable to 3D.
- A useful 3D simulation should be able to handle geometry and material variation in the z-dimension easily.
- Smart memory management scheme must be developed so that the 3D package can be run on anything from a laptop to a supercomputer.

We have successfully overcome most of these difficulties and created an option for 3D modeling. We call this capability "3D-Current Flow" or "3D-Flow" since it allows current to flow both in all three dimensions. We will explain the design ideas behind this model in the following subsections.

6.3.2 From 2D to 3D

Three-dimensional mesh generation is very complex and is still the subject of intense research by many groups. However, we have a two-dimensional mesh generation and refinement system which works very well: it is natural to extend this system as far as we can.

As usual, we find the parallel cross sections of a device that has the most variation in physical variables and define them as x, y planes. Within these planes, we use the same 2D mesh generation routines as before. The z direction will have a relatively slower variation and can be sparsely meshed to reduce the overall number of mesh points: every z mesh point will be accompanied by a 2D mesh plane.

For example, a ridge waveguide laser would use the waveguide cross-section as the x, y direction and the z axis would be the longitudinal propagation axis. On the other hand, an LED with complex electrode shapes would define the x, y planes to be parallel to the electrodes and the z axis would be perpendicular to the QW layers. In a cylindrical system, 3D simulations imply a lack of rotational symmetry and the x, y (or r, z_r) planes would stack together in the θ direction.

6.3.3 3D structure and material input

A 3D simulation in .sol is always defined by using the **3d_solution_method** statement with the option 3d_flow=yes. Other options can also be added to control the way the mesh planes connect to each other as will be explained later in this chapter.

The next step of the input is to define regions that have the same material properties and composition. These are called 'segment" or "z-segment" and are defined by the **z_structure** statement. A segment can have multiple mesh planes: these will share the same material properties such as bandgap and doping but physical variables such as potential and carrier concentrations are allowed to change between planes.

To define a 3D volume, at least 2 mesh planes must be defined. However, at least 3 mesh planes must be present to get a variation along the z-axis. Segments containing a single mesh plane are allowed but must have zero thickness: additional planes and segments will be needed to define a 3D volume.

After all the **z_structure** statements have been issued, multiple **load_mesh** statements must be issued. Each of these statements must use the zseg_num parameter to identify the segment they belong to. In the special case where there is only a single segment, this can be omitted and $zseg\_num = 1$ will be assumed.

After the mesh has been loaded, the material properties for each segments must be defined. This must be done inside blocks bracketed by **begin_zmater** ... **end_zmater**. These blocks must also be numbered by zseg_num with the exception of the case with a single segment: in this case, $zseg\_num = 1$ will be assumed and the _zmater statements can be omitted.

Material properties that need be defined in this way include:

- material declarations: **load_macro** and **define_material**
- active region declarations: **active_reg** and **get_active_layer**
- doping statements
- contact definitions

Note that for material declarations, a macro should be loaded only once, in the segment where a material first appears. Further occurrences of this material should invoke **define_material** instead of **load_macro**.

Just as materials can reoccur between segments, so can contacts the same boundary conditions applies everywhere and the electrode current is also summed up for all segments. If each segment is to be biased independently from the other, then different contact numbers must be used. For example, it is possible for a tunable DBR laser to have a shared bottom electrode (#1) with split top contacts for the gain (#2), tuning (#3) and DBR (#4) segments.

The 2D .layer files can be used in a 3D simulation to facilitate this input: the .mater and .doping files it generates can be included inside the _zmater blocks. However, when processed by itself, a layer file will always number materials 1,2,3,... as it encounters them which can cause conflicts between segments. To avoid this, make sure to use the **previous_layer** statement to refer to the preceding segment's .layer file: this will fix numbering issues and will also detect materials that have already been declared so it will issue **define_material** statements where appropriate.

However, contacts in each 2D layer file should be numbered according to the final 3D structure and whether or not they will be shared between segments.

6.3.4 3D mesh generation

As explained above, the three-dimensional mesh consists of a series of two-dimensional mesh planes stacked together. The position of these mesh planes is determined by the length of the segments and the number of planes that have been assigned. This number must be assigned manually by the user, just like the number of mesh points in a column or layer.

When a segment has multiple planes, there will usually be planes at the beginning and end of the segment. If there are multiple segments, this can cause a problem since two mesh planes cannot exist in the same z position. The software has two ways of dealing with this, depending on the value of the xy_compatible_mesh parameter in **3d_solution_method**:

- If set to "no" (the default), mesh planes on either side of the segment boundary will be shifted by a small value.
- If set to "yes", then one of conflicting the planes will be eliminated and no shifting is required.

Care should be taking when using xy_compatible_mesh=yes. If there is a hetero-junction at a segment boundary, then eliminating the extra plane can have disastrous consequences on convergence. Just like the 2D case, the closely spaced mesh points on either side of the junction are needed to get convergence.

For the other planes inside a segment, their position can be also be controlled via mesh point ratios: the **z_structure** statement supports a feature similar to that of **put_mesh** in the 2D mesh generator.

6.3.5 Discretization in 3D

As explained in subsection 6.1.2, every mesh point has a polygon associated with it. To connect mesh points between planes, we extrude this polygon in either direction

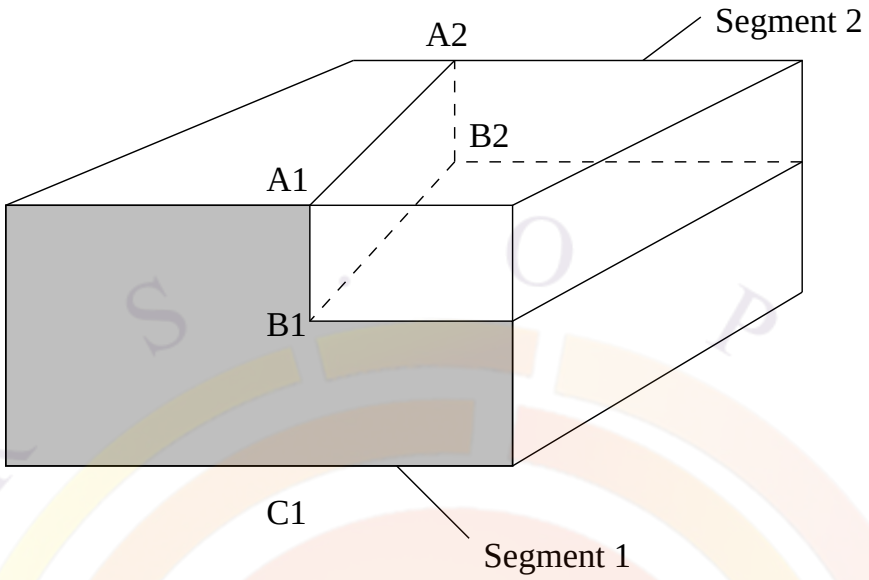


Figure 6.3: Schematic of taper lines between two different segments.

to the midpoint of the preceding and following mesh plane. This forms prism-like 3D objects that are different from the traditional tetrahedral elements used in 3D FEM.

Flux conservation is consistent with the box method described earlier: connecting mesh points are assigned part of the flux based on the overlap between the two extruded polygons. A projection is used to account for the case where the two mesh planes are not parallel.

6.3.6 Taper structures

To explain how tapers are handled in the software, we need to introduce the concept of “taper lines" as illustrated in Fig. 6.3.

Consider a structure with a taper extending from points A1-B1 on segment 1 to A2-B2 on segment 2. We define a “taper line” A1-C1 on segment 1 and a corresponding taper line A2-C2 (point C2 is hidden behind the figure). These taper lines are used to extend a particular boundary from one plane to the corresponding boundary on the other.

This will divide the mesh on either side of the taper line and force mesh points to connect across planes without crossing the taper lines: this is especially important when dealing with semiconductor/insulator interfaces. As a result, the extrusion process described above no longer follows the z axis: it follows taper lines.

To define a taper in .sol, two segments must be defined with the taper_pointsj

parameters: multiple sets of taper points can be used to define multiple taper lines. The two segments must not overlap and the taper region will extend in the empty space between the last plane of the first segment and the first plane of the second segment.

Note that a very common misconception is that material properties in tapers are linearly interpolated between planes: for the electrical calculations, this is completely untrue. Tapers only control the connectivity between mesh planes: actual mesh planes must be assigned at various z positions in order to sample the variation of physical properties.

6.3.7 3D simulator control

Once the 3D material data and mesh has been created, a 3D simulation is relatively straightforward. The .sol file works the same as before and bias can still be applied to specified electrodes. When the Newton solver is called, all mesh planes will be solved simultaneously to obtain the 3D solution. The main difference is that 2D simulations use currents in A/m units (since the z depth is not known). A 3D simulation, just like a cylindrical simulation, can compute the full current in Amperes.

For simulations with split contacts between segments, we advise caution when biasing as you may inadvertently create a short between neighboring electrodes in certain situations. This can usually be avoided by biasing multiple electrodes simultaneously.

For plotting results, the .plt file can still be used: plotting statements such as **lplot_xy**, **lplot_xyz**, vplot_xy, etc... can be used to view 3D data. CrosslightView can also plot data from 3D simulations.

6.4 3D modeling in PICS3D

PICS3D adds an extra complication to the 3D modeling. As will be discussed in Chapter 16 and Sec. 18.4, the lasing behavior is controlled by an additional set of equations which depend on the longitudinal round-trip gain (RTG). These equations are based on the propagation of the optical mode in the cavity and are most often written using transfer matrix formalism.

As such, RTG evaluation requires gain and index profiles at various points in the cavity. These points can be considered an “optical mesh” and do not necessarily match the position of the electrical mesh planes described in the previous section. The optical mesh always follows the direction of the light propagation in the cavity so by the usual convention, it always follows the z-direction. For edge emitting devices like DFB lasers, the z-direction is defined as the direction in which x-y mesh planes are stacked: we will assume we are dealing with such a device for the rest of this

section.

With this in mind, we can consider how to evaluate the RTG equations. The problem here is one of consistency: to calculate the propagation, the gain profile is needed which requires knowing the local photon density and the associated longitudinal hole burning. However, the photon density itself is only known by calculating the RTG.

In older versions of PISC3D, the solution to this problem was to use a quasi-3D approach and tabulate the gain at various photon densities. The RTG equations were then solved separately from the main Newton solver to get the photon density and longitudinal gain profile. However, this method sometimes led to convergence issues.

As of version 2008, PICS3D couples the RTG equations and solves them in the main Newton solver. The RTG equations are still use the "optical mesh" but the modal gain and index for the propagation are directly interpolated from the electrical mesh at every iteration of the Newton solver. This means that all PICS3D simulations must be done in full 3D. For edge-emitting devices, PICS3D relies on the ability to solve multiple electrical mesh planes simultaneously to interpolate the gain and index profiles on the optical mesh. This increased self-consistency makes the model fully 3D and greatly increases the convergence and stability of the software.

As a result, all edge-emitting laser simulations in PICS3D must use the **3d_solution_method** statement with **3d_flow=yes**. However, the **z_connect** option is available to turn on or off the electrical coupling between mesh planes: only the ability to interpolate the gain from multiple mesh planes is actually required. In many devices, the longitudinal current contribution is negligible so turning off the electrical connection between mesh planes has a few advantages:

- it can often result in a smaller and more stable sparse matrix since the connecting elements between planes will be left out of the sparse matrix entirely
- it can avoid unintentionally shorting contacts between segments in certain bias configurations

6.4.1 3D VCSEL modeling in PICS3D

For cylindrical devices such as VCSELs, the z-axis is as defined in Sec. 6.2. This means that the propagation is normal to the QW layers and the optical mesh is perpendicular to what is used in edge-emitting devices. In addition, rotational symmetry implies that only a single mesh plane is required to do a full 3D model: **3d_solution_method** is not required for most cases.

However, the same basic principles of the coupled RTG method still apply: optical properties are derived from the electrical mesh points and used to calculate the mode propagation. The coupled RTG equations are solved in the main Newton

solver alongside the rest of the electrical problem.

PARS OPTICS GROUP

Chapter 7

AC ANALYSIS AND HIGH FREQUANCY PARAMETERS

For many users of Crosslight simulation software, assessment of high frequency performance of their devices is a key reason of the modeling. Analysis of high frequency behavior of a semiconductor device is more than solving the wave propagation in a passive medium. The response of a semiconductor to high frequency modulation is rather complex and requires special transformation of the drift-diffusion equations. This chapter gives an outline of the AC analysis theory for semiconductor devices.

7.1 Theory

7.1.1 Introduction

AC small signal analysis for semiconductor equations was first developed by Laux [12] in 1985. A modified version of this technique including the deep level trap model was later developed by Li and Dutton [13]. We shall follow their formulation in this section. We derive the formulas in subsection 7.1.2. The numerical techniques are given in section 7.2.

7.1.2 Basic formulas

We re-write the basic semiconductor equations and trap dynamic model as follows:

$$-\nabla\cdot\left(\frac{\epsilon_0\epsilon_{dc}}{q}\nabla V\right)=-n+p+N_D(1-f_D)-N_Af_A+\sum_j N_{tj}(\delta_j-f_{tj}) \tag{7.1}$$

$$\nabla\cdot J_n-\sum_j R_n^{tj}-R_{sp}-R_{st}-R_{au}+G_{opt}(t)=\frac{\partial n}{\partial t}+N_D\frac{\partial f_D}{\partial t} \tag{7.2}$$

$$\nabla\cdot J_p+\sum_j R_p^{tj}+R_{sp}+R_{st}+R_{au}-G_{opt}(t)=-\frac{\partial p}{\partial t}+N_A\frac{\partial f_A}{\partial t} \tag{7.3}$$

$$R_n^{tj}-R_p^{tj}=N_{tj}\frac{\partial f_{tj}}{\partial t} \tag{7.4}$$

where:

$$R_n^{tj}=c_{nj}nN_{tj}(1-f_{tj})-c_{nj}n_{1j}N_{tj}f_{tj} \tag{7.5}$$

$$R_p^{tj}=c_{pj}pN_{tj}f_{tj}-c_{pj}p_{1j}N_{tj}(1-f_{tj}) \tag{7.6}$$

The basic variables involved in the AC analysis are V, the potential; n and p, the electron and hole concentrations, respectively; and f_{tj}, the trap occupancy of the jth trap species. To simplify our derivation, we use the following notation to represent Equations (7.1) to (7.4):

$$F_1(V,U_n,U_p,f_{tj})=0 \tag{7.7}$$

$$F_2(V,U_n,U_p,f_{tj})=\left(\frac{\partial n}{\partial U_n}+N_D\frac{\partial f_D}{\partial U_n}\right)\frac{\partial U_n}{\partial t} \tag{7.8}$$

$$F_3(V,U_n,U_p,f_{tj})=\left(-\frac{\partial p}{\partial U_p}+N_A\frac{\partial f_A}{\partial U_p}\right)\frac{\partial U_p}{\partial t} \tag{7.9}$$

$$H_j(U_n,U_p,f_{tj})=N_{tj}\frac{\partial f_{tj}}{\partial t} \tag{7.10}$$

where we have changed the variables n and p into U_n and U_p, their respective normalized quasi-Fermi levels. For the electron or hole transport equations (F_2 or F_3) the first term on the right hand side represents $\frac{\partial n}{\partial t}$ or $\frac{\partial p}{\partial t}$, giving the time dependence of electron or hole concentration; the second term gives the time-dependent change in carrier concentration associated with the trap dynamics.

In general, the AC input signal (voltage) and solution variables are expressed as:

$$\xi=\xi_0+\Delta\xi e^{i\omega t} \tag{7.11}$$

where $\xi_0$is the DC solution and $\Delta\xi$ is the complex AC solution under a modulation frequency $\omega=2\pi f$. This form ξ can represent any variable of the system (V, U_n, U_p, or f_{tj}) as a complex solution.

By substituting ξ back into the trap equation and doing a Taylor series expansion to first order at the DC solution, Equation (7.4) becomes:

$$\frac{\partial H_j}{\partial U_n}\Delta U_n + \frac{\partial H_j}{\partial U_p}\Delta U_p + \frac{\partial H_j}{\partial f_{tj}}\Delta f_{tj} = N_{tj} i\omega \Delta f tj \tag{7.12}$$

Solving the above for Δf_{tj} gives:

$$\Delta f_{tj} = \left(\frac{\partial H_j}{\partial U_n}\Delta U_n + \frac{\partial H_j}{\partial U_p}\Delta U_p\right)\left(N_{tj} i\omega - \frac{\partial H_j}{\partial f_{tj}}\right)^{-1} \tag{7.13}$$

$$\Delta f_{tj} = \frac{-\frac{\partial H_j}{\partial U_n}\left(\frac{\partial H_j}{\partial f_{tj}} + i\omega N_{tj}\right)\Delta U_n - \frac{\partial H_j}{\partial U_p}\left(\frac{\partial H_j}{\partial f_{tj}} + i\omega N_{tj}\right)\Delta U_p}{(N_{tj}\omega)^2 + \left(\frac{\partial H_j}{\partial f_{tj}}\right)^2} \tag{7.14}$$

which can be rewritten as:

$$\Delta f_{tj} = F_n(\omega)\Delta U_n + F_p(\omega)\Delta U_p \tag{7.15}$$

By similarly expanding the Poison equation and electron and hole continuity equations as a first order Taylor Series and then substituting the above expression for Δf_{tj}, we obtain the following equations:

$$\frac{\partial F_1}{\partial V}\Delta V + \frac{\partial F_1}{\partial U_n}\Delta U_n + \frac{\partial F_1}{\partial U_p}\Delta U_p + \frac{\partial F_1}{\partial f_{tj}}(F_n\Delta U_n + F_p\Delta U_p) = 0 \tag{7.16}$$

$$\frac{\partial F_2}{\partial V}\Delta V + \frac{\partial F_2}{\partial U_n}\Delta U_n + \frac{\partial F_2}{\partial U_p}\Delta U_p + \frac{\partial F_2}{\partial f_{tj}}(F_n\Delta U_n + F_p\Delta U_p) = \left(\frac{\partial n}{\partial U_n} + N_D\frac{\partial f_D}{\partial U_n}\right) i\omega\Delta U_n \tag{7.17}$$

$$\frac{\partial F_3}{\partial V}\Delta V + \frac{\partial F_3}{\partial U_n}\Delta U_n + \frac{\partial F_3}{\partial U_p}\Delta U_p + \frac{\partial F_3}{\partial f_{tj}}(F_n\Delta U_n + F_p\Delta U_p) = \left(-\frac{\partial p}{\partial U_p} + N_A\frac{\partial f_A}{\partial U_p}\right) i\omega\Delta U_p \tag{7.18}$$

We rewrite these 3 equations in 3×3 matrix form as:

$$(J + D)\xi = 0 \tag{7.19}$$

for each mesh point in the semiconductor, where the J matrix or Jacobian contains the first 3 terms in each equation while D contains the remaining terms concerning the traps and the right-hand side of the equation.

Using this notation with $\xi = (\Delta V, \Delta U_n, \Delta U_p)$ the D matrix at each point is of the form:

$$D = \begin{pmatrix} 0 & \frac{\partial F_1}{\partial f_{tj}} F_n(\omega) & \frac{\partial F_1}{\partial f_{tj}} F_p(\omega) \\ 0 & -(\frac{\partial n}{\partial U_n} + N_D \frac{\partial f_D}{\partial U_n}) i\omega + \frac{\partial F_2}{\partial f_{tj}} F_n(\omega) & \frac{\partial F_2}{\partial f_{tj}} F_p(\omega) \\ 0 & \frac{\partial F_3}{\partial f_{tj}} F_n(\omega) & -(-\frac{\partial p}{\partial U_p} + N_A \frac{\partial f_A}{\partial U_p}) i\omega + \frac{\partial F_3}{\partial f_{tj}} F_p(\omega) \end{pmatrix} \tag{7.20}$$

Consider the boundary points and the internal mesh points separately so that:

$$J \to J + J_B \tag{7.21}$$

and

$$D \to D + D_B \tag{7.22}$$

The matrix equation can then be rewritten in a separated form for internal and boundary mesh points as follows:

$$(J + D)\xi + (J_B + D_B)\xi_B = 0 \tag{7.23}$$

AC Dirichlet boundary conditions imply that ΔU_n and ΔU_p=0 so that the AC voltage drive at the contact has the boundary solution:

$$\xi_B = \begin{pmatrix} \Delta V_B \\ 0 \\ 0 \end{pmatrix} \tag{7.24}$$

where ΔV_B is the applied AC voltage of one volt.

Since the first column in the D matrix must be zero the matrix equation becomes:

$$(J + D)\xi = -J_B \cdot \xi_B \equiv B \tag{7.25}$$

To simplify the numerics, we now wish to write equations involving only real numbers:

$$D = D_r + iD_i \tag{7.26}$$

$$\xi = \xi_r + i\xi_i \tag{7.27}$$

The matrix is thus rewritten as:

$$(J + D_r + iD_i)(\xi_r + i\xi_i) = B \tag{7.28}$$

By expanding the real and imaginary parts, a new expanded matrix can be written:

$$\begin{pmatrix} J + D_r & -D_i \\ D_i & J + D_r \end{pmatrix} \begin{pmatrix} \xi_r \\ \xi_i \end{pmatrix} = \begin{pmatrix} B \\ 0 \end{pmatrix} \tag{7.29}$$

7.2 Numerical techniques in AC analysis

Our task has been reduced to solving the expanded matrix equation detailed in the previous subsection. The AC solution matrix (ξ_r, ξ_i) is easily solved using the sparse matrix inverter to invert the expanded Jacobian and multiply by the matrix on the RHS. However, the inversion can be slow because the expanded Jacobian J+D is twice the order of original Jacobian matrix obtained from the DC solution. We use direct solution techniques to perform the matrix inversion so that convergence is guaranteed for all frequencies.

In our simulator, the AC analysis is implemented as a post-processor, i.e., the analysis is done after the DC solution is obtained from the main solver input file (with .sol extension). This is done in two steps:

- The AC analysis matrix is constructed from Jacobian matrix elements used for DC simulation. To enable export of the additional data from the usual DC analysis, we use the **more_output** statement.
- The AC analysis is activated by the input file with extension .plt. The input file must specify which data set (thus the DC bias condition) is used for the AC analysis. The statement **ac_voltage** is used to apply a 1V AC signal on one of the contacts (or electrodes). The resulting AC current data are stored in an output file defined by the user for later plotting.
- Once the AC current is calculated, a statement such as **plot_ac_curr** is used to plot the results.

The basic outputs are the complex AC currents at all the electrodes and the AC current distribution, in response to the applied AC voltage (one volt) at one of the electrodes. The following important quantities can be extracted from the AC current:

- The conductance matrix elements are computed from the real part of the electrode AC current with one volt applied AC bias at one of the electrodes.
- The capacitance matrix elements are computed from the imaginary part of the electrode AC current, divided by ω, with one volt applied AC bias at one of the electrodes.
- The current gain can be computed from ratio of the AC current magnitudes between two electrodes, with one volt applied AC bias at one of the electrodes.
- The voltage gain is slightly more complicated. It must be done in two steps. For example, we wish to obtain the voltage gain $\frac{\Delta V_2}{\Delta V_1}$. This may be written as:

$$\frac{\Delta V_2}{\Delta V_1} = \frac{\Delta I_2}{\Delta V_1} \times \frac{\Delta V_2}{\Delta I_2} \tag{7.30}$$

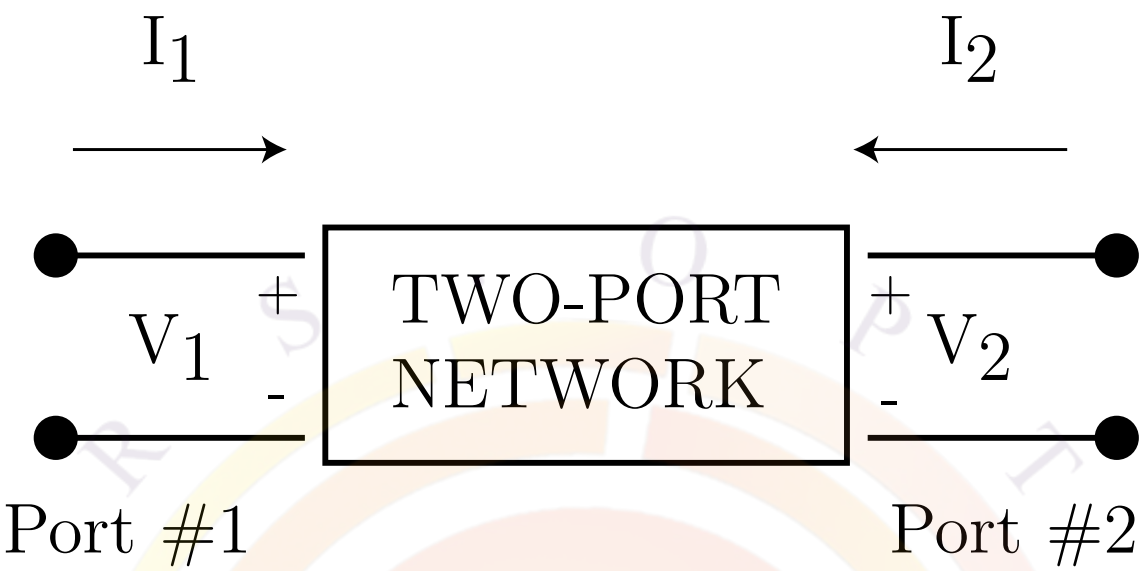


Figure 7.1: Schematics of a general 2-port network. Please note the direction of current flow.

which means we must apply the **ac_voltage** command twice to two different electrodes to obtain the individual ratios.

7.3 Two-Port Network Parameters

Linear networks, or nonlinear networks operating with signals sufficiently small to cause the networks to respond in a linear manner, can be completely characterized by parameters measured at the network terminals (ports) without regard to the contents of the networks. Once the parameters of a network have been determined, its behavior in any external environment can be predicted without regard to the contents of the network. We shall introduce some useful 2-port parameters in this section and explain how they are extracted from DC simulations.

A general 2-port network is indicated in Fig. 7.1. Please note that the positive direction of the current flow is different than in the simulator and all 2-port parameters related to current flow direction will be different by a sign from current flow calculated from the simulator which assigns a positive sign to all current flowing out of a device.

In an AC analysis of semiconductor devices, it is common to share one contact for the input and output. For example, it is often convenient to use the common emitter configuration in a bipolar transistor. By default, all contacts of a device have zero AC voltage bias from a DC simulation. Thus, it is sufficient to specify a single contact as input or output since all other contacts are grounded in AC voltages.

Such a situation is ideal for the computation of the Y-parameters which are defined as follows (referring to Fig. 7.1).

$$\begin{aligned} I_1 &= Y_{11}V_1 + Y_{12}V_2 \\ I_2 &= Y_{21}V_1 + Y_{22}V_2 \end{aligned} \tag{7.31}$$

It is often convenient to express the Y-parameters relative to the system characteristic admittance $Y_c = 1/Z_c$ where Z_c is the system characteristic impedance (often set to be 50 Ohm). We denote the relative admittance using lower case letters here, for example, $y_{11} = Y_{11}/Y_c$.

Based on our discussions in previous sections, the complete set of Y-parameters can be obtained if we excite the device twice, once at the input and once at the output contact. Once the Y-parameters are obtained, the important S-parameters can be computed from the following equations (please refer to any text books on microwave theories):

$$\begin{aligned} s_{11} &= \frac{(1-y_{11})(1+y_{22})+y_{12}y_{21})}{(1+y_{11})(1+y_{22})-y_{12}y_{21}} \\ s_{12} &= \frac{-2y_{12}}{(1+y_{11})(1+y_{22})-y_{12}y_{21}} \\ s_{21} &= \frac{-2y_{21}}{(1+y_{11})(1+y_{22})-y_{12}y_{21}} \\ s_{22} &= \frac{(1+y_{11})(1-y_{22})+y_{12}y_{21}}{(1+y_{11})(1+y_{22})-y_{12}y_{21}} \end{aligned} \tag{7.32}$$

The theories discussed in this section strongly suggest a convenient way of studying the 2-port parameters: if you wish to understand the S-parameters, always go back to the Y-parameters which may be easily understood from a simple AC-voltage excitation of the input or output. Also, a convenient way to check the correctness of 2-port parameters should be to plot the Y-parameters which may be compared with AC-current responses from single-terminal excitations, keeping in mind the sign difference mentioned earlier in this section.

7.4 AC Photo-response

In standard AC analysis, the AC source is voltage applied to one of the electrodes. The source of perturbation appears as the B vector on the right hand side of Equation 7.28 and is non-zero only at boundary nodes where the AC voltage is applied.

When a device is illuminated by a modulated optical signal, similar modulation vector can be set up. The contribution of the optical signal comes from the generation-recombination term which is distributed throughout the whole area sensitive to the

light signal. Therefore, the vector B is non-zero in all mesh points where there is light illumination.

To activate the AC photo-response model, the approach is similar to that of AC voltage but instead of using **ac_voltage**, we use **ac_light**. The output results are shown as AC current on a certain electrode, much in the same way as for an AC voltage bias.

7.5 Laser Diode Modulation Response

Laser diode modulation response is a special case of AC analysis with additional variable of photon density. Similar to the AC photo-response described in the previous section, the same photon-carrier interaction is through the same generation/stimulated recombination term: $R_{st} = v_g g_m S(x, y, z)$ where v_g is the group velocity of light, g_m is the modal optical gain and S is the bulk photon density. In the case of photo-sensitive devices, the optical gain becomes negative and carriers are generated instead of annihilated. An additional photon rate equation must be solved together with all other equations of the drift-diffusion system.

To activate the modulation model, AC small voltage excitation (through **ac_voltage**) is used. The corresponding AC current is computed and used in the plotting the laser diode modulation response.

Chapter 8

QUANTUM WELLS AND INTERBAND TRANSITIONS

This chapter presents the quantum well models and related interband optical transitions in Crosslight simulation software. Different levels of approximations for quantum well models are described along with the corresponding optical gain/absorption calculations. The theory of many-body gain enhancement implemented in the software is also explained near the end of the chapter.

8.1 Quantum Well Models

8.1.1 Introduction

Quantum well models are among the most sophisticated models used in Crosslight Software and there are different levels of approximation available. They are summarized in Fig. 8.1 and detailed below:

- 1. The simple (also the default) quantum model assumes a a single, symmetric, flat-band and step-wise potential profile. Quantum wells in a MQW region are assumed to be isolated from each other and the wave functions do not overlap.
- 2. The complex MQW removes the restriction of symmetry and allows us to treat asymmetric wells (2a) or coupling between wells (2b).
- 3. The self-consistent MQW model removes the restriction of flat-band and couples the potential with charge density in a self-consistent manner. This can be used in isolated quantum wells (3a) or in a coupled complex MQW (3b).

Note that an extra layer of complexity/accuracy can be introduced by considering valence mixing models when solving the Hamiltonian.

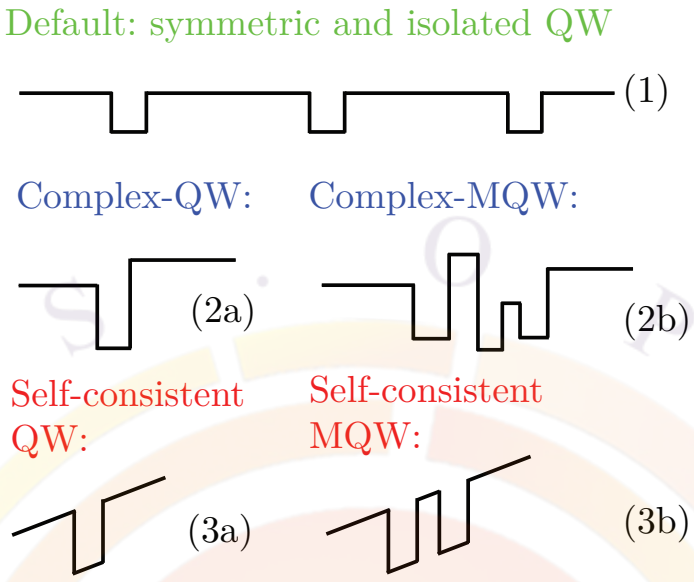


Figure 8.1: Different levels of approximation for quantum wells models in Crosslight

8.1.2 Simple quantum wells

When a confining potential well gets narrow, it is necessary to treat it as a quantum well. We consider the case of a simple quantum well without external applied field. All the quantum levels for the sub-bands of Γ, L, light holes and heavy holes (and crystal field holes for wurtzite) are computed from well known formulas in quantum mechanics for a square quantum well [14].

As the external field is applied, we allow the quasi-Fermi level to vary as a function of distance. The density of states and quantum levels are assumed to be the same as if there were no applied field.

The quantum well levels are calculated at every bias point during an actual simulation because the bandgap of the active region is a function of carrier density. As the bias increases, the higher carrier density in the quantum well reduces the bandgap and changes the quantum well depth. Therefore the quantum levels must be recomputed at every bias point.

Note that by default, the heavy and light holes are assumed to be parabolic and decoupled. A simplification has been made so that, by default, all the in-plane effective masses of heavy hole or light hole subbands are assumed to be same. If the user wishes to alter the effective mass of a particular subband, the software allows him/her to do so with the **modify_qw** statement.

8.1.3 Anisotropic parabolic approximation

Inclusion of biaxial strain in a quantum-well has become a common practice for many heterojunction devices. Strain offers an additional degree of freedom and can produce some desirable effects such as lower threshold current for a laser diode.

Strain is known to cause the valence band of a III-V semiconductor to split into separate light hole (LH) and heavy hole (HH) bands (or CH for wurtzite structure) which are strongly non-parabolic. A rigorous treatment of strain effects on the band structure is rather complicated [15] and the gain function involves an integration over k-space.

We may approximate the non-parabolic band structure by an anisotropic parabolic one. One can show that with proper choices of parameters, a good approximation to the non-parabolic band structure can be obtained. This approximation greatly simplifies the calculation of gain, spontaneous emission and carrier concentration and allows the incorporation of strain effects into the program.

For zincblende structures, a simplified analytical band structure of a strained quantum well has been developed [15] using an efficient decoupling method to transform the original 4×4 valence band Hamiltonian into two blocks of 2×2 upper and lower Hamiltonians. As a result of the decoupling, an analytical expression can be derived [15]. To be consistent with our convention the y-axis (equivalent to the z-axis in [15]) is defined to be perpendicular to the quantum-well plane and the z-axis to be along the direction of light propagation.

We use the following expression for the bulk valence band energy [15]:

$$-E = \frac{\hbar^2\gamma_1}{2m_0}(k_x^2 + k_y^2) \pm \left[\left(\frac{\hbar^2\gamma_2}{2m_0}(k_x^2 - 2k_y^2) + \zeta\right)^2 + 3\left(\frac{\hbar^2\gamma_2}{2m_0}k_x^2\right)^2 + 12\left(\frac{\hbar^2\gamma_3}{2m_0}\right)^2 k_x^2 k_y^2\right]^{1/2} \tag{8.1}$$

$$\varepsilon = \frac{a_0 - a(x)}{a_0} \tag{8.2}$$

$$\delta E_{sh} = b\left(1 + 2\frac{c_{12}}{c_{11}}\right)\varepsilon \tag{8.3}$$

$$\zeta = \frac{1}{2}\delta E_{sh} \tag{8.4}$$

where the valence band is given in the $k_x - k_y$ plane. The information on strain is contained in ζ; compressive strain is represented by a negative ζ.

The valence band mixing effect for bulk material has been taken into account since the coupling between heavy and light holes has been included in the above equations.

Note that the in-plane direction for Equation (8.1) is the [100] crystal direction, and is different from the [110] direction. For many applications, it is desirable to average the energy dispersion in both the [100] and [110] directions. One can show that the average can be approximated by replacing γ_2 in the second term under the square root of Equation (8.1) by $(\gamma_2 + \gamma_3)/2$: this is called the axial approximation. The simulator provides an option to turn on this approximation.

To convert the non-parabolic bulk band structure into a suitable form, the software fits the valence band to the following anisotropic parabolic band expression, over a range large enough to cover all the optical transitions in k-space:

$$E = -\frac{\hbar^2}{2m_{vx}m_0}k_x^2 - \frac{\hbar^2}{2m_{vy}m_0}k_y^2 \ . \tag{8.5}$$

The quality of the fit is found to be reasonable for practical cases. Such an approximation is appropriate not only because of the reasonable quality of the fit, but also because the approximation has not lost the basic effects of strain, *i.e.*, it causes the symmetry in the band structure to be broken.

The splitting of the heavy and light hole bands is described as [15]:

$$\delta E_{hy} = 2a(1 - c_{12}/c_{11})\varepsilon, \tag{8.6}$$

$$E_g^{hh} = E_g^u + \delta E_{hy} - \delta E_{sh}, \tag{8.7}$$

$$E_g^{lh} = E_g^u + \delta E_{hy} + \frac{1}{2}\left[\delta E_{sh} + \Delta - \sqrt{\Delta^2 + 9\delta E_{sh}^2 - 2\delta E_{sh}\Delta}\right]. \tag{8.8}$$

Once the band structure is simplified to an anisotropic parabolic form, the program uses a one-band model to solve for the subbands of the quantum well from the effective masses perpendicular to the quantum well plane. By default, the mixing between different subbands is ignored and all the subbands of heavy holes (or light holes) are assumed to have the same effective in-plane mass. The resulting subbands from such a theory agree qualitatively with k.p theories with multiband full mixing effects, especially for subbands near the band edges and near the Γ center. Since the most important subbands are those near the band edges, the subband model here is expected to be reasonable. This approach of a one-band model with parabolic subbands has been used extensively for many years and is included in many frequently cited papers (see for example, [16]-[17]).

Once the parabolic subbands are found, one can apply conventional approaches to treat the carrier concentration and the optical transition probabilities. Specifically, the effective mass perpendicular to the plane (m_{vy}) determines the quantum subband levels (or quantum confinement effects) and the optical transition energies. The (2D) density of states and joint density of states of each subband depends on the effective mass in the plane (m_{vx}).

For those users who wish to adjust in-plane effective masses for different subbands, so that the theory here agrees with other k.p models, the program provides an option to do so. That is, the user can shift a subband level or change a subband in-plane mass to match any other theory.

8.1.4 Carrier density in anisotropic parabolic approximation

We use the effective masses parallel to the plane of the quantum well for the evaluation of carrier concentrations.

$$n = \sum_j \rho_j^{x0} kTln\left[1+e^{(E_{fn}-E_j)/kT}\right] + unconfined\ electrons, \tag{8.9}$$

$$p = \sum_i \rho_i^{x0} kTln\left[1+e^{(E_i-E_{fp})/kT}\right] + unconfined\ holes, \tag{8.10}$$

where the subscript i denotes all confined states for the heavy and light holes, and j designates those for the Γ and L bands. The unconfined carriers are calculated using Fermi statistics as described in Section 5.1.2. Note that the effective mass perpendicular to the plane, (m_{vy}), affects the quantum levels and therefore the distribution of carriers in the quantum well.

8.1.5 Valence mixing

Early theories of bulk and quantum well semiconductor lasers assumed parabolic subbands for both the conduction and valence bands. Many early classic papers on quantum well semiconductor lasers used the concept of effective mass, which is equivalent to the parabolic band approximation. A partial list of publications includes Refs.[16, 17].

The reason for the use of parabolic band, regardless of its inaccuracy, is very simple: the computation of the energy subbands is not the end of the story and one has to use the subbands to calculate other more interesting physical quantities, such as the optical gain. Calculation of the optical gain for a non-parabolic subband is not easy and one is much better off staying with a parabolic band approximation.

More recent theories of quantum well subbands are mostly based on k.p theory with valence band mixing effects. These usually involve solving a 4×4 [15], 6×6 [18] or 8×8 [19, 20] Hamiltonian of the Luttinger-Kohn type and imposing an envelope function approximation in solving the quantum well subband structures (e.g. [21]). The valence subband structures obtained from this approach are complicated and heavily mixed in many cases. For some cases, the effective masses of some of the

hole subbands are negative at the Γ point in k-space, making it impossible to use analytical approaches.

Generation of the subbands using a valence mixing model is not difficult and has been done previously by many authors [15, 18–20, 22]. In contrast, the computation of quasi-Fermi level and optical transition probabilities is not easy because the subbands are non-parabolic and the wave functions can be heavily mixed.

The detailed formulation of k.p theory for quantum well subband structure calculations has been described in many publications. We have closely followed the formulas of S.L Chuang [15]. Details will not be repeated here; rather, an outline of physical models is presented to give an overview of the theory.

In the k.p theory for a bulk crystal, the basic problem is the following: given the energy level and wave function at a symmetry point of the k-space (e.g. the Γ point), how does one calculate the band structure near such a symmetry point?

Based on the symmetry properties of the crystal, a 4×4 or 6×6 Luttinger-Kohn Hamiltonian for the Γ valence band can be constructed, depending on how many valence bands are involved. As a solution of the bulk Luttinger-Kohn Hamiltonian, the energy dispersion is expressed in terms of Luttinger numbers and quadratic equations in k [23] [24]. Note that the energy is accurate to the k^2 term and no further. The wave function at a general k-point is expressed as a linear combination of the wave function at the symmetry point.

If the solid crystal is under perturbation, one may assume that the perturbed wave function can still be expressed as a linear combination of the wave functions at the symmetry point. Within the envelope function approximation, the wave function of the perturbed crystal (due to the potential of a quantum well or of an impurity) can be written as (see Eq. (IV.12) in [23]):

$$\Psi = \sum_{j=1} F(\mathbf{r})\phi_j(\mathbf{r}), \tag{8.11}$$

where ϕ is the periodic part of the Bloch function obtained at the Γ point in the absence of any perturbation. In the work of Luttinger and Kohn [23], the perturbing impurity potential was assumed to be "gentle" and involving only tens of meV. Today the application of the envelope function approximation has been very much broadened.

For a quantum well, the envelope function $F(\mathbf{r})$ in the linear combination is determined by two boundary conditions: the continuity of the wave function and the continuity of the current density. As a result of the solution of the envelope function, the subband energy dispersion and the wave function can be obtained. Due to the non-vanishing off-diagonal terms in the Luttinger-Kohn Hamiltonian, the wave function is a mixture of heavy hole and light hole functions (hence the name valence mixing).

In general, the valence subband structure obtained from such a model has the following features:

- 1. The energy and wave function at the Γ point are the same as those derived from the decoupled methods discussed in previous sections. This makes it possible to label each subband as heavy hole and light hole based on the base function there. For this reason, many concepts developed in the effective mass approximation can still be used. For example, the selection rule and matrix element for optical transitions at or near the Γ point can still be used.
- 2. The key difference of this model is the anti-crossing behavior of the subbands in the valence mixing model, *i.e.*, the subband of the light hole can not cross that of the heavy hole. This causes some major discrepancies for structures where light holes and heavy holes are close. The software has captured this major feature of anti-crossing by adjusting the in-plane effective masses of the subbands so that the subbands do not cross each other.
- 3. Some of the subbands have negative effective masses at the Γ point. In the effective mass theory for optical gain calculation, negative masses are not allowed. Therefore, at first glance, there is nothing one can do about this in the effective mass theory. If one takes another look at the subband structure over a larger k-range, the effective mass eventually returns to a positive value. If a large enough k fitting range is considered, one can still approximate the funny-looking subband by a parabolic band with positive effective mass. The approximation is not so bad considering the density of states is to be broadened by a scattering lifetime, and the exact location of the energy level is meaningless in a practical calculation anyway.

In summary, Crosslight simulator has two levels of models regarding strains and valence mixing effects. By default, the simulator fits the non-parabolic bulk valence band dispersion with two parabolic curves in each direction to obtain anisotropic effective masses. Efficient computation of carrier density and interband optical transition may be achieved this way. By setting **valence_mixing**=*yes* in the **active_reg** or **set_active_reg** statements, a full computation of subbands using k.p theory is performed. Carrier densities and interband optical transitions are obtained using numerical integral over the non-parabolic subbands, resulting in longer computation time.

8.1.6 Complex MQW active regions

In previous subsections, it has been implied that quantum wells in an MQW system do not couple with each other. This is a reasonable approximation if the wells are far apart and the wave function decays significantly in the barriers. There are

circumstances in which we need to consider the effect of coupling between quantum wells.

Initially, it appeared that extension of a non-coupled model to a coupled one should be straightforward: just use the potential for two wells instead of one. However in an actually simulation, we must solve problem of localizing the carriers in a complex coupled MQW structure.

Consider the simple case of two wells. As the two wells are brought closer together, the degenerate subbands start to split. From the view point of wave mechanics, the wave function of each energy level belong to both wells. However, drift-diffusion theory (classical theory) requires that we must know where the carriers are located.

Therefore, we face the task of deciding which well will be assigned the carriers from each coupled confined state. Similarly, for optical interband transition, the process takes place in the whole coupled complex structure. Again, we must decide in which well a given transition takes place. The situation is like quantum tunneling where we have a contradiction: the wave nature in a quantum theory versus the particle nature in drift-diffusion theory. We must create a reasonable method to bridge the gap.

For carrier density calculation, we localize the coupled confined states as follows. Suppose the probability of finding an electron of a confined state in well 1 is p_1 and that of well 2 is p_2. In general, $p_1 + p_2 < 1$ and the remaining probability is spent in the barriers. We consider that the electron is localized in well 1 with a probability of $p_1/(p_1 + p_2)$. Similar consideration is given for holes.

For optical transition from the valence band to the conduction band, we regard the optical process take place in well 1 with a chance of $p_1 q_1/(p_1 q_1 + p_2 q_2)$, where q_1 and q_2 are probabilities of finding the hole in well 1 and 2, respectively.

Note that as we relax the approximations of the simple QW model, it may also be desirable to have quantum well designs with non-symmetric barriers. The same system is used as with the coupled well situations: complex MQW macros. These special macros must be used to define a complex structure and use "type=strained_complex" in the **layer_type** statement. By convention, complex macros have names start with "cx-" in the material database.

Regions where the complex active region is defined must be labeled by the statements **begin_complex** and **end_complex**. This will be done automatically by layer.exe if a complex macro is used. Users should remember that since the whole MQW structure needs to be solved at once, barrier layers must also invoke the complex MQW macros.

Some restrictions are present in the complex MQW macros. The first is that an odd number of layers must be used: this is normally not a problem since most designs use a barrier/well/..../barrier layout. If need be, split one of the outer barriers into two to get an odd number of layers.

Related to this assumption, the software will only calculate the gain in even-numbered layers since they are presumed to be wells. For complicated structures such as wells with graded compositions or Type II band alignment (where the electrons and holes are localized in different layers), this can cause a problem. Use the **inner_bar_gain** statement to force gain calculations in all the layers of the complex MQW. Also note that outer barriers are always excluded from the gain calculations as they only serve as a boundary layers where the wave can decay before reaching zero: see Fig. 8.4 below.

As alternative to the above, complex quantum well regions can also be declared using the "library" method of Sec. 3.5. This automatically declares all tagged layers as being part of the same quantum-coupled region and does not rely on the barrier/well/barrier approximation for the inner barriers. Outer barriers are still excluded from the gain calculations though.

8.2 Self-consistent carrier density model

8.2.1 Localized confined states

As we discussed previously, the default approach in our simulator is to solve the quantum confined states in flat-band condition (assuming no electric field). When there is a local variation of potential, we assume it is small enough that flat-band solution is still valid but only introduces a local correction to the confined energy level (see Fig. 8.2).

For simplicity of numerical computation, we also assume that carrier density is only confined within the well and is computed according to the local Fermi level and local confined energy level. The following formula for confined 2D carrier electron density is used:

$$\begin{aligned} n_{2D}(x,y) &= \frac{1}{d_w}\sum_j \rho_j^0 kTln\left[1+exp[(E_{fn}(x,y)-E_j(x,y))/kT]\right], \quad inside\ well \\ &= 0, \quad outside\ well \end{aligned} \tag{8.12}$$

where d_w is the well thickness. The subscript j denotes all confined states.Please note that both the Fermi level and confined level are assumed to be spatially dependent. ρ_j^0 is the 2D density of states.

The variation of the confined level $E_j(x,y)$ follows that of potential, much in the same way the conduction band edge follows the potential change (see Fig. 8.2). Please note that the spatial variation in this model is uniform except for variations introduced by variation in Fermi level and confined level. We shall refer to this model as "uniform profile" density model.

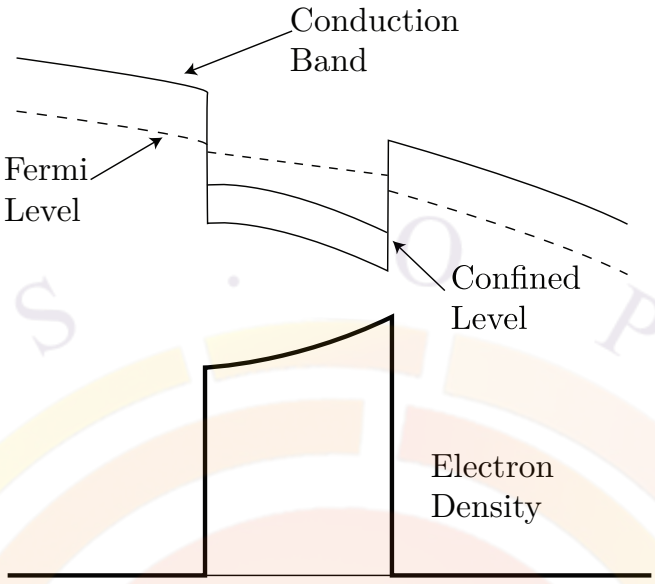


Figure 8.2: Default model of quantum well carrier density calculation.

The above model is simple to implement and efficient to compute. It gives the right MQW behavior if the realistic band structure is close to flat-band condition and we do not care about the details of the carrier density within the scale of the quantum well. Such is the case of most MQW lasers under lasing condition when the band structure is close to flat-band under forward bias and high injection condition.

However, we may run into trouble if the details of carrier density indeed matter when the well is heavily tilted to one-side or another due to strong electrical field such as piezoelectric field. We will consider this case in the next subsection.

The above discussion is limited to electron density but discussion for hole density is completely similar.

8.2.2 Self-consistent carrier density distribution

When the potential well is under strong electric field such as in the case of piezoelectric field, the well will be tilted to one side. The confined wave function will also be tilted to one side (see Fig. 8.3). Furthermore, the optical transition between conduction and valence band will have a different dipole moment than under flat-band condition. The tilting of the well will have strong consequences to current overflow which may affect the device performance.

As an option in our software, the self-consistent electron density is given as

$$n_{2D}(x,y) = \sum_j g_n^j(y)\rho_j^0 kTln\left[1 + exp[(E_{fn}(x,y) - E_j)/kT]\right] \tag{8.13}$$

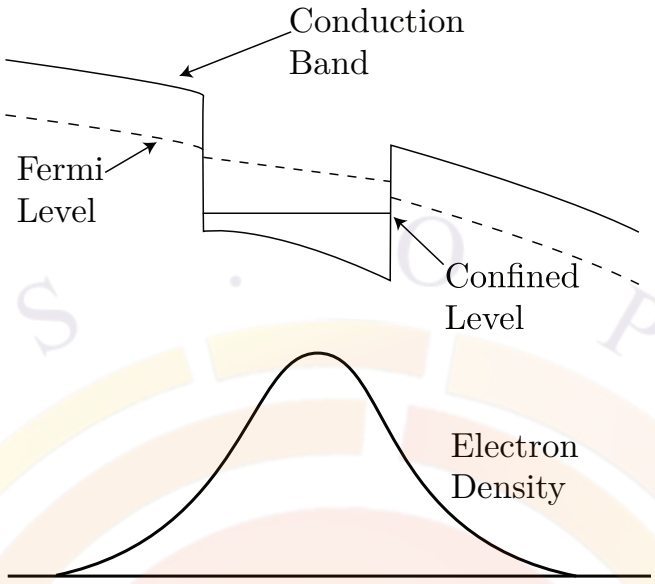


Figure 8.3: Schematics of the self-consistent carrier density model for a single quantum well.

where $g_n^j(y)$ is the electron wave function assuming the well is parallel to x-axis. The confined level E_j is no longer a function of position. Computation of the above density distribution requires more work because the density at one point depends on a global function $g_n^j(y)$. We also have to update the wave function and confined energy levels at different biases.

Similar expressions can be obtained for the hole subbands.

To achieve complete self-consistency, the following procedures are used:

- 1) At equilibrium, solve the potential using flat-band approximation. The default uniform profile model is used for the density. This gives us the initial potential distribution.

- 2) At equilibrium, solve the potential again with potential obtained in 1) above. The self-consistent density model is used. A new density profile is obtained in this step.

- 3) Iterate 2) above until numerical self-consistency is achieved for density profile and potential distribution.

- 4) Once we achieve self-consistency in equilibrium after a number of numerical iterations, we increase the bias and repeat 2) and 3) above.

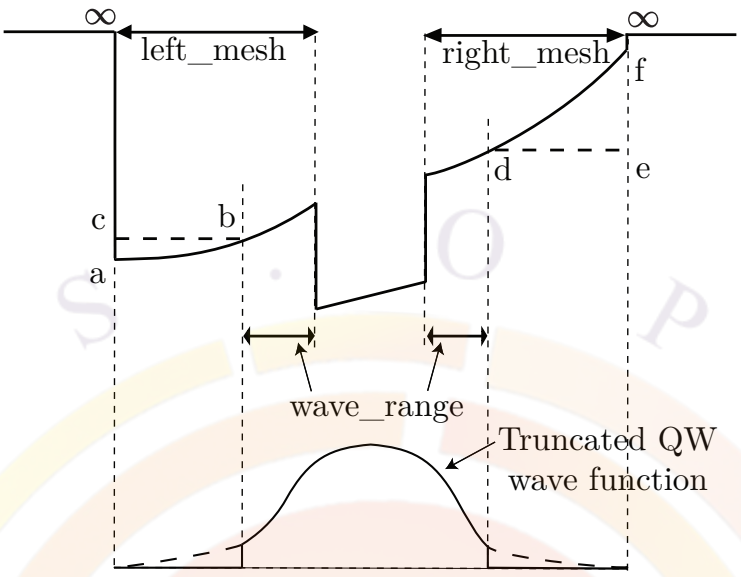


Figure 8.4: Schematic diagram explaining the self-consistent wave boundary conditions

8.2.3 Wave function boundary conditions

When solving the Schrödinger equation under self-consistent conditions, the most important parameters are the boundary conditions on the left and right side of the QW or MQW region. We will use the example of a single quantum well in Fig. 8.4 to explain in detail.

Analytical solutions to the Schrödinger equation state that the wave functions decays to zero at infinity. In a numerical solution, we must use a finite value instead: one may assume an infinite potential wall a certain distance away from the QW region. In order not to perturb the position of the energy levels or the shape of the wave function, a large enough value must be used. However, larger values can cause numerical accuracy issues so by default, a distance equal to three times the well width is used. This may be overridden by the **left_mesh** and **right_mesh** parameters of **modify_qw**.

In addition, we place some restrictions to the shape of the confining potential. The potential profile used in the solution must not extend too far since stable confined quantum states may not exist when one side of the potential is lowered too much by the applied field. Therefore, we choose a reasonable distance (**wave_range** in the **self_consistent** statement) to truncate the potential profile (at points b and d) on Fig. 8.4 and artificially extend the potential (to points c and e of the figure instead of the original a and f). Note that the value of **wave_range** may be cut short if a material interface is detected.

The wave function outside the region specified by **wave_range** is ignored and trun-

cated to zero. This does not affect the shape of the wave function or the position of the energy levels like the position of **left_mesh** and **right_mesh**. It only forces the wave to be normalized to the [b,d] range. This may cause a bit of inaccuracy in the carrier distribution if the wave has not decayed sufficiently but this model prevents perturbations near the artificial boundary wall from affecting the results of the simulation. We also note that any states with energies above the confining potential at both ends of this range are considered unconfined states.

After this adjustment of the potential profile, reasonable confined states can be obtained in most cases. If more confined states are required than those provided by this fix, the parameters **more_cond_energy** and **more_val_energy** of the **modify_qw** statement may be used to increase the search range for confined levels.

8.2.4 MQW regions

By default, the software will treat multiple wells with same composition as the same active region in order to save on computation time. When a self-consistent calculation is used, this approximation is no longer valid since even identical wells can be subjected to different local potentials.

In order to force the software to treat individual wells as different active regions, use the **independent_mqw** statement in the .layer file. This will assign different material numbers to every quantum well. Users should expect a slower simulation with this setting since more computational work is required.

8.2.5 Cases of valence mixing and complex MQW

In the case of valence mixing model, the valence subbands are no longer parabolic and we must abandon the formulas given in the previous subsections. The confined hole density can be written as an integral over the non-parabolic subbands $E_j(k)$:

$$p = \sum_j g_p^j(y) \frac{1}{2\pi} \int_{-\infty}^{E_k(0)} [1 - f_v(E_j(k), E_{fp})] dk^2 \tag{8.14}$$

where f_v is the Fermi function for the valence band.

For complex MQW with coupled wells, the same expression can be used except that we need to use a proper density partition scheme to make sure that the correct portion of carrier density from a quantum level be allocated to each well.

An additional complication arises for thick complex MQW regions under a strong local potential. The entire MQW region will be tilted but, as mentioned above, the confined energy levels are no longer a function of position in a self-consistent simulation. This means that sometimes, the solver will find energy levels only at the

outer edges of the complex MQW region: the wave functions for these levels may have very poor overlap.

8.2.6 Quantum wells in quantum-MOS and HEMT

When the gate of an MOS structure is under bias, the interface of silicon and SiO2 forms a triangular-well for electrons or holes, depending on the polarity of the bias. A similar situation arises in high mobility electron field effect transistors (HEMT) where the internal electric field causes the heterojunction to form a triangular well to confine the 2D electron gas system.

A triangular well formed in such a way may be regarded as a special case of a quantum well discussed above. To derive such a structure from a simple symmetric flat-band quantum well we take the following steps:

- 1. The left barrier is lowered to the point of zero barrier height
- 2. External and internal electric field is allowed to tilt the potential in such a way that a triangular well is formed between the bottom of the well and right barrier.

Thus, a realistic quantum well model for quantum-MOS and HEMT may be achieved if we apply the complex-MQW and self-consistent models to a non-symmetric quantum well. Since our models are well established for quantum wells with two barriers, we prefer to treat triangular wells as if they have a left-barrier of zero height.

In the case of quantum-MOS, some special treatment is required since the conduction band should be treated with anisotropic electron masses which may affect the conduction band quantum states, depending on the crystal orientation with respect to the SiO2.

8.3 Interband Optical Transition

8.3.1 Introduction

Many direct-bandgap devices handled by Crosslight simulation software interact with light in one way or another. For indirect bandgap devices such as silicon MOS, this section may be skipped.

For active devices such as laser diodes and LED's, the interband optical gain and spontaneous emission spectrum are important. For photo-sensitive devices such as photo-detectors, the interband absorption (opposite of gain) is important.

In this section, we start with the basic theory of optical gain based on a parabolic band structure. The parabolic band model is the default model and is also our starting point for more complex optical gain models. The parabolic band restriction may be removed if a numerical integral is used to replace the effective masses.

Note that the formulas for quantum wells we consider here are rather general and are also applicable to bulk material. In the case of bulk material, only one subband need to be considered with three-dimensional reduced density of states.

8.3.2 Interband transition model

Similar to the derivation of the absorption coefficients for a solid, the local gain due to a transition from a conduction band labeled j to a valence band labeled i can be written as [25],

$$\begin{aligned} g_{ij}(E^0_{ij}) &= \int P_{ij}\left(\frac{\epsilon_1}{\overline{n}c}\right)\rho_{ij}dE_{ij} \\ &= \int \left(\frac{2\pi}{\hbar}\right) \mid H_{ij} \mid^2 (f_j - f_i)\delta(E_{ij} - \hbar\omega)\left(\frac{\epsilon_1}{\overline{n}c}\right)\rho_{ij}dE_{ij} \\ &= \left(\frac{2\pi}{\hbar}\right) \mid H_{ij} \mid^2 (f'_j - f'_i)\left(\frac{\epsilon_1}{\overline{n}c}\right)\rho_{ij} \end{aligned} \tag{8.15}$$

$$\rho_{ij} = \rho^0_{ij} h(\hbar\omega - E^0_{ij}), \tag{8.16}$$

$$\mid H_{ij} \mid^2 = \left(\frac{q}{m_0}\right)^2 \left(\frac{2\hbar\omega}{4\epsilon_1\epsilon_0\omega^2}\right) M^2_{ij}. \tag{8.17}$$

where $\overline{n}$ is the real part of the refractive index. f_i and f_j are the Fermi functions for the ith and the jth levels, respectively, and f'_i and f'_j are given by

$$f'_i = \left\{1 + exp\left[(E^0_i - \frac{m_{ij}}{m_i}(E - E^0_{ij}) - E_{fp})/kT\right]\right\}^{-1}, \tag{8.18}$$

$$f'_j = \left\{1 + exp\left[(E^0_j + \frac{m_{ij}}{m_j}(E - E^0_{ij}) - E_{fn})/kT\right]\right\}^{-1}. \tag{8.19}$$

Note that the above formulas for the gain are rather general and are applicable to quantum wells as well as to bulk material. In the case of bulk material, only one pair of energy bands need be considered. Three-dimensional reduced density of states and bulk momentum matrix elements are to be used. In the case of quantum wells, many subbands in both the valence band and the conduction band need to be considered. 2D density of states and quantum well momentum matrix elements are to be used. Therefore one can consider the expression for the bulk material as a special case of that for quantum wells. The following discussion assumes the formulas for quantum wells.

The gain function is the sum of g_{ij} over all the subbands for the allowed transitions. The software uses the following expression for the density of electrons and holes in the quantum well:

$$n = \sum_j \rho_j^0 kTln\left[1 + e^{(E_{fn} - E_j)/kT}\right] + unconfined\ electrons \tag{8.20}$$

$$p = \sum_i \rho_i^0 kTln\left[1 + e^{(E_i - E_{fp})/kT}\right] + unconfined\ holes \tag{8.21}$$

where the subscript i denotes all confined states for the heavy and light holes, and j designates those for the Γ and L bands. The number of unconfined carriers are calculated using Fermi statistics as described in Section 5.1.2.

At the current stage of model development the software assumes that the electron states outside the quantum wells can be treated as unbound states so that three dimensional Fermi statistics can be applied to these regions for the carrier density. This assumption is reasonable in most applications, and greatly simplifies the simulation since the potential profile is generally a strong function of bias voltages.

All the quantum levels for the sub-bands of Γ, L, light holes and heavy holes are computed from well known formulas in quantum mechanics for a square quantum well [14]. The quantum well levels are calculated at every bias point during an actual simulation because the bandgap of the active region is a function of carrier density. As the bias increases, the higher carrier density in the quantum well reduces the bandgap and changes the quantum well depth. Therefore the quantum levels must be re-computed at every bias point in a self-consistent simulation.

Note that by default, the heavy and light holes are assumed to be decoupled and can be treated separately. Simplification has been made so that, by default, all the in-plane effective masses of heavy hole or light hole subbands are assumed to be same. If the user wishes to alter the effective mass of a particular subband, the simulator allows him/her to do so with the “modify_qw" statement. Further discussion on the effect of valence band mixing is given in Sections 8.1.3 and 8.1.5.

The simulator uses TE (transverse electromagnetic) mode as the default. The user should set the polarization mode to TM (transverse magnetic) mode if a large tensile strain is present in the material. We use the following anisotropic dipole moment for

the heavy and light hole transitions in a quantum well [26]:

$$A_{hh} = \frac{3+3cos^2(\theta_e)}{4} \quad (TE), \tag{8.22}$$

$$A_{lh} = \frac{5-3cos^2(\theta_e)}{4} \quad (TE), \tag{8.23}$$

$$A_{hh} = \frac{3-3cos^2(\theta_e)}{2} \quad (TM), \tag{8.24}$$

$$A_{lh} = \frac{1+3cos^2(\theta_e)}{2} \quad (TM), \tag{8.25}$$

$$M_{hh} = A_{hh} O_{ij} M_0 \ , \tag{8.26}$$

$$M_{lh} = A_{lh} O_{ij} M_0 \ , \tag{8.27}$$

where A_{hh} and A_{lh} are the quantum well dipole moment enhancement factors. M_0 is the dipole moment of the bulk material given by the following expression:

$$M_0 = \frac{1}{6}\frac{m_0}{m_e}\frac{E_{g0}(E_{g0}+\Delta)}{E_{g0}+2\Delta/3}. \tag{8.28}$$

$cos(\theta_e)$ is defined as

$$cos(\theta_e) = E_{ej}/[E_{ej} + m_r/m_e(E - E_{ij})] \quad forE > E^0_{ij}, \tag{8.29}$$

$$cos(\theta_e) = 1 \quad forE \leq E^0_{ij}. \tag{8.30}$$

where E_{ej} is the electron confined subband energy of level j and m_r is the reduced mass.

A few interesting points about dipole enhancement are discussed here. The above equations show that the heavy hole transition is favored under TE polarization while the light hole is favored for TM polarization. With a particular polarization (TE or TM) the bulk moment is split between the heavy and light holes and they average out to the bulk moment. It is interesting to note that the light hole under TM has a larger dipole moment than the heavy hole under TE. Therefore, from the point of view of the larger dipole moment, the light hole transition is very attractive. A major drawback for light hole transition is that, due to light mass, the quantum mechanic wave function tends to spread out more than for the heavy hole. As a result, the overlap integral (O_{ij} above) is reduced.

8.3.3 Lorentzian model of gain broadening

Broadening of intra-band scattering significantly reduces the local gain function and must be considered. To describe the broadening of the quantum levels, the software

uses a line shape function $L()$ in a convolution integral with the optical gain. The final expression for the gain function in the quantum well then becomes

$$g_{qw} = \sum_{i=j} \int g_{ij}(E_x)\tau/\hbar L \left[(E_x - E_{ij}^0)/(\hbar/\tau)\right] dE_x. \tag{8.31}$$

The simplest line shape function is the Lorentzian shape function:

$$L(E_x - E_{ij}^0) = \frac{1}{\pi} \frac{\Gamma_0}{(E_x - E_{ij}^0)^2 + \Gamma_0^2}, \tag{8.32}$$

where Γ_0 is the constant half width of the shape function.

The expression in Equation (8.31) cannot be integrated analytically and an efficient numerical approach for the evaluation of the integral is necessary to perform the 2-D simulation. The program has approximated the Fermi functions f_i and f_j with linear combinations of four exponential terms, which gives an error of less than 0.04% . The Lorentzian shape function is approximated with a linear combination of three exponential terms to give an error of less than 1.2% for a range of -8 to +8 for a standard Lorentzian function.

Equation (8.31) is used to evaluate the stimulated emission and dielectric constants in the basic equations at all the points of the 2D mesh.

The corresponding local loss can be written as

$$\alpha_{qw} = -\alpha_{fn} n - \alpha_{fp} p - \alpha_{act}, \tag{8.33}$$

where the first two terms are due to free carrier absorption in the quantum well. α_{act} is an adjustable background loss term to account for losses in the active quantum well for mechanisms other than interband transitions and free carrier absorption.

When a laser is biased near threshold, the current is mainly determined by the spontaneous emission rate and/or Auger recombination. Therefore the evaluation of the spontaneous emission rate is important. The following expression for the spontaneous emission rate is used [27]:

$$r_{sp}^{qw}(E) = \sum_{i=j} \left(\frac{2\pi}{\hbar}\right) \mid H_{ij} \mid^2 f_j'(1 - f_i') D(E) \rho_{ij}, \tag{8.34}$$

where D(E) is the optical mode density in the material which has a refractive index of $\overline{n}$, given by [27],

$$D(E) = \frac{\overline{n}^3 E^2}{\pi^2 \hbar^3 c^3}. \tag{8.35}$$

The software has used the same broadening line shape functions for the spontaneous emission spectrum in Equation (8.34) as for the gain function. This allows the user to compare his/her experimental data with the broadened spontaneous spectrum.

The broadened spontaneous emission spectrum is, however, not used in the main simulation. The reason is that since carrier recombination involves emission at all frequencies, it is only necessary to integrate the un-broadened spectrum in Equation (8.34) over all possible frequencies:

$$R_{sp}^{qw} = \int_0^\infty r_{sp}^{qw}(E)dE. \tag{8.36}$$

Note that Equations (8.31) and (8.36) are given in terms of the quasi-Fermi levels which can be treated directly as variables for the Newton's method used by the software.

The choice of τ is important since it reduces the peak gain and directly determines the threshold current. Also note that the lasing frequency is determined by the maximum of the modal gain g_m at the present injection level rather than the *local* gain g, since it is the modal gain that appears in the rate equation. The former is an average gain over the optical field. For simplicity the software uses the wavelength at the peak mode gain as the single wavelength appearing in the wave equation.

The effective bandgap for optical gain calculations is reduced because of exchange effects [16]. Such a bandgap shrinkage is given by: $\Delta E_g = A_x \left(\frac{n+p}{2}\right)^{1/3}$. Since this term increases with carrier concentration, it causes a "red shift" tendency in the optical gain spectrum as the carrier concentration is increased. The peak optical gain also has a "blue shift" tendency due to the band filling effect (*i.e.*, the Fermi level separation becomes larger as more subbands are filled). The blue shift effect is usually stronger than the red shift effect due to bandgap shrinkage, and one often observes a blue shift in experiment [28] [29].

In the case of anisotropic parabolic approximation for strained quantum wells, the optical gain formulas can simply be extended by using the in-plane quantities (labeled "x") as follows. The local gain due to a transition from the j^{th} in the conduction band to i^{th} in the valence band is

$$g_{ij}(E_{ij}^0) = \left(\frac{2\pi}{\hbar}\right) \mid H_{ij} \mid^2 (f_j' - f_i') \left(\frac{\epsilon_1}{\overline{n}c}\right) \rho_{ij}^x, \tag{8.37}$$

$$\rho_{ij}^x = \rho_{ij}^{x0} h(\hbar\omega - E_{ij}^0), \tag{8.38}$$

$$\mid H_{ij} \mid^2 = \left(\frac{q}{m_0}\right)^2 \left(\frac{2\hbar\omega}{4\epsilon_1\epsilon_0\omega^2}\right) M_{ij}^2, \tag{8.39}$$

where f_i and f_j are the Fermi functions for the i^{th} and the j^{th} levels, respectively. The gain function is the sum of the g_{ij} over all allowed transitions.

Similar to the treatment of the gain function, the spontaneous emission rate can be written as

$$r_{sp}^{qw}(E) = \sum_{i,j} \left(\frac{2\pi}{\hbar}\right) \mid H_{ij} \mid^2 f_j'(1 - f_i')D(E)\rho_{ij} \ , \tag{8.40}$$

where $D(E)$ is the optical mode density.

The program assumes that the dipole moment of a strained quantum well system is the same as that for its unstrained counterpart. This assumption should be accurate near the Γ point. As in the case of the unstrained quantum well, the overlap integral in the dipole moment is assumed to be k-independent in this model.

8.3.4 Landsberg model of gain broadening

The simplest model for gain broadening is to use the following Lorentzian function in a convolution integral:

$$g(E) = \int g^0(E_1) F_l(E_1 - E) dE_1, \tag{8.41}$$

where g^0 is the gain function without broadening. The Lorentzian function is given by the expression:

$$F_l(E_1 - E) = \frac{1}{\pi} \frac{\Gamma_0}{\Gamma_0^2 + (E_1 - E)^2}, \tag{8.42}$$

where Γ_0 is the half-width of the broadened energy level and is assumed constant. The assumption of constant Γ_0 means that the gain spectrum is broadened to the same degree across the whole spectrum. The advantage of this model is that it is relatively simple to implement and results in a most stable solution in the 2D simulations.

According to Landsberg's model, the broadening across the spectrum should not be uniform.

The basic mechanism for Landsberg's model is carrier-carrier scattering. The original formulas were derived for bulk material [30] [31] and were extended successfully by Zielinski *et.al.* [28] [29] for quantum well laser modeling. This model is phenomenological and is the most simple form of non-Lorentzian shape function. The success of such a model in comparison with experiment is very encouraging [28] [29].

According to Landsberg's model [30] [31] [32], the half-width Γ is a complicated function of the transition energy which has its maximum value at the bottom of the band (or band edge) and decreases to zero as the energy approaches the quasi-Fermi level. Therefore the half-width is a function of both the transition energy and the carrier density. The software has used the following approximation according to Martin and Stormer [32]:

$$F_l(E_1 - E) = \frac{1}{\pi} \frac{\Gamma(E_1)}{\Gamma(E_1)^2 + (E_1 - E)^2}, \tag{8.43}$$

where

$$\Gamma(E_1) = \Gamma_0 \left[1 - 2.229 \frac{E_1}{E_{fn} - E_{fp}} + 1.458 \left(\frac{E_1}{E_{fn} - E_{fp}} \right)^2 - 0.229 \left(\frac{E_1}{E_{fn} - E_{fp}} \right)^3 \right]$$

$$for\ E_{fn} - E_{fp} \geq 0\ \&\ 0 \leq E_1 \leq E_{fn} - E_{fp} \tag{8.44}$$

For bulk material a Γ_0 value of 1.2 meV was found. For quantum wells the broadening is much larger (up to 30 meV). The implementation of this model is important for accurate modeling of the gain spectrum, especially for quantum wells. The maximum broadening Γ_0 in the Landsberg model is treated as a fitting parameter for the case of quantum wells.

Note that Equation 8.44 is valid only when the peak gain is positive, *i.e.*, only when the difference between the quasi-Fermi levels is greater than the bandgap of the quantum well subbands. The transition energy must also be between the band gap and this difference. To extend the spectrum to other conditions, the formulas proposed by Zielinski *et.al.* [28] are used:

$$g(E) = I_{conv}(g^0, E) \quad forE < E_{fn} - E_{fp}, \tag{8.45}$$

$$g(E) = I_{conv}(g^0, E) + g^0(E) \quad forE \geq E_{fn} - E_{fp}, \tag{8.46}$$

with

$$I_{conv}(g^0, E) = \sum_{i,j} \int_{E_{ij}}^{E_{fn}-E_{fp}} \frac{1}{\pi} \frac{\Gamma(E_1) dE_1}{\Gamma(E_1)^2 + (E_1 - E)^2}$$

$$for\ E_{fn} - E_{fp} \geq E_{ij}, \tag{8.47}$$

and

$$I_{conv}(g^0, E) = 0$$

$$for\ E_{fn} - E_{fp} < E_{ij}. \tag{8.48}$$

The two-piece function is continuous across the spectrum because because $g^0(E) = 0$ when $E = E_{fn} - E_{fp}$.

In the current version of the software, no broadening is imposed on the gain of the bulk material for two reasons. First, the level broadening coefficient is relatively small (1.2meV), and second, the un-broadened gain spectrum does not have a sharp edge near the bandgap as in the quantum well case. Therefore the error caused by not using broadening in bulk material gain is relatively small.

8.3.5 Landsberg broadening and experimental data

This Section gives an example of how the user can calibrate the gain model with experimental data. For a description of the program's commands and syntax, please refer to the Reference Manual. The gain function here uses the Landsberg broadening formula, discussed in Section 8.3.4, which is based on well established physical models for emission spectra from recombination of electron-hole pairs. It uses an energy and carrier density dependent half width in the Lorentzian broadening function. The broadening results in a more symmetric quantum well gain spectrum than the Lorentzian broadening function with constant Γ. The Landsberg model is usually in close agreement with experiment and is strongly recommended when an experimental gain spectrum is used for the purpose of calibration.

The experimental gain data are taken from Ref. [29]. Since the details of the multiple quantum well system were not given in the reference (except for the composition and well width), the optical confinement factor was fitted according to the information accompanying the published data.

Here, two sets of experimental data for quantum well and bulk InGaAs/InP are used to compare with the results from the gain module of the software. Detailed input files can be found in the standard examples that come with installation of the software. The input files are named "lands.gain", which models the multiple quantum well device, and "lands2.gain", which models the bulk device.

Implementation of the Landsberg broadening model is very simple. The parameter **broadening=landsberg** is set in the statement **active_reg** in "*.sol". First we consider an InGaAs/InP quantum well structure of well thickness 0.0105 μm. A carrier scattering lifetime of 0.2×10^{-13} seconds is used, and a background loss of 45 cm^{-1} is assumed for this structure. The result of the program's gain spectrum together with the experimental data from Ref. [29] is given in Figure 8.5.

In another example, the bulk InGaAs/Inp structure is considered. The background loss is again assumed to be 45 cm^{-1}. The optical gain spectrum for the bulk material along with experimental data from Ref. [29] is given in Figure 8.6.

To plot experimental data on the same plot as the simulated gain curve, the parameter **include_data** is used in the file "lands.gain". The quantum well model of the software agrees with the published data very well using the same carrier density as that suggested in Ref. [29]. The carrier density for the bulk material is slightly different from the value suggested in Ref. [29] in order to obtain a good fit of the data. This slight difference may be due to the lack of broadening for bulk material gain in the program. Note that the current version of the software assumes the broadening effect is negligible for bulk material (which is a good approximation compared with quantum well gain). In both cases the quality of the fit and the values of the parameters are reasonable according to the information given in Ref. [29].

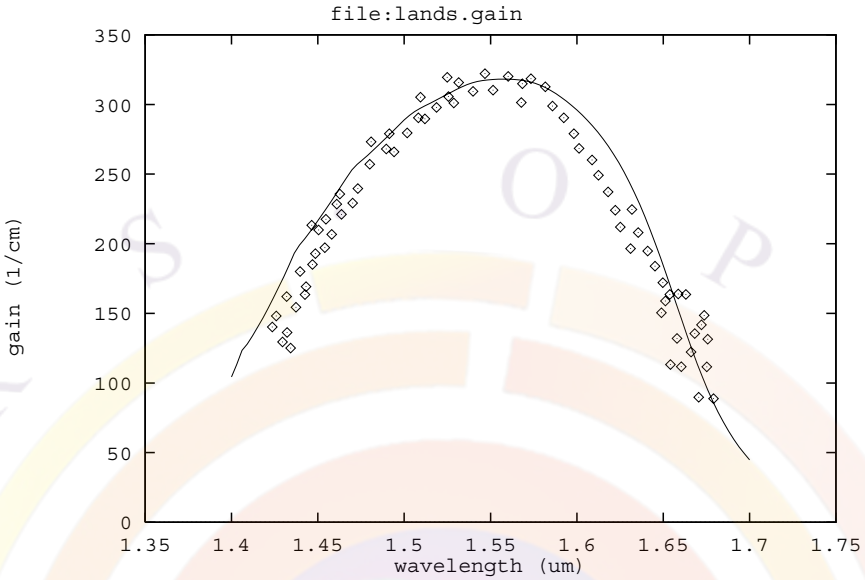


Figure 8.5: Gain spectrum of InGaAs/InP quantum well structure as compared with experimental data. Landsberg type gain broadening is used in the gain model.

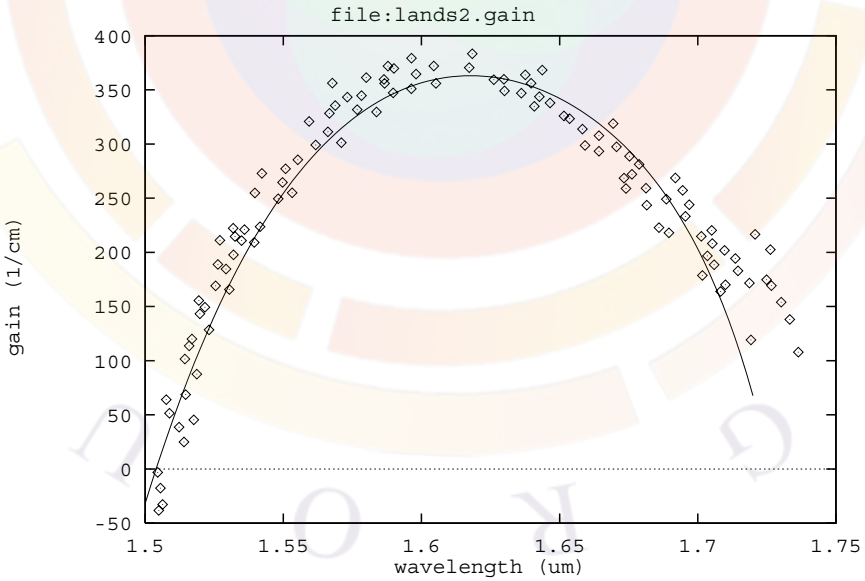


Figure 8.6: Gain spectrum for bulk InGaAs/InP as compared with experimental data.

8.3.6 Gain integral with valence mixing

In the case of valence mixing, the valence bands are not parabolic and we must use a different formula for the optical gain spectrum. Based on theories developed in Ref. [33], the gain spectrum can be expressed in numerical integration over k_t.

$$g(E) = \frac{g_0}{2\pi t E} \sum_{i,j} \int_0^\infty \frac{(\pi/\Gamma) f_{dip}(k_t) M_b (f_j - f_i) dk_t^2}{1 + (E_{cj}(k_t) - E_{kpi}(k_t) - E)^2/\Gamma^2} \tag{8.49}$$

where t is the quantum well thickness; $\Gamma = \hbar/\tau_{scat}$ is broadening due to intraband scattering relaxation time τ_{scat}; E_{cj} is the jth conduction subband; E_{kpi} is the ith valence subband from the k.p calculation; the sum is over all possible valence and conduction subbands; g_0 is a constant defined as

$$g_0 = \frac{\pi q^2 \hbar}{\varepsilon_0 c m_0^2 \overline{n}} \tag{8.50}$$

where q is free electron charge; $\overline{n}$ is the real part of the refractive index; all other symbols have their usual meanings.

M_b is the bulk dipole momentum given by:

$$M_b = \frac{1}{6} \frac{m_0 q}{m_c} \frac{E_{g0}(E_{g0} + \Delta_{so})}{E_{g0} + (2/3)\Delta_{so}} \tag{8.51}$$

where E_{g0} is the unstrained bandgap; m_c is the effective mass of the conduction band; Δ_{so} is spin-orbit coupling energy.

The dipole factor due to non-parabolic subbands are given by the following overlap integral for the case of symmetric well:

$$\begin{aligned} f_{dip} &= (3/2)[< c|g_1(k_t, z) >^2 + (1/3) < c|g_2(k_t, z) >^2] \quad (TE) \\ f_{dip} &= 2 < c|g_2(k_t, z) >^2 \quad (TM) \end{aligned} \tag{8.52}$$

where $|c>$ is the conduction band wave function; $|g_1>$ and $|g_2>$ are the valence band envelop function.

f_j and f_i are the Fermi functions expressed as follows:

$$\begin{aligned} f_j^{-1} &= 1 + exp\left[\frac{1}{kT}\left(E_{cj0} + \frac{\hbar^2 k_t^2}{2 m_0 m_c} - E_{fn}\right)\right] \\ f_i^{-1} &= 1 + exp\left[\frac{E_{kpi}(k_t)}{kT}\right] \end{aligned} \tag{8.53}$$

where E_{cj0} is the bottom of jth conduction subband.

8.3.7 Non-linear gain model

Conventional optical gain is a concept based on linear response theory, *i.e.*, only first order perturbation is used to treat a system under an external optical field. For semiconductor lasers, it is often necessary to include the higher order effects (or non-linear effects). The optical gain tends to decrease as a function of photon density when non-linear effects are included. Such an effect is also called gain suppression.

The non-linear gain effect is responsible for the spectral hole burning effect in semiconductor lasers [34].

The non-linear gain effect is commonly expressed as [34].

$$g = g_0(1 - \epsilon S). \tag{8.54}$$

where ϵ is the non-linear gain suppression coefficient which is not to be confused with the dielectric constant, and S is the photon density. Since the correction term is typically of the order of a few percent, one can use the following alternative form:

$$g = \frac{g_0}{1 + \epsilon S} \tag{8.55}$$

Such a formula has the numerical advantage that the correction term can never make the optical gain negative. Equation (8.55) has the same format as the gain saturation for a simple, two-level system [35].

In the case of multi-lateral modes, the photon density S is replaced by the sum of the densities of all the modes.

8.4 Refractive Index Model

8.4.1 Index change due to interband transitions

In many applications, especially DFB lasers in fiber optic communications, the change of refractive index as a function of injection carriers is an important parameter. It affects the laser linewidth, FM noise and modulation characteristics. Therefore it is desirable for laser designers to evaluate the index change for a particular device structure. The program implements such a change in refractive index due to interband optical transitions based on the standard Kramers-Kronig formulas.

A modified version of the Kramers-Kronig formula relating the real and imaginary parts of the refractive index of an arbitrary material can be written as [36]:

$$\Delta\overline{n}(E) = -\frac{c\hbar}{\pi e}\int_0^\infty \frac{\Delta g(E') - \Delta g(E)}{(E' - E)(E' + E)}dE' \tag{8.56}$$

The integral above is difficult to evaluate numerically because of the singularity at $E = E'$. To obtain an accurate integral, we separate it into the following three parts:

$$\begin{aligned}\Delta\overline{n}(E) &= -\frac{c\hbar}{\pi e}\int_0^{\infty}\frac{\Delta g(E')-\Delta g(E)}{(E'-E)(E'+E)}dE' \\ &= -\frac{c\hbar}{\pi e}\int_0^{E-\varepsilon}\frac{\Delta g(E')-\Delta g(E)}{(E'-E)(E'+E)}dE' \\ &\quad -\frac{c\hbar}{\pi e}\int_{E+\varepsilon}^{\infty}\frac{\Delta g(E')-\Delta g(E)}{(E'-E)(E'+E)}dE' \\ &\quad -\frac{c\hbar}{\pi e}\int_{E-\varepsilon}^{E+\varepsilon}\frac{\Delta g(E')-\Delta g(E)}{(E'-E)(E'+E)}dE' \qquad (8.57)\end{aligned}$$

where ε is a small number. The third term may be written as

$$-\frac{c\hbar}{\pi e}\frac{dg}{dE}\int_{E-\varepsilon}^{E+\varepsilon}\frac{1}{E'+E}dE' \qquad (8.58)$$

and this can be evaluated analytically.

To ensure the correct implementation of the standard model, we have compared our calculations with experimental data obtained by C.H. Henry *et.al.* [36] for a bulk GaAs material system. The calculation is done at transparency conditions. Note that in Henry's paper, the transparent density was estimated to be about $0.8\times10^{24}m^{-3}$ while our calculation gives a value of $1.5\times10^{24}m^{-3}$. Since our value of transparent density agrees with values from a more recent theory by Vahala *et.al.*[37], we have decided to use this theoretical value here. Our result (also included in LASTIP's examples under examples/index_change/henry.gain) agrees well with that from Henry's paper (see Figure 8.7).

Once the index change is calculated, it is trivial to compute the alpha factor (or the linewidth enhancement factor), which is very important for DFB lasers and related devices used in telecommunications. Again we have calculated the index change for bulk GaAs (see Figure 8.8) and find that the results are similar to those from theoretical calculations by Vahala *et.al.* [37].

It is clear from Figures 8.7 and 8.8 that the index change is very sensitive to the wavelength being used. Therefore any effort to improve the alpha factor or the index change should take into account the spectral dependence. A small shift in the operating wavelength may mean a large improvement or degradation.

8.4.2 Other models of index change

Crosslight software also implemented for the following forms of index change model.

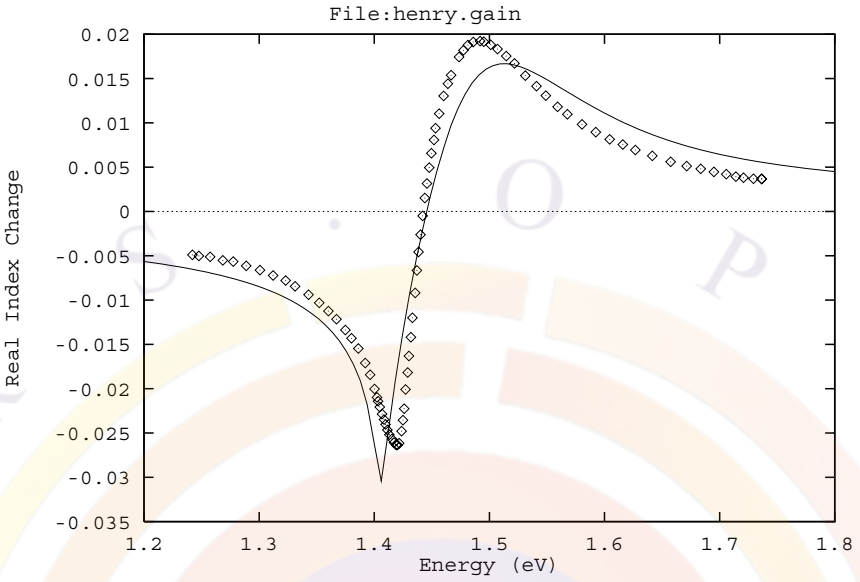


Figure 8.7: Index change with respect to equilibrium. The index change is evaluated at transparent conditions for bulk GaAs material. The points are experimental data from the work of C.H. Henry *et.al*.

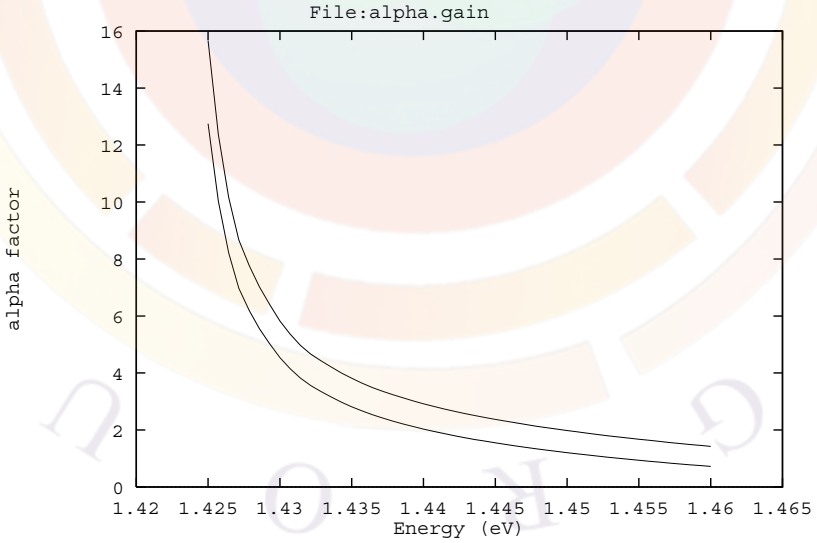


Figure 8.8: The linewidth enhancement factor (alpha factor) as a function of wavelength at two different carrier densities ($1.8 \times 10^{24} m^{-3}$ and $2.5 \times 10^{24} m^{-3}$).

- 1) Use a line with enhancement factor α_{lw} to compute the index change based on local gain change as follows.

$$\Delta\overline{n}(E) = -\frac{\alpha_{lw}}{2k_0}(g - g_0) \tag{8.59}$$

- 2) Free-carrier/plasma part [1]:

$$\Delta\overline{n}(E) = -\frac{q^2 n}{2m_e\omega^2\varepsilon_0\overline{n}} \tag{8.60}$$

- 3) Linear model:

$$\Delta\overline{n}(E) = [a_n^{(a)} + b_n^{(a)}(T - 300)](n - n_0) + [a_p^{(a)} + b_p^{(a)}(T - 300)](p - p_0) \tag{8.61}$$

8.5 Optical Gain with Many Body Coulomb Interaction

Coulomb interaction between electrons and holes confined in an active region of laser diodes can significantly enhance it's optical gain value and influence the emission spectrum. In particular, it has been reported that, Coulomb enhancement is stronger in short wavelength blue-green ZnCdSe and InGaN quantum well lasers compare to the long wavelength III-V lasers, due to the lower value of dielectric constant in wide-bandgap materials[38]. We discover that another major reason for wurtzite material system to have higher many-body (MB) effect is that the due to imbalance of density of states between the conduction and the valence bands, the occupancy is much different in the two bands, which causes the enhancement factor to increase.

The role of Coulomb interaction between electrons and holes confined into an active region of laser diode can be explained within a simple physical picture; while Coulomb force attracts electrons and holes at a closer distance, e-h radiative recombination rate increases, which manifests in enhancement in intensity of spontaneous emission as well as in gain magnitude. In addition Coulomb interaction modifies spontaneous emission and gain spectra of laser diodes by shifting the gain peak towards lower energies.

One of the inherent characteristics of electron-hole plasma is screening of the Coulomb potential. The modified, screened Coulomb potential $V_s(q,\omega)$, as described in the momentum and frequency domain, relates to an unscreened Coulomb potential, V_q, as:

$$V_s(q,\omega) = \frac{V_q}{\varepsilon(q,\omega)}, \tag{8.62}$$

[1] Starting with version 2010, $\overline{n}$ is used instead of ε in the denominator to fix a longstanding error in the model

where $\varepsilon(q,\omega)$ is the longitudinal dielectric function. If one neglects the dynamic aspect of screening contained in a frequency dependence of the dielectric function,$\varepsilon(q,\omega)$, one can obtain the following dielectric function using the plasmon-pole approximation[39–41]:

$$1/\varepsilon(q) = 1 - \omega_{pl}^2/\omega_q^2, \tag{8.63}$$

where ω_{pl} is the plasma frequency, while ω_q is an effective plasmon mode frequency.

In particular, in case of electrons and holes confined in a quantum well, one should use a two-dimensional (2D) Coulomb interaction, given by:

$$V_q = e^2/2\varepsilon_0 q, \tag{8.64}$$

and 2D plasma frequency:

$$\omega_{pl}^2 = \frac{e^2 N_{2D} q}{2\varepsilon_b \varepsilon_0 m_r} \tag{8.65}$$

where $N_{2D} = n \times w_{qw}$ is the 2D carrier density in a quantum well width of w_{qw}, ε_b is the static background dielectric constant in the area of quantum well; $1/m_r = 1/m_e + 1/m_h$, is the reduced effective mass of conduction band electrons and valence band holes in directions parallel to the quantum well plane. The effective plasmon frequency ω_q is given in 2D case by:

$$\omega_q^2 = \omega_{pl}^2(1 + q/\kappa^2) + \frac{C_{pl}}{4}\left(\frac{\hbar q^2}{2mr}\right)^2, \tag{8.66}$$

where C_{pl} is a unitless constant typically in a range 1-4, while κ^2 is the square of inverse screening length for electrons and holes confined in the quantum well, which can be derived as[42]:

$$\kappa^2 = e^2/(\pi\hbar^2\varepsilon_b\varepsilon_0)\sum_{i,j}[m_{cj}f_e(E_{cj})/d_{cj} + m_{vi}f_h(E_{vi})/d_{vi}], \tag{8.67}$$

Here, summation $\sum_{i,j}$ is over all sub-bands characterized by energies, parallel masses, and a wave-function average width, E_{cj}, m_{ej}, d_{cj}, and E_{vi}, m_{vi}, d_{vi} for jth conduction band and ith valence band.

$$d_{nk} = \pi\hbar/\sqrt{2m_n E_{nk}}, \tag{8.68}$$

where index n= c or v for conduction and valence bands, respectively.

Consider, first, the effect of carrier Coulomb interaction on the spectrum of e-h inter-band recombination, often ascribed as bandgap-renormalization effect. It's physical meaning is "band-gap narrowing" effect, which can partially be explained by the screening of conduction band electron-electron Coulomb repulsion by positively charged valence band holes available near electrons at close proximity. This Coulomb Hole (CH) contribution to the band-gap reduction is approximately given in 2D by:

$$\Delta E_{CH} = -2E_0 a_0 \kappa \times ln\left[1 + \sqrt{32\pi N_{2D}/C_{pl}\kappa^3 a_0}\right] \tag{8.69}$$

where a_0 is the Bohr radius of quantum well electron-hole exciton, given by:

$$a_0 = 4\pi\hbar^2\varepsilon_b\varepsilon_0/e^2 m_{rj} \tag{8.70}$$

and E_0 is corresponding effective Rydberg energy:

$$E_0 = \hbar^2/(2m_{rj}a_0^2). \tag{8.71}$$

Additional contribution to Coulomb hole band-gap renormalization is a self-energy correction to e-h plasma due to exchange interaction, which also results in reduction of band-gap energy as:

$$\Delta E_{gSX} = -\frac{2E_0 a_0}{\kappa}\int_0^\infty dkk\frac{1 + C_{pl}\kappa a_0 k^2/32\pi N_{2D}}{1 + q/\kappa + C_{pl}a_0 k^3/32\pi N_{2D}}[f_e(E_{cj\mathbf{k}}) + f_h(E_{vj\mathbf{k}})] \tag{8.72}$$

Therefore the net band-gap narrowing due to Coulomb-hole and exchange interaction is:

$$\Delta E_g = \Delta E_{gCH} + \Delta E_{gSX} \tag{8.73}$$

and the renormalized band-gap:

$$E_{g\ ren} = E_g + \Delta E_g \tag{8.74}$$

This effective band-gap energy reduction can be observed in a red shift of all recombination energy values $E_cv(\mathbf{k})$ specified at a given parallel momentum $\mathbf{k}$ of jth-conduction and ith-valence quantum well subbands, as:

$$E_{cv}(\mathbf{k}) = E_g + \Delta E_g + E_{cj\mathbf{k}} + E_{vi\mathbf{k}} \tag{8.75}$$

where $E_{cj\mathbf{k}}$ and $E_{vi\mathbf{k}}$ are energies of electrons and holes from jth-subband of conduction and ith-subband of valence band quantum well in the active region.

The spectral wave function, $g(E_{cv})$ consists of a sum of $g_{ji}(E_{cv})$ contributions from transitions between jth-subband electrons and ith-subband holes, given by [42]:

$$\begin{aligned} g_0(E_{cv}) &= \sum_{j,i} g_{ji}(E_{cv}) = \pi e^2\hbar/(\varepsilon_0 c^3 m_0^2 n_{av})\sum_{j,i}(1/E_{cv})|M_{ji}(E_{cv})|^2\rho_{r,ji} \\ &\quad \times[f_e(E_{cj\mathbf{k}}) + f_h(E_{vj\mathbf{k}}) - 1], \end{aligned} \tag{8.76}$$

where $|M_{ji}(E_{cv})|^2$ is the transition matrix element at transition energy E_{cv} , $\rho_{r,ji}$ is the reduced density of states between jth-conduction subband, and ith-valence subband; n_{av} is the background refractive index. In practice laser diode spectra are broadened, primarily due to the strong carrier-carrier scattering and other intraband scattering mechanisms, o as well as lattice disorder. Consequently, gain becomes a convolution of an un-broaden gain function of Eq. (8.76) with a broadening function L(x), as:

$$g_B(\hbar\omega) = \int_{E_{g0}}^\infty g_0(E_{cv})L(\hbar\omega - E_{cv})dE_{cv} \tag{8.77}$$

where E_{g0} is an energy of the lowest e-h recombination transition between ground levels of quantum well in conduction and valence band of active region. Typically, a Lorentzian broadening function shape is assumed in L(x), given by:

$$L(\hbar\omega - E_{cv}) = (1/\pi)\Gamma_{cv}(\hbar\omega, E_{cv})/[\Gamma_{cv}^2(\hbar\omega, E_{cv}) + (\hbar\omega - E_{cv})^2], \tag{8.78}$$

and parameterized by Lorentzian width, Γ_{cv}, which is related to scattering time by $\Gamma_{cv} = \hbar/\tau_{cv}$ (note that some sources define Γ_{cv} as equal $\hbar/2\tau_{cv}$). For simplicity, we choose Γ_{cv} to be a constant parameter in our calculations (i.e., we neglect its dependence upon $\hbar\omega$ and E_{cv} energies).

Finally, if Coulomb interaction is included in gain spectral function derivation, $g(\hbar\omega)$ of equation (8.77) becomes:

$$g(\hbar\omega) = Real\left\{\int_{E_{g0}}^{\infty} \frac{g_0(E_{cv})}{1 - q_1(E_{cv}, \hbar\omega)}\left[1 - i\frac{(E_{cv} - \hbar\omega)}{\Gamma_{cv}}\right] L(\hbar\omega - E_{cv})dE_{cv}\right\} \tag{8.79}$$

where

$$q(E_cv, \hbar\omega) = \frac{-ia_0E_0E_{cv}}{\pi\kappa|M_{ji}(E_{cv})|} \int_0^{\infty} dk'k' \frac{|M_{ji}(E_{cv'}|}{E_{cv'}} \times \frac{f_e(E_{cj\mathbf{k}'}) + f_h(E_{vi\mathbf{k}'}) - 1}{(\Gamma_{cv} + i(E_{cv} - \hbar\omega)} \times \Theta(\mathbf{k}, \mathbf{k}'), \tag{8.80}$$

and

$$\Theta(\mathbf{k}, \mathbf{k}') = \int_0^{2\pi} d\theta \frac{1 + C_{pl}\kappa a_0 q^2/32\pi N_{2D}}{1 + q/\kappa + C_{pl}\kappa a_0 q k^3/32\pi N_{2D}}, \tag{8.81}$$

and

$$q^2 = k^2 + k'^2 - 2kk'cos\theta, \tag{8.82}$$

and θ is the angle between in-plane vectors **k** and **k'**.

The Eq. (8.79) gives the final integral formula for Coulomb enhanced gain spectral function, numerically calculated in Crosslight simulation software. User has an option to include, band-gap renormalization effect as defined in Eq. (8.73) and independently Coulomb contribution as formulated in Eq.(8.79), along with a custom definition of ε_b, C_{pl}, Γ_{cv}, and n_{av} values as input parameters.

8.6 Local Gain and Optical Confinement

In previous sections, we assume a flat band situation with isolated quantum wells and we assume the carrier spreads uniformly throughout the well, except with slight tilting by the applied field. Similarly the local optical gain is assumed to be uniform with slight modification by the Fermi levels deviating from equilibrium.

The local optical gain is written in terms of the joint density of state

$$\rho_r = \frac{m_r}{\pi t \hbar} \tag{8.83}$$

times a Fermi function term which is causing the small change in the uniform gain. Here m_r is the reduced mass and t is the well thickness.

Without loss of generality, we consider only the z-dimension here. We define the modal gain as an integral of the local gain as follows:

$$g_m = \frac{\int_{QW} g(z)w(z)^2 dz}{\int w(z)^2 dz} \tag{8.84}$$

Using a slow varying assumption, one can pull the g(z) out of the integrand and define an optical confinement factor:

$$\Gamma = \frac{\int_{QW} w(z)^2 dz}{\int w(z)^2 dz} \tag{8.85}$$

Please note that Crosslight simulator uses only the integral form in Eq. (8.84). The Γ factor is NOT being used. It is printed only for comparison with other theories.

In a self-consistent simulation of optical transition within an arbitrary MQW region, wells and barriers are not well defined and we can no longer decide a well thickness. For example, if one wishes to grade the composition near the well, it is no longer possible to draw a line between well and barrier and the use of QW thickness loses its meaning. The optical gain written in a flat band model would have to be re-derived using the overlap of wave functions.

According to the Fermi's golden rule (see for example, [43]), we decide that the modal optical gain is proportional to an integral involving the envelop functions, $f_c(z)$ and $f_v(z)$, and the optical wave (treated as scalar here) w(z):

$$g_m \propto M_b^2 \Big\{ \int f_c(z)w(z)f_v(z)dz \Big\}^2 \tag{8.86}$$

The above equation is the most accurate definition of modal gain integral as far as spatial overlap variation is concerned since it is derived from the first principle. Please note that atomic part of the wave function has been included in the bulk dipole moment M_b^2. However, it is a common practice to use a separate the optical field intensity $w^2(z)$ which can be measured in near field and far field. Also, it is convenient to be able to isolate a local gain from the above integral so that we can "see" how it varies with distance. We proceed as follows:

We assume that total electron or hole wave function (envelop times the atomic part) is somehow localized and will be uncorrelated over a distance greater than the atomic

size. For convenience of discussion, we replace the integral by a sum over a mesh grid with grid spacing larger than the correlation length:

$$g_m \propto \left\{ M_b^2(z_i) \sum_i f_c(z_i) w(z_i) f_v(z_i) \Delta z_i \right\}^2 \tag{8.87}$$

We apply the square and let the cross terms cancel each other due to the random phase of the total electron/hole wave function. Then, we are able to write (with proper normalization):

$$g_m \propto \sum_i M_b(z_i)^2 f_c(z_i)^2 w(z_i)^2 f_v(z_i)^2 \Delta z_i \tag{8.88}$$

At this point, the modal gain comes out as a well defined quantity given the envelop function and the optical wave function. However, in a drift-diffusion finite element analysis, it is desirable to introduce a local gain since we wish to see a distribution of the local gain over a complex MQW so that stimulated recombination and spontaneous radiative recombination can have a continuous distribution (instead of appearing as a block box model). Comparison of Eqs. (8.84) and (8.88) leads to the following local gain variation.

$$g(z) \propto f_c(z)^2 f_v(z)^2 \tag{8.89}$$

This is a simple statement that the local optical gain is proportional to overlap of electron/hole carrier densities. Therefore, it makes sense to define a position dependent joint density of state (JDOS) as being proportional to such an overlap:

$$\rho_r(z) = \frac{< m_r > H(z)}{\hbar \pi} \tag{8.90}$$

where $< m_r >$ is the reduced mass averaged over the wave function overlaps:

$$< m_r >= \int m_r(z) H(z) dz \tag{8.91}$$

and the weighting function is defined as

$$H(z) = f_c^2(z) f_v^2(z) / \int f_c^2(z) f_v^2(z) dz \tag{8.92}$$

Compared with the conventional QW optical gain using 2D JDOS, the key difference is now we replace 1/t (t=well thickness) by the weighting function H(z). Please note that the weighting function has a unit of 1/length and may be regarded as a position dependent effective well thickness. At the center of the well, we have a tightly confined wave function and thus a thinner well thickness. Given the revised $\rho_r(z)$ above, we can define a position dependent local gain for all of the gain formulas based on effective mass theory.

For gain formula with valence mixing such as that in Eq. (8.49), the modification is similar: we once again replace 1/t (t=well thickness) by H(z). We do not need to use any averaged effective mass here since the gain integral for valence mixing already includes the effect of spatial averaging by using a solution of the energy band obtained over the whole potential profile.

Finally, we outline the actual implementation of optical gain model in a self-consistent solution as follows. To make the modal optical gain to be as close to the rigorous definition in Eq. 8.86, we first solve the envelop wave functions for electrons and holes over the whole coupled MQW region. Then we compute the total modal gain dipole moment by evaluating the wave function overlap before applying the square as in Eq. 8.86. Finally, we use the weighting function H(z) to define a position dependent JDOS in the local gain expression before performing the modal gain integral using $w(z)^2$. To be more specific, the self-consistent modal gain is written as

$$g_m = \frac{\int g(z)w(z)^2 dz}{\int w(z)^2 dz} \tag{8.93}$$

The local gain is defined as

$$g(z) = g_0 H(z) A_{ij} M_b^2 O_{ij}^2 \tag{8.94}$$

where g_0 is a gain coefficient depending on modal index and wavelength, M_b is the bulk dipole moment, O_{ij} is the overlap integral between hole wave function of level i and electron wave function of level j, and A_{ij} is the corresponding QW dipole moment enhancement factor. Compared with the original formula from the Fermi-Golden rule, the only approximation we make is to use $w(z)^2$ directly in the modal gain integral instead of using $w(z)$ before applying the square. We expect the difference to be minor for slowly varying fields. Please note that using H(z) of a different form has no consequence for the modal gain as long as it is normalized to unity. The advantage of this separation of $w(z)^2$ makes it easy to reduce to the classical definition of modal gain and optical confinement factor in the limit of bulk active layers.

The same weighting function H(z) has been used in the definition of local spontaneous recombination which directly affect the radiative source profile within a complex MQW system in a self-consistent solution.

8.7 Exciton Electro-absorption in Quantum Wells

8.7.1 Introduction

It is well known, from semiconductor physics, that an inter-band light absorption is not a simple derivative of a free electron-hole (e-h) pair generation. Both e-h

energy, as well as their motion are affected by mutual Coulomb attraction, and some of those effects were already considered in a context of many-body Coulomb correction to QW optical gain (Section 8.5 of this manual). In extreme case, strongly correlated, hydrogen atom-like, bound electron-hole state, called exciton, can be formed in semiconductors. It is characterized by a ground state binding energy, or effective Rydberg :

$$R_y^* = \frac{m_r e^4}{2(4\pi\epsilon_s\hbar)^2}, \tag{8.95}$$

and a spacial average separation distance, or Bohr radius:

$$a_0^* = \frac{4\pi\epsilon_s(\hbar)^2}{m_r e^2}, \tag{8.96}$$

where $m_r = m_e m_h/(m_e + m_h)$ is the electron-hole reduced mass, ϵ_s is a substrate static dielectric permittivity, ($\epsilon_s = \kappa\epsilon_0$, with κ dielectric constant). Excitons can be experimentally observed in a near band-gap absorption edge spectral region, and their spectrum can be effectively varied in a controlled fashion (modulated) by application of an external electrostatic field. This can be directly utilized in electro-absorption modulators for fiber-optics application. In particular, an effective modulator can be based on Quantum-Confined Stark Effect (QCSE)effect observed in thin QW and MQW heterostructures. Due to dimensional confinement realized in these structures, such an effect is quite pronounced, implying strong overlap of e-h wave function and enhancement of the oscillator strength of interband transitions, along with the rise in exciton binding energy. For example in case of GaAs/AlGaAs QW structures excitonic absorbtion becomes an important distinct component of a near band-edge absorption spectra at room temperature, while GaAs bulk excitons are typically observed only at low temperatures.

At higher densities (strong excitation limit) attractive Coulomb force between electron-hole pairs becomes screened by free carriers, and excitons ultimately decompose into free electron-hole plasma and absorption saturates. Therefore, proper model of screening in quasi 2D electron-hole plasmas, confined into semiconductor QW, and MQW region, becomes important in electro-absorption modulator modeling especially at high excitation limit.

8.7.2 Screened Coulomb Potential in Quantum Wells

Many-body physics and their theoretical treatment of Coulomb interaction in quasi 2D QW electrons and hole systems is quite complex, and is usually treated within reasonable approximations. A well known 3D screened Coulomb interaction potential, given by Yukawa potential with screening constant parameter, κ is as follow:

$$V_s^{3d}(r) = \frac{q^2}{4\pi\epsilon_s}\frac{\exp(-\kappa r)}{r} \tag{8.97}$$

often becomes inadequate when applied to calculation of exciton binding energy in 2D electron-hole heterostructure systems. In particular, 3D-Coulomb potential overestimates long range screening effect, while a 2D-Screened Coulomb potential

$$V_s^{2d}(r) = \frac{q^2}{4\pi\epsilon_s}\left[\frac{1}{r} - \kappa\int_0^\infty \frac{dqJ_0(qr)}{(q+\kappa)}\right], \tag{8.98}$$

underestimates it. Here, zero-order Bessel function, $J_0(qr)$, is used. It turns out that for thin electron and hole layers as implemented in semiconductor inversion layers, accumulation layers or MQW heterostructures, we need an intermediate potential $V_s(r)$ between $V_s^{2d}(r)$ and $V_s^{3d}(r)$, which we shall refer to as a *quasi-2D screened Coulomb potential*. The electron and hole spatial density distribution across such layers (in z-direction) is defined by density probability function, given by envelope wave function square, as $|\phi(z_e|^2$, and $|\phi(z_h|^2$. First we shall use a single pole approximation at low frequency limit to derive a screened Coulomb potential in momentum space, which can be expressed as:

$$V_s(q) = \frac{V_q^{2d}}{\varepsilon(q)}F(q) \tag{8.99}$$

where V_q^{2d} is an unscreened 2D Coulomb potential

$$V_q^{2d} = \frac{e^2}{2\varepsilon_b\varepsilon_0 q}, \tag{8.100}$$

$F(q)$ is a form-factor, which represents the deviation of the Coulomb potential from a perfect 2D case (of zero thickness layer limit). The form factor is usually expressed in some forms of overlap integral of the quantum well wave functions. The mathematical details of its evaluation are out of the scope of this document.

The dielectric function:

$$\frac{1}{\epsilon(q)} = 1 - \frac{\omega_{pl}^2}{\omega_q^2}, \tag{8.101}$$

and plasma frequency

$$\omega_{pl}^2 = \frac{e^2}{2\varepsilon_b\varepsilon_0} \times qF(q)[\sum_i \frac{n_{c,i}}{m_{c,i}} + \sum_j \frac{n_{v,j}}{m_{v,j}}]. \tag{8.102}$$

Again, similarly to Eq. (8.66) of Section 8.5, the effective plasmon frequency is given by:

$$\omega_q^2 = \omega_{pl}^2\left(1 + \frac{1}{\kappa F(q)}\right) + \frac{C_{pl}}{4}\left(\frac{\hbar q^2}{2m_r}\right)^2, \tag{8.103}$$

Subsequently, From $V_s(q)$ of Eq. (8.99) we can find quasi-2D Coulomb potential in a real space, using Fourier-Bessel transform of $V_s(q)$, which gives:

$$V_s(r) = \int_0^\infty qV_s(q)J_0(qr)dq \tag{8.104}$$

The quasi-2D Coulomb potential, $V_s(r)$ of Eq. (8.104), can be then employed to obtain binding energy of exciton in QW and MQW, and seems the most appropriate in use for this applications. Nevertheless we left to the user an option, to use in his simulation exciton models using 3D and 2D Coulomb potentials as defined in Eqs. (8.97), for comparative studies.

8.7.3 Exciton Binding Energy in Quantum Well

In our treatment, the confining potential along z-direction, along with wave-functions and corresponding energy values QW sub-bands are based on self-consistent solution of a Schrödinger and Poisson Equations. The solution is later applied to calculate binding energy of exciton via the form-factor, and finally absorption spectrum of excitons in MQW or QW can be calculated. At this time we limited our derivation to the ground, 1s, excitonic states, but this can be easily extended into excited exciton state as well as to excitons at higher QW levels, in the future.

Consider a wave function for a ground state exciton in a QW as:

$$\Phi(z_e, z_h, r) = \phi(z_e)\phi(z_h)\phi_{exc}(r), \tag{8.105}$$

where $\phi_{exc}(r)$ is a chosen trial wave function for a 1s exciton in QW to be:

$$\phi_{exc}(r) = (\frac{2}{\pi})^{1/2}\frac{1}{\lambda_{exc}}\exp(-r/\lambda_{exc}). \tag{8.106}$$

Here λ_{exc} represents an in-plane exciton Bohr radius, while $\phi(z_e)$,$\phi(z_h)$ are ground state electron and heavy or light hole envelope wave functions. Coulomb attraction of e-h pairs is assumed to be week, compare to the effect of QW confinement in z-direction. As a result, electron and hole wave functions are assumed to be unaffected by long range Coulomb interaction, which only modifies their energy values(true in the limit of 2D exciton Bohr radius, larger than the size o the z-confinement).

We choose to use Ritz variational method for binding energy estimate. The method relies on minimization of the exciton energy upon the trial function parameter, λ_{exc}:

$$E_{exc}(\lambda_{exc}) = E_g + E_{e0} + E_{ho} - E_{b,exc}, \tag{8.107}$$

which is equivalent to finding a maximum in exciton binding energy,E_{bexc}:

$$E_{b,exc}(\lambda_{exc}) = \frac{<\Phi|\hat{\mathbf{H}}|\Phi>}{<\Phi|\Phi>}, \tag{8.108}$$

where

$$\hat{H} = -\frac{\hbar^2}{2m_r}(\bigtriangledown_r^2 + \bigtriangledown_z^2) + V_{QW}(z) + V_s(r), \tag{8.109}$$

and after partial integration:

$$E_{b,exc}(\lambda_{exc}) = \frac{\hbar^2}{2m_r\lambda_{exc}^2} + < \phi_{exc}(r)|V_s(r)|\phi_{exc}(r) > \tag{8.110}$$

with respect to the 2D Bohr radius λ_{exc} parameter of a trial function $\phi_{exc}(r)$ given by Eq.(8.106).

8.7.4 Quantum Well Exciton Absorption

A general absorption coefficient formula can be applied to the exciton transition as well, as given by:

$$\alpha(\hbar\omega) = A_0 \sum_n |\varphi_n(0)|^2 \delta(E_n + E_g - \hbar\omega)n, \tag{8.111}$$

where

$$A_0 = \frac{\pi e^2 |\hat{\mathbf{e}} \cdot \mathbf{p}_{cv}|^2}{n_r c\epsilon_0 \omega m_0^2} \tag{8.112}$$

$\mathbf{p}_{cv}$ is an interband transition matrix element in QW, n_r is a refractive index, c-speed of light,$\phi_n(r)$ is a normalized wave function of exciton. Here, summation over n extends to both bound and continuum state excitons. For QW bound exciton states we have the oscillator strength given by :

$$\phi_n(0) = \frac{1}{\pi a_0^2 (n - \frac{1}{2})^3}, \tag{8.113}$$

and

$$E_n = \frac{R_y}{(n - \frac{1}{2})^2} \tag{8.114}$$

Initially, it is sufficient to include only ground state heavy and light hole excitons (n=1), although in general case the bound exciton absorption coefficient becomes:

$$\alpha_B(\hbar\omega) = A_0 \sum_{n=1}^{\infty} \frac{2}{R_y a_0^2 \pi (n - \frac{1}{2})^3} \times \delta[\epsilon + \frac{1}{(n - \frac{1}{2})^2}], \tag{8.115}$$

where $\epsilon = (E - E_g)/R_y$.

The continuum-state exciton contribution to absorption is:

$$\alpha_C(\hbar\omega) = A_0 \int_0^{\infty} dE \frac{S_{2D}(E)}{2\pi R_y a_0^2} \delta(E + E_G - \hbar\omega), \tag{8.116}$$

where

$$S_{2D}(\epsilon) = \frac{2}{1 + \exp(-2\pi/\epsilon)} \tag{8.117}$$

is known as Sommerfeld enhancement factor.

The total absorption is equal $\alpha_B(\hbar\omega) + \alpha_C(\hbar\omega)$:

$$\alpha_{exc}(\hbar\omega) = \frac{A_0}{2\pi R_y a_0^2}[4 \times \sum_{n=1}^{\infty}\{\frac{1}{(n-\frac{1}{2})^3}\delta(\epsilon + \frac{1}{(n-\frac{1}{2})^2})] + S_{2D}(\epsilon)\} \tag{8.118}$$

Finally, we include the finite line-width effect by introducing the Lorentzian broadening function to obtain

$$\alpha_{exc}(\hbar\omega) = \frac{A_0}{2\pi R_y a_0^2}\left[4 \times \sum_{n=1}^{\infty}\frac{1}{\pi(n-\frac{1}{2})^3}\frac{\gamma}{\left(\epsilon + (n-\frac{1}{2})^{-2}\right)^2 + \gamma^2} + \int_0^{\infty}\frac{d\epsilon'}{\pi}\frac{\gamma S_{2D}(\epsilon')}{(\epsilon'-\epsilon)^2+\gamma^2}\right] \tag{8.119}$$

8.8 Franz - Keldysh Effect

The Franz-Keldysh Effect (*FKE*) is a widely known phenomenon observed in the inter-band absorption spectrum of bulk semiconductors. It is also known as photo-assisted Zener tunneling. It only occurs in the presence of an external DC electrostatic field **F** and is related to the bulk *DC-Stark Effect*. It is similar to the *Quantum-Confined Stark Effect* (*QCSE*) observed in semiconductor Quantum Wells (*QW*).

FKE and *QCSE* both have their uses in modern electroabsorption modulators.

8.8.1 Solution of the Schrödinger Equation

Consider the case of a single non-interacting particle charge, q in a uniform external dc-electrostatic field acting along z-direction, $\mathbf{F} = [0, 0, F]$. It can be described by the *Schrödinger Equation* :

$$\left(\frac{-\hbar^2}{2m_r}\nabla^2 + q\mathbf{F}\cdot\mathbf{r}\right)\Phi(\mathbf{r}) = \mathrm{E}\Phi(\mathbf{r}), \tag{8.120}$$

where $m_r = m_e * m_h/(m_e + m_h)$. is the reduced effective mass. One can search the solution of Eq.(8.120) in the form of:

$$\Phi(\mathbf{r}) = \frac{e^{ik_x + ik_y}}{\sqrt{A}}\Phi_{E_z}(z), \tag{8.121}$$

where the z-dependent part of the wave function $\Phi(z)$ obeys:

$$\left(\frac{-\hbar^2}{2m_r}\frac{d^2}{dz^2} + qFz\right)\Phi(z) = \mathrm{E}_z\Phi(z), \tag{8.122}$$

with a total energy relative to the bottom of the band given by:

$$E = \frac{-\hbar^2}{2m_r}(k_x^2 + k_y^2) + E_z. \tag{8.123}$$

The solution to the Eq. (8.122) can be expressed in terms of Airy Functions $Ai(Z)$, satisfying the proper normalization and boundary condition of exponential decay in the limit $z \to +\infty$, where variable:

$$Z = \sqrt[3]{\frac{2m_r qF}{\hbar^2}} \times \left(z - \frac{E_z}{qF}\right) \tag{8.124}$$

Electron and hole states in an external electrostatic field $\mathbf{F} = \hat{z}F$ can be characterized by the the wave function for a given set of quantum numbers (kx, ky, Ez) satisfying Eq. (8.121).

8.8.2 Absorption Spectrum

The inter-band absorption calculation can be obtained from a sum over all electron-hole states:

$$\alpha(\hbar\omega) = A_0 \sum_n |\varphi_n(0)|^2 \delta(E_n + E_g - \hbar\omega), \tag{8.125}$$

where

$$A_0 = \frac{\pi q^2 |\hat{\mathbf{e}} \cdot \mathbf{p}_{cv}|^2}{n_r c \epsilon_0 \omega m_0^2} \tag{8.126}$$

Substituting the sum over all quantum number states by the integration over E_z continuous spectrum in Eq. (8.121), one can obtain :

$$\alpha(\hbar\omega) = A_0 \times 2 \sum_{k_x,k_y} \int_0^\infty dE_z |\varphi(0)|^2 \delta\left[\frac{\hbar^2}{2m_r}(k_x^2 + k_y^2) + E_z + E_g - \hbar\omega\right], \tag{8.127}$$

For typical III-V compound semiconductor zincblende structure we can substitute into Eq. (8.127) $E_t = \frac{\hbar^2}{2m_r}[k_x^2 + k_y^2]$ and use quasi-continuous integration over dE_t instead of sum

$$\frac{2}{A}\sum_{k_x,k_y} = 2\int \frac{d^2\mathbf{k}_t}{(2\pi)^2} = \frac{m_r}{\pi\hbar^2}\int dE_t \tag{8.128}$$

One can obtain absorption coefficient from Eq. (8.127) as follows:

$$\alpha(\hbar\omega) = \frac{A_0}{2\pi}\frac{m_r}{\pi\hbar^2}\sqrt{\hbar\theta_F}\int_\eta^\infty d\eta Ai^2 = \frac{A_0}{2\pi}\frac{m_r}{\pi\hbar^2}\sqrt{\hbar\theta_F}[-\eta Ai^2(\eta) + Ai'^2(\eta)], \tag{8.129}$$

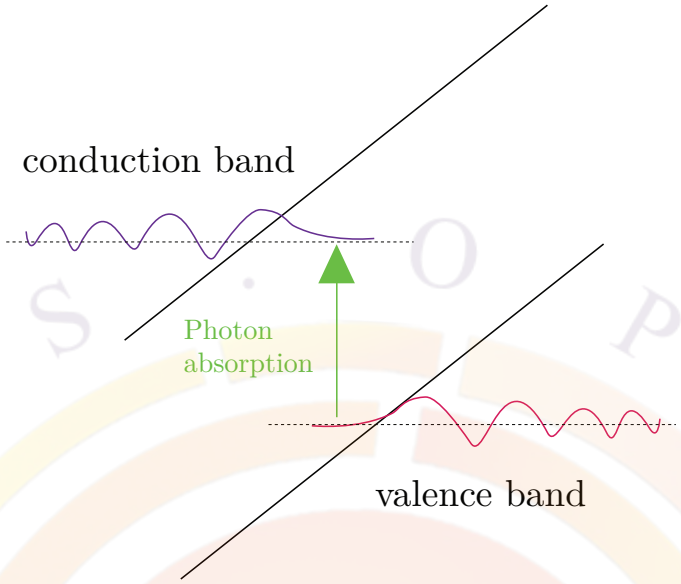


Figure 8.9: Concept of Franz-Keldysh effect of photo-assisted absorption in bulk semiconductors in external electrostatic field. The electron and hole wave-functions become Airy functions, which in external electrostatic field "tunnel" into the band-gap region allowing for overlap of electron and hole wave-functions and absorption of photons with energies below the band-gap energy.

where $Ai'(\eta)$ is the derivative of $Ai(\eta)$ over η, $\hbar\theta_F = \sqrt[3]{\frac{\hbar^2 q^2 F^2}{2m_r}}$, and $\eta = \frac{E_g - \hbar\omega}{\hbar\theta_F}$

Fig. 8.9 illustrates the net result of applying an electrostatic field on inter-band absorption spectra. The apparent oscillations above the band-gap energy and enhanced exponential decay in below-bandgap energy region are signature of the Franz-Keldysh effect. Fig. 8.10 compares the absorption spectrum with and without the external field as a result of the Franz-Keldysh effect.

8.9 Interband Optical Transition Model for Quantum Dots

This section formulates the optical gain and spontaneous emission spectra for a quantum dot device. We consider an ensemble of quantum dots (QDOT) within a multiple quantum well system. For a self-assembled quantum dots device [44], the dots are closely stacked up on top of each other and there may be strong quantum mechanical coupling between the them. In the following formulation, we assume that the optical spectrum theory for quantum well has been well established [33] and we shall focus on quantum dots.

The formulas here are similar to those in [45]. In a quantum complex containing

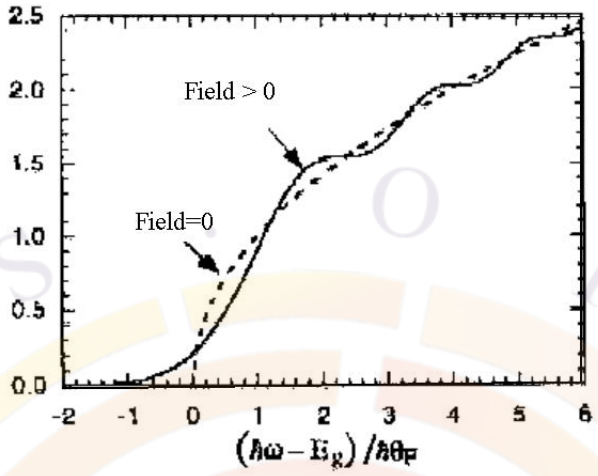


Figure 8.10: The associated absorption spectrum for finite field $F \neq 0$ (solid curve). The dashed line is the free electron and hole absorption spectrum without applied electrostatic field.

both wells and dots, the total 2D area; carrier concentrations can be expressed as

$$n_{2d} = n_{qw} + n_{qd} \tag{8.130}$$

The QDOT electron concentration can be written as

$$n_{qd} = \sum_{i,s} 2N_{qd}(s) f_c(E_{i,s}) \tag{8.131}$$

where the sum i is over all energy levels and s is over quantum dots of difference sizes. Please note the factor of two taking into account spin degeneracy. $N_{qd}(s)$ is the areal density of quantum dots with size s. The Fermi function is given by

$$f_c(E_{i,s}) = \frac{1}{1 + exp[(E_{i,s} - E_{fn})/kT]} \tag{8.132}$$

In general, analytical expression for energy level $E_{i,s}$ is not available and one has to solve a 3D numerical eigenvalue problem to find the confined energies and associated wave functions ψ_{is} for a 3D dot embedded within a well [45]. The APSYS software includes numerical solutions of quantum states for a general 3D quantum dot structure.

Having solved the quantum states of the 3D dot/well complex, we proceed to formulate the optical gain spectrum as follows:

$$g_{qd}(E) = \sum_s \sum_{i,j} \frac{\pi q^2 \hbar M_b^2 N_{dq}}{E \varepsilon_0 c_0 m_0^2 n_r t_{cmplx}} | < \psi_{is} | \psi_{js} > |^2 g_{cp} G_s(E - E_{ijs})(f_c - f_v) \tag{8.133}$$

where g_{cp} is a multiple dot coupling factor which is used avoid double counting. It is inversely proportional to the number of coupling dots. For example, if we have two dots coupling to each other, the number of states will be doubled and we need to divide by two to avoid double counting. M_b is the bulk matrix element and t_{cmplx} is the thickness of the well/dot complex. The sum is over all possible confined electron and hole states and over all dot sizes. The last factor is the population inversion factor determined by quasi-Fermi levels of electrons and holes. Other symbols have their usual meanings. The function G_s is a Gaussian broadening function representing intraband and inhomogeneous broadening defined as follows:

$$\begin{aligned} \Delta &= \hbar/\tau \\ G_s(E) &= \frac{1}{\Delta\sqrt{2\pi}} exp\left[-\frac{(E-E_{ijs})^2}{2\Delta^2}\right] \end{aligned} \tag{8.134}$$

The material gain of the active well/dot complex contains contribution from the quantum well, i.e., from states above the quantum dot. Depending on the detailed structure and film quality, the states of the quantum wells may not be perfectly extended through the while well. One may regard some of the states from the quantum wells or the wetting layers as being truncated by the quantum dots. Thus it is convenient to introduce a factor to control or describe the contribution from the wells as follows

$$g_{2d} = r_{qw} g_{qw} + g_{qd} \tag{8.135}$$

where g_{qw} and g_{qd} are optical gain contribution from quantum wells and dots, respectively. r_{qw} is a factor between zero and unity related to the areal ratio of the well area over that of the dots. The above optical gain formulas are important for application involving laser diodes (LD) and vertical cavity surface emitting lasers (VCSEL).

Having found the formula for the optical gain, we can express the spontaneous emission rate using the relation between optical gain and spontaneous emission [2]. The spontaneous emission is given as follows:

$$r_{qd}^{sp}(E) = \frac{q^2 n_r E M_b^2}{\pi\varepsilon_0 m_0^2 \hbar^2 c_0^3} |<\psi_{is}|\psi_{js}>|^2 g_{cp} G_s(E-E_{ijs}) N_{qd}(s)/t_{cmplx} f_c(1-f_v) \tag{8.136}$$

The total spontaneous emission rate should also include contribution from the quantum wells in analogy with the optical gain:

$$r_{2d}^{sp}(E) = r_{qw} r_{qw}^{sp}(E) + r_{qd}^{sp}(E) \tag{8.137}$$

Integration of the above over all photon energies results in the total carrier radiative recombination rate as a dominant recombination mechanism for many devices based on forward biased p-n junctions:

$$R_{sp} = \int r_{2d}^{sp}(E) dE \tag{8.138}$$

The spontaneous emission model is important for applications involving light emitting diode (LED), LD and VCSEL. For LED, the EL spectrum comes from the spontaneous emission rate in Eq. 8.137 directly while radiative recombination rate in Eq. 8.138 determines the internal quantum efficiency (IQE). For LD and VCSEL, the radiative recombination rate is often the dominant leakage mechanism upon which the threshold behavior depends.

The software also offers a model for a quantum well to include a quantum dot-like density of states (DOS). This may be used to account for high brightness LED from InGaN-based MQW where a pure quantum well model can not explain the high brightness which is thought to come from localized, dot-like states associated with indium segregation.

The QDOT model in the software usually requires a separate calculation of 3D quantum states with given shapes such as disk or sphere. The QDOT-like model skips such details and makes it easier to manage: the model uses a DOS reduction factor to represent 3D quantum confinements which result in lower DOS than pure quantum well.

The DOS reduction factor is defined as the ratio of total number of quantum states in a dot over that in the well. Such a factor many be estimated if shapes of the dots are known. For a simple case, one may use states in a quantum box against those in a quantum well.

To make analogy with reduced DOS in quantum well against those in the bulk, the actual reduction ratio can be estimated if the potential profile of the well is known. Polarization charge and self-consistent calculation may be enabled in such QDOT-like states.

Chapter 9

IMPACT IONIZATION AND TUNNELING

This chapter describes our model of impact ionization and quantum tunneling. We will show that there is similarity between interband tunneling and impact ionization. Both result in generation of electron and hole pairs from a reverse biased p-n junction. Intraband tunneling involves only one type of carrier and the main issue is how to reconcile the quantum nature of wave mechanics with the classical behavior of drift-diffusion.

9.1 Impact Ionization Model

9.1.1 Introduction

Impact ionization is defined through the following expression for generation rate (see Ref. [1]):

$$G = \alpha_n n v_n + \alpha_p p v_p. \tag{9.1}$$

where α_n is the electron ionization rate defined as the number of electron-hole pairs generated by an electron per unit distance traveled; α_p is similarly defined for holes. Both α_n and α_p are strongly dependent on the electric field.

Equation (9.1) is somewhat difficult to implement in a drift-diffusion model because the model is not directly concerned with the velocity. Since impact ionization occurs only in high field region where the drift terms dominates, Selberherr [46] suggested the use of J/q in place of nv for ease of implementation:

$$G = \alpha_n J_n/q + \alpha_p J_p/q \tag{9.2}$$

The issue remains how to choose values of α_n and α_p. Crosslight proposes different models commonly found in the literature but it is expected that the user will choose

the most appropriate model for a given situation and provide calibrated parameters for that model. Some sample values for these parameters are provided below and may help in the calibration process. The default model settings for the commands which implement these models usually correspond to data for silicon.

9.1.2 Chynoweth model

A commonly used expression for α_n was first proposed by Chynoweth [47]:

$$\alpha = \alpha_n^{\infty} e^{-\frac{F_{cn}}{F}} \tag{9.3}$$

where F_cn is the critical field and α_n^{∞} is the inverse of the mean free path between impacts when $F = \infty$. A similar expression can be written for the holes.

This expression was later generalized by Selberherr [46]:

$$\alpha = \alpha_n^{\infty} e^{-\left(\frac{F_{cn}}{F}\right)^{\kappa_n}} \tag{9.4}$$

Another improvement to the model was also added in the form of a temperature dependence for both α_n^{∞} and F_{cn}. This is expressed in the form a scaling factor γ:

$$\gamma = \frac{tanh\left(\frac{h\omega_{op}}{2kT_0}\right)}{tanh\left(\frac{h\omega_{op}}{2kT}\right)} \tag{9.5}$$

where $h\omega_{op}$ is the energy of the optical phonon.

The parameters above are often fitted for a specific range of field values: multiple ranges are often necessary to cover the entire region of interest in a simulation.

9.1.3 Baraff model

Another theory of impact ionization is Baraff's three-parameter theory which takes into account the temperature dependence. A convenient formula for Baraff's theory was given by Crowell and Sze [48]:

$$\alpha\lambda = e^{g(r,x)} \tag{9.6}$$

$$\begin{aligned} g(r,x) &= (11.5r^2 - 1.17r + 3.9 \times 10^{-4})x^2 \\ &+ (46r^2 - 11.9r + 1.75 \times 10^{-2})x \\ &+ (-757r^2 + 75.5r - 1.92) \end{aligned} \tag{9.7}$$

(9.8)

where

$$r =< E_p > /E_I \tag{9.9}$$

and

$$x = E_I/qF\lambda \tag{9.10}$$

In the above, $< E_p >$ is the optical phonon energy and λ is the optical phonon scattering mean free path. E_I is the ionization energy for which the following formula was suggested [48]:

$$E_I = 3E_g/2 \tag{9.11}$$

The temperature dependence of $< E_p >$ and λ is given by

$$< E_p >= E_p tanh\left(\frac{E_p}{2kT}\right) \tag{9.12}$$

$$\lambda = \lambda_0 tanh\left(\frac{E_p}{2kT}\right) \tag{9.13}$$

which follows the same behavior as the temperature scaling in the Chynoweth model.

9.1.4 Lackner model

The Lackner model[49] is a modification of the basic Chynoweth model defined above. It introduces a correction term $1/Z$ where

$$Z = 1 + \frac{F_{cn}}{F}e^{-\frac{F_{cn}}{F}} + \frac{F_{cp}}{F}e^{-\frac{F_{cp}}{F}} \tag{9.14}$$

As in the Chynoweth model, the temperature dependance of these terms is included via the γ scaling factor.

9.1.5 Okuto-Crowell model

This model[50] takes into account band bending effects and temperature dependence through the following empirical model:

$$\alpha = a(T)F^x e^{-\left(\frac{F_{cn}(T)}{F}\right)^y} \tag{9.15}$$

where both $a(T)$ and $F_{cn}(T)$ have a linear dependence on temperature. The field exponential terms are usually set as $x = 1$ and $y = 2$.

9.1.6 Dopant-dependant model

This model implements the basic Chynoweth model defined above for a single field range. It also implements some empirical relations for the dependence of the critical field on the doping concentration.

Baliga[51] describes this relationship as:

$$F_{cn} = F_0(N_D)^x \tag{9.16}$$

where N_D is the donor concentration (in cm^{-3}) and $x = \frac{1}{8}$ is typically used.

Sze[1] proposes a different relationship:

$$F_{cn} = \frac{F_0}{1 - \frac{1}{3} log_{10}\left(\frac{N_D}{10^{16}}\right)} \tag{9.17}$$

where N_D is the donor concentration in cm^{-3}.

In both of these models, only the effects of donor concentration are considered at this time. The critical field for holes is set to a constant value.

9.1.7 Mean free path model

This is an extremely simplified phenomenological version of the Chynoweth model. Instead of defining the critical field as a fitting parameter, we define the mean free path and calculate the critical field assuming the ionization energy is equal to the bandgap:

$$F_{cn} = \frac{E_g}{\lambda_0} \tag{9.18}$$

9.1.8 Some Useful Parameters for Impact Ionization Models

Semiconductors used for microelectronics

For Baraff's model, the following parameters are taken from Sze [1]:

For the Chynoweth model, many possible fits are possible. Overstraeten and de Man[52] produced one set of data while Grant[53] proposed another. Their results are summarized in Tables 9.2 and 9.3.

It is clear that the results of the simulation will vary greatly depending on the parameters that are used. For the sake of comparison, all three of the above models for silicon are plotted in Fig. 9.1.

Material	E_p electrons (eV)	λ_0 electrons (Å)	E_p holes (eV)	λ_0 holes (Å)
Si	0.063	76	0.063	55
Ge	0.037	105	0.037	105
GaAs	0.035	58	0.035	58

Table 9.1: Commonly used impact ionization parameters for Baraff's model

Electrons		
$\alpha_n^\infty(\times 10^8 m^{-1})$	$F_{cn}(\times 10^8 V/m)$	Field range $(\times 10^8 V/m)$
0.703	1.231	$0.175 < F < 0.6$
Holes		
$\alpha_p^\infty(\times 10^8 m^{-1})$	$F_{cp}(\times 10^8 V/m)$	Field range $(\times 10^8 V/m)$
1.582	2.03	$0.175 < F < 0.4$
0.671	1.69	$F > 0.4$

Table 9.2: Overstraeten and de Man's fit to the Chynoweth model for silicon

Semiconductors used for optoelectronics

The impact ionization coefficients for compound semiconductors used in optoelectronics are less well known than those of silicon. When fitting the experimental data published in Ref. [1] to the basic Chynoweth model, we obtain the results in Table 9.4.

For InGaAsP material lattice matched to InP, we can use linear interpolation of InP and In(0.53)Ga(0.47)As.

9.2 Intraband Quantum Tunneling

9.2.1 Introduction

In highly doped heterojunction or Schottky contacts, an important current transport mechanism is the quantum tunneling effect. In highly doped Schottky contacts and heterojunctions, the quantum barriers are so thin that significant amount of tunneling current goes through the barriers. It is also the key transport mechanism in a realistic Ohmic contact which is essentially a highly doped Schottky contact.

We derived the formulas for tunneling current at the top of the barrier in subsection 9.2.2. The formulas are generalized to an arbitrary point in subsection 9.2.3. The tunneling transparencies for a few important cases are given in subsection 9.2.4.

Electrons		
$\alpha_n^\infty(\times 10^8 m^{-1})$	$F_{cn}(\times 10^8 V/m)$	Field range $(\times 10^8 V/m)$
2.6	1.43	$F < 0.24$
0.62	1.08	$0.24 < F < 0.53$
0.5	0.99	$F > 0.53$
Holes		
$\alpha_p^\infty(\times 10^8 m^{-1})$	$F_{cp}(\times 10^8 V/m)$	Field range $(\times 10^8 V/m)$
2.0	1.97	$0.2 < F < 0.53$
0.56	1.32	$F > 0.53$

Table 9.3: Grant's fit to the Chynoweth model for silicon

Material	$\alpha_n^\infty(\times 10^8 m^{-1})$	$F_{cn}(\times 10^8 V/m)$	$\alpha_p^\infty(\times 10^8 m^{-1})$	$F_{cp}(\times 10^8 V/m)$
GaP	7.38	2.02	7.38	2.02
InP	8.36	1.55	1.57	0.994
In(0.53)Ga(0.47)As	34.4	0.833	125	1.02
GaAs(0.88)Sb(0.12)	0.0997	0.25	0.092	0.316

Table 9.4: Impact ionization rates for some compound semiconductors

Finally, we discuss the numerical treatment of quantum tunneling in drift-diffusion model in subsection 9.2.5.

We use the standard WKB theory to give formulas for the transparency in subsection 9.2.3. Discussion of implementation of the tunneling current is given in subsection 9.2.4.

9.2.2 Tunneling current at top of barrier

Analytical formulas incorporating tunneling effects at the top of the potential barrier were derived by Grinberg et. al. [54] whose approach we shall follow here. Similar to [54] we assume that Boltzmann statistics can be used because the quasi-Fermi level is usually much lower than the top of the barrier. We use the Schematic in Fig. 9.3 and consider the drift current on top of the barrier X_m

$$J = qvn_m + qvn_m\alpha_T \tag{9.19}$$

where α_T is the tunneling coefficient to be derived. The velocity v is dependent on mobility or the thermionic emission properties at the top of the barrier.

Here the basic understanding is that the quasi-Fermi level is relatively flat around the barrier such that the energy distribution of the carriers can be expressed by a

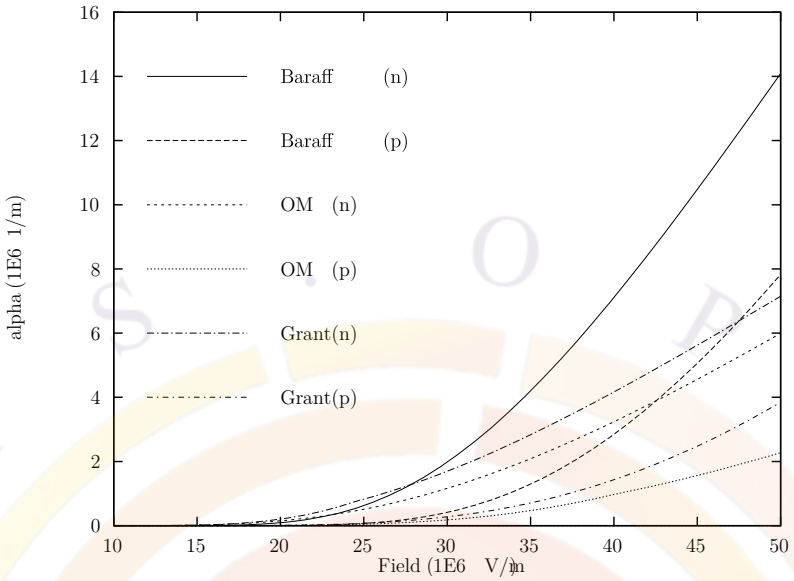


Figure 9.1: Electron and hole impact ionization coefficients of silicon for different models. Baraff is for Baraff's three-parameter theory; OM is for the parameters of Chynoweth's formula from the work of Overstraeten and de Man; Grant is for parameters of Chynoweth's formula from the work of Grant. n and p are used to denote parameters for electrons and holes, respectively.

simple Boltzmann function (or exponential function). We assume that the carriers with energy between U_0 and U_m are capable of tunneling through the barrier and appears at the other side of the it, giving rise to the additional term in Eq. (9.19). For carriers with energy greater than U_m, they are accounted for by the standard drift-diffusion model.

We define an energy dependent carrier distribution n_E such that

$$n = \int n_E dE \tag{9.20}$$

Using Boltzmann distribution

$$n_E = g(E) exp\left(\frac{U_m - E}{kT}\right) \tag{9.21}$$

where $g(E)$ is the density of states which is a much slower function of energy than the exponential function. For numerical convenience, we ignore the slower energy dependence and replace it by an average constant n_{Em} so that Eqn. 9.20 still holds. As we shall see later, our final results do not depend on the choice of constant n_{Em}.

As we are able to express the energy distribution as follows

$$n_E = n_{Em} exp\left(\frac{U_m - E}{kT}\right) \tag{9.22}$$

the tunneling current can be expressed as

$$J_{tun} = \int_{U_0}^{U_m} qvn_E D_T(E)dE \tag{9.23}$$

$$= qvn_{Em}\int_{U_0}^{U_m} exp\left(\frac{U_m - E}{kT}\right) D_T(E)dE \tag{9.24}$$

where $D_T(E)$ is the energy dependent tunneling transparency. We note that the carrier density at the top of the potential barrier can be expressed in term of n_{Em}:

$$n_m = \int_{U_m}^{\infty} n_{Em} exp\left(\frac{U_m - E}{kT}\right) dE \tag{9.25}$$

$$= n_{Em}kT \tag{9.26}$$

Therefore the tunneling current can be written as

$$J_{tun} = qvn_m(kT)^{-1}\int_{U_0}^{U_m} exp\left(\frac{U_m - E}{kT}\right) D_T(E)dE \tag{9.27}$$

The tunneling coefficient α_{Tm} at the top of the barrier is given by

$$\alpha_{Tm} = (kT)^{-1}\int_{E_0}^{U_m} exp\left(\frac{U_m - E}{kT}\right) D_T(E)dE \tag{9.28}$$

9.2.3 Tunneling current at an arbitrary point

In a 2D simulation, all physical quantities vary in space. We have to assume that the tunneling current also varies in space to make the model compatible with the 2D drift-diffusion model. In another word we have to decide what the tunneling effect is on the current at an arbitrary point away from the top of the barrier. It would be wrong to assume that the same tunneling current appears everywhere in the whole device, although this is the simple case for many textbook tunneling examples.

We must point out that the treatment of tunneling mechanism we give here is of semi-classical nature. The reason is that quantum tunneling assumes the carrier is described by a wave function and can not be localized accurately, while the drift-diffusion model assumes that the carriers are localized and have a definite spatial distribution. Therefore, we should try to satisfy both requirements by choosing a proper area for the tunneling model. If the area of consideration is too small, it conflicts with the wave nature of the problem. If the area is too large, the model contradicts with particle nature of drift-diffusion model. It is clear that the transport mechanism at a point far from the quantum barrier is unrelated to the quantum tunneling.

Again we assume that only those carriers with energy between U_0 and U_m contribute to tunneling. We wish to relate the tunneling current at an arbitrary point to the local current density.

We give the distribution function at an arbitrary point x,

$$n_E(x) = n_{Ex} exp\left(\frac{U(x)-E}{kT}\right) \tag{9.29}$$

The tunneling current can be expressed by

$$\begin{aligned} J_{tun} &= \int_{U_0}^{U_m} qvn_E(x) D_T(E) dE & (9.30)\\ &= qvn_{Ex} \int_{U_0}^{U_m} exp\left(\frac{U(x)-E}{kT}\right) D_T(E) dE & (9.31) \end{aligned}$$

Using the same derivation as in the previous subsection, we re-write the current as

$$\begin{aligned} J_{tun} &= qvn_{Ex} \int_{U_0}^{U_m} exp\left(\frac{U(x)-E}{kT}\right) D_T(E) dE & (9.32)\\ &= qvn(x)(kT)^{-1} \int_{U_0}^{U_m} exp\left(\frac{U(x)-E}{kT}\right) D_T(E) dE & (9.33)\\ &= qvn(x)(kT)^{-1} exp\left(\frac{U(x)-U_m}{kT}\right) \int_{U_0}^{U_m} exp\left(\frac{U_m-E}{kT}\right) D_T(E) dE & (9.34)\\ &= qvn(x) exp\left(\frac{U(x)-U_m}{kT}\right) \alpha_{Tm} & (9.35) \end{aligned}$$

The above formula suggest that all we need to do is to multiply the tunneling coefficient at the top of the barrier by a Boltzmann exponential factor $exp\left(\frac{U_m-E}{kT}\right)$. This is consistent with our common sense that in regions farther away from the barrier, the tunneling effect is less pronounced.

9.2.4 Tunneling transparency

In the current version of the our simulation program, we have included the following models for the tunneling transparency.

- 1. Transparency for a rectangular barrier. Following Ref. [55], for a rectangular barrier of barrier height U_m and thickness d_0, we have the following exact solution for the tunneling transparency:

$$D(E) = \left(1 + \frac{sinh^2\left(\sqrt{\frac{2m^*(U_m-E)}{\hbar^2}}a\right)}{4(E/U_m)(1-E/U_m)}\right)^{-1} \tag{9.36}$$

- 2. Abrupt heterojunction barrier. One side of the potential barrier has a vertical wall and the other side has a profile described by a quadratic formula [56]. The formula was worked out and is given as follows [54] [56] :

$$D(E) = exp\left(-\frac{U_m}{E_{00}}\left[\sqrt{1-R_x}+\frac{1}{2}R_x ln(R_x) - R_x ln(1+\sqrt{1-R_x})\right]\right) \tag{9.37}$$

$$E_{00} = \frac{\hbar q}{2}\sqrt{\frac{|\ N_d - N_a\ |}{m^*\varepsilon}} \tag{9.38}$$

$$R_x = E/U_m \tag{9.39}$$

where N_a and N_d are the doping concentrations for acceptors and donors, respectively. U_m and E are measured from the lowest point of the potential profile on the side with a smooth profile.

- 3. Arbitrary barrier with smooth potential profile on both sides. The formula derived from WKB theory can be found in Ref. [57].

$$D(E) = exp\left(-\frac{2}{\hbar}\int_a^b \sqrt{2m^*[U(x)-E]}dx\right) \tag{9.40}$$

Please note that the requirement for the above formula is that the potential profile must be smooth. Significant error may arise if this requirement is not satisfied. For example, we compare the exact solution with the WKB theory in Fig. 9.2 for a rectangular barrier with abrupt vertical wall on both sides. We find that substantial error can result for such a case.

- 4. Propagation matrix method may be used to treat arbitrary barrier. This method cuts up the potential barrier into piece-wise constant and find the analytical solution in the form of propagation matrices. This method is rather accurate but the computation time required is relatively long.

9.2.5 Numerical evaluation of tunneling current

In a 2D simulation of semiconductor, the evaluation of the tunneling current is not simple since the formulas we developed are all one-dimensional. We suggest the following steps for treating the tunneling current:

- 1)Identify a significant potential barrier. This usually occurs in Schottky contacts, heterojunctions and between quantum wells. The user should mark down a rectangular area with which the tunneling current distribution is to be calculated.

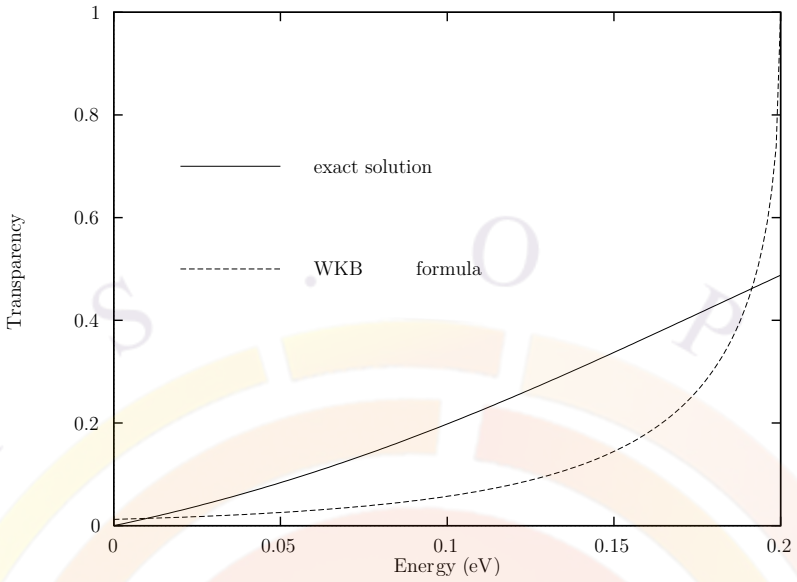


Figure 9.2: Comparison of the transparency from the WKB with the exact solution for a rectangular potential barrier. The barrier thickness is 20 Åand the barrier height is 0.2 eV. The relative effective mass of the carrier is assumed to be 0.2.

- 2)Select a one dimensional segment for the calculation. Since the potential distribution is 2D, selection of the 1D segment reduces the amount of calculation involved.
- 3)A current enhancement factor $1 + \alpha_T$ is calculated along the selected 1D segment.
- 4)Multiply current flow component along the 1D segment with $1 + \alpha_T$ distribution within the rectangular area. For a current flux between two nodes in an arbitrary direction with an angle θ_T from the 1D segment, we multiply $1 + \alpha_T cos(\theta_T)$.

9.2.6 Coupled MQW and intraband tunneling

We wish to discuss the relation between coupled MQW and intraband tunneling. The two models are closely related to each other. They both come from the solution of the same quantum mechanical wave equation.

The quantum levels of a coupled MQW are solutions of confined states of the wave equation whereas intraband tunneling deals with solution of unconfined states. The carriers tunnel through the barriers of an MQW system via standing waves of confined states while traveling waves are used to describe the intraband tunneling effects in the above subsections.

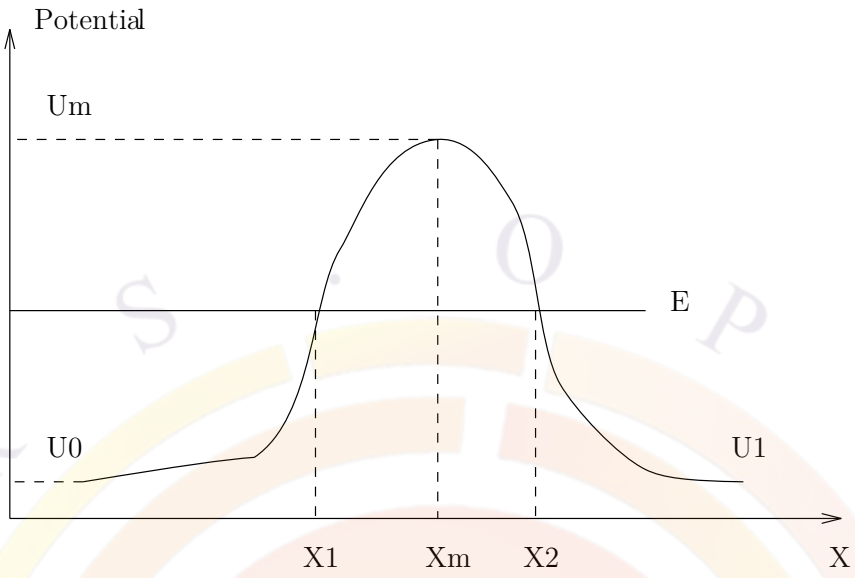


Figure 9.3: Schematic for the potential barrier

In the tunneling model, the carrier states are assumed to be unconfined on both sides of the barrier. One must not use both models to describe the same quantum state. For example, if confined states are already used to described a coupled MQW system with standing waves distribution between the wells, one must not apply the above tunneling model to treat carrier transport between the wells within the same region.

Similarly in the case of quantum-MOS, the quantum states in the triangular well under the gate have already been treated as confined (or partly confined if quantum wave leakage to gate is considered) states. Since the penetration of the wave function from the confined states into the oxide/gate has already been considered, there is no need to apply tunneling model to these confined states. Alternatively, if direct tunneling method is used to treat the current flow through the barrier between confined states in the triangular well and the unconfined states at the gate of an MOS, the the carriers penetrating the oxide must be removed from the transport equations to avoid double counting the tunneling current.

9.2.7 High-resistance heterojunction and quantum tunneling

We consider a situation of current injection in the classical drift-diffusion model when high doping is used around an abrupt heterojunction. The peak of the potential barrier depletes the mobile carriers while ionized dopants generates a high internal electric field layer. The high internal field creates a steep potential profile which

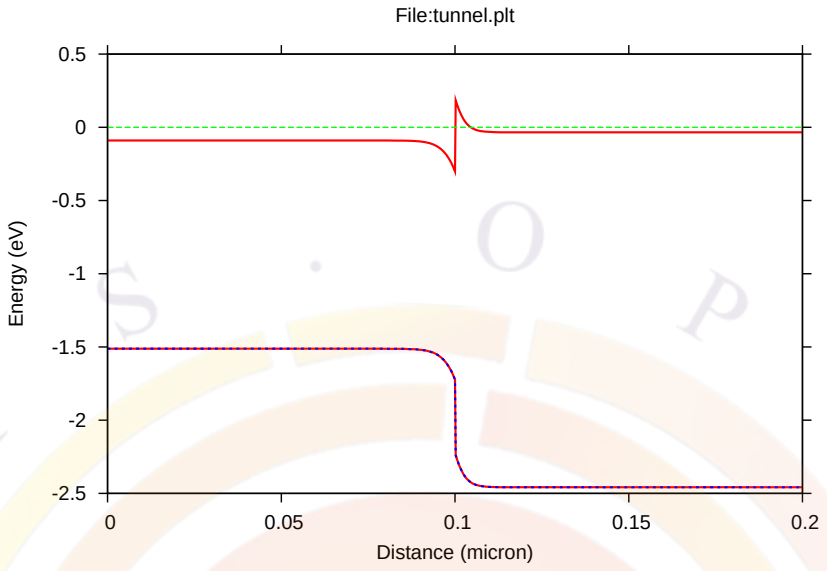


Figure 9.4: Band diagram of a highly doped GaAs/InGaAlP junction in equilibrium.

pushes the barrier higher. Since all mobile carriers are depleted within the barrier, the resistivity of this barrier layer is extremely high. A small current will create a large voltage drop. Worse still, if the carriers flow from the abrupt side of the barrier, the abrupt potential profile prevents any carriers to fill up depletion layer from the abrupt side while the applied bias adds to the internal field and further depletes the other side of the barrier and hence widens the depletion layer. As a result, it becomes extremely difficult to inject any amount of current into the device with even tens of volts.

To illustrate the above barrier-depletion/voltage-drop effect, we simulate (using drift-diffusion model only) a highly n-doped junction of GaAs/InGaAlP with electrons flowing from the side of GaAs (smaller bandgap). The ideal abrupt junction forms a sharp barrier as shown in Fig. 9.4 in equilibrium. Fig. 9.5 illustrates the narrow depletion layer in the plot of electron concentration. As voltage biased is used to inject barriers from the side of GaAs, the carriers are unable to flow into the depletion layer while the applied field adds to the internal field to create a wider depletion layer as indicated in Fig. 9.6. The corresponding band diagram shows a large voltage drop in the widened depletion layer (see Fig. 9.7).

In reality, a huge voltage drop is not observed in a highly doped heterojunction due to the following reasons. First, the heterojunction is never ideally abrupt and we can conclude that the high barrier in Fig. 9.4 is exaggerated. It can easily be shown in a simulation that the barrier height is extremely sensitive to the abruptness of the junction. Secondly, the quantum tunneling effect makes it easy for mobile carriers to go over the barrier.

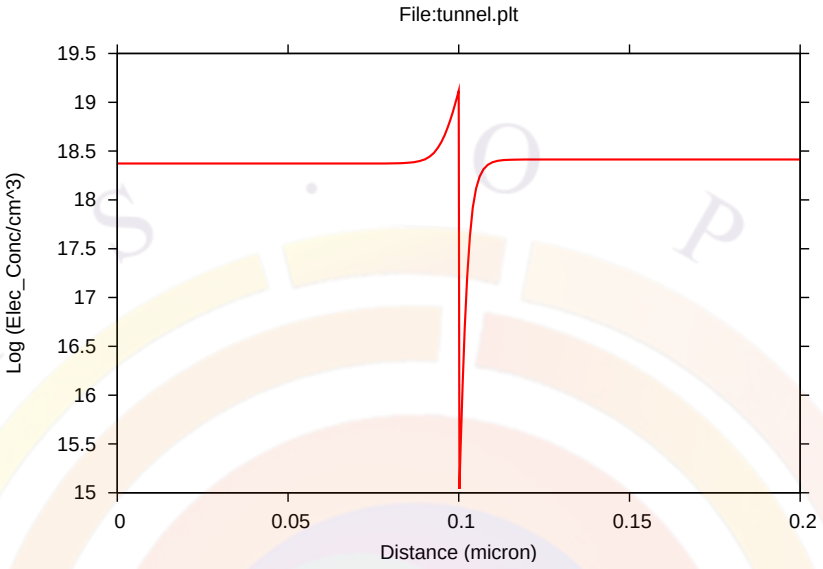


Figure 9.5: Electron concentration distribution of the heterojunction in equilibrium.

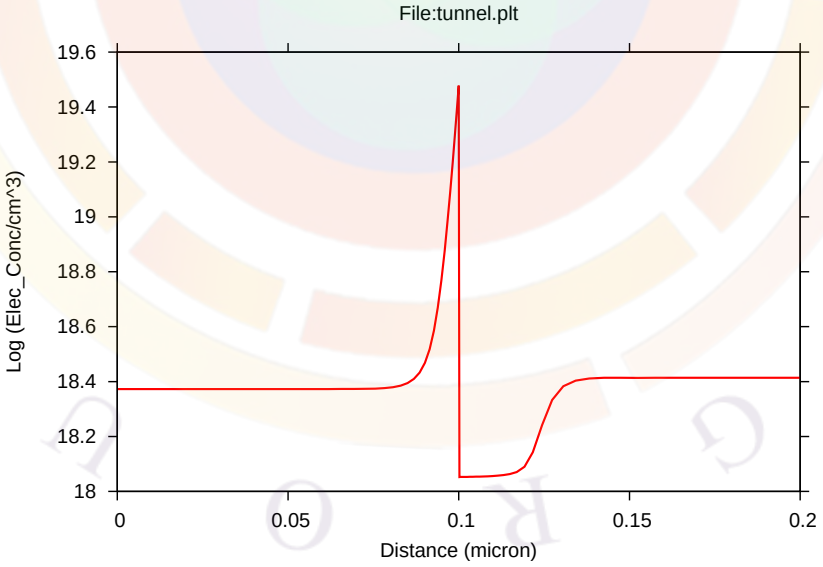


Figure 9.6: Electron concentration distribution of the heterojunction at a bias of five volts.

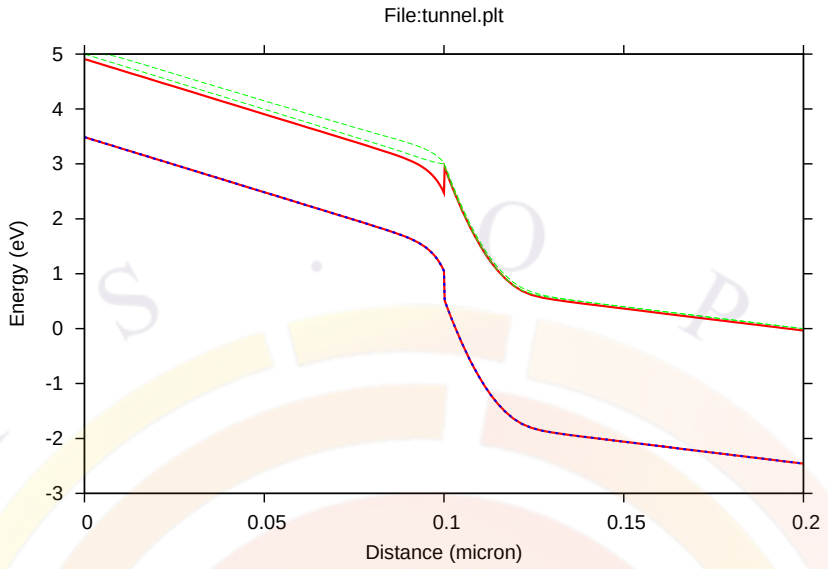


Figure 9.7: Band diagram of the heterojunction at a bias of five volts.

The analysis above lead us to propose the following approaches of modeling highly doped abrupt junctions. If we really wish to have a realistic model of transport through a highly doped junction, we should specify abruptness of the junction accurately based on material interface quality analysis/experiment. Then we are ready to apply the quantum tunneling model described in this section. We find that using tunneling on ideally abrupt junction usually underestimates the amount of current and still results in a voltage drop larger than that observed in experiment. We thus conclude that ideally abrupt junction model is inherently flawed under high doping condition.

On the other hand, if we wish to avoid dealing with the issues of deciding on the abruptness and using tunneling altogether, we can artificially grade the junction over a distance large enough to avoid forming a barrier but small enough not to disturb other part of the device. For example, if the junction in the above example is far away from the active region of a laser diode, it is safe to grade the junction over a distance of 100 Åwithout upsetting the active region while avoiding the formation of the barrier totally.

9.3 Interband Tunneling in Semiconductors

9.3.1 Theory

Consider the case of a direct tunneling between valence and conduction bands of semiconductor in external applied electric field. A p-n junction designed for such

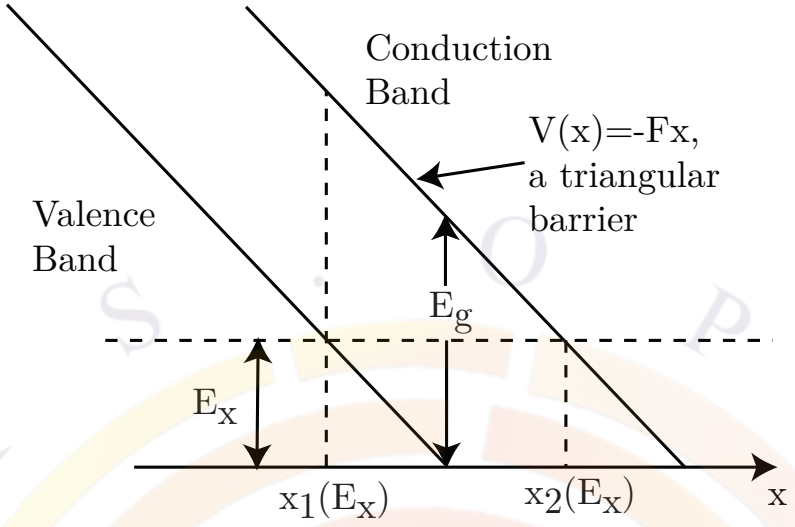


Figure 9.8: Schematic illustrating the Zener tunneling process.

a purpose is commonly referred to as a tunnel junction (TJ) or an Esaki junction. Such type of tunneling is also referred to as Zener tunneling as in a Zener tunneling diode under reverse bias. We assume a direct gap semiconductor with a parabolic band with a minimum or a maximum at a central Γ-point at ($\mathbf{k}$=0).

When strong external electric field is applied to a semiconductor, tunneling of electrons between conduction and valence band can take place generating electron-hole pairs. Newly created carriers will drift in a strong electric field and can result in impact ionization. Thus both impact ionization and Zener tunneling electron-hole generation rate should be included in drift-diffusion current solver together.

Similar to impact ionization, generation rate originating from Zener tunneling can be treated in a local electric field approximation, which is a reasonable approximation considering the tunneling distance is typically less than 100 Angstrom for a semiconductor of 1eV bandgap energy. Zener tunneling is usually observed for semiconductors at a field greater than F=10^6 V/cm.

Consider the electron state of energy E, tunneling from the valence to the conduction band as presented in Fig 9.8. Within the forbidden bandgap where E obeys E_g > E > 0, we have decayed wave function with imaginary wave vector $k_x = i\kappa$.

Neither the exact dependence of k_x, nor the potential barrier that the tunneling electron experiences, upon distance, x, is known through the forbidden gap. Similarly the effective mass remains an unknown and questionable parameter, while electron tunnels through the forbidden gap. Nevertheless, we can usually assume the tunneling mass takes a value somewhere between the conduction band mass and valence band mass. Following the derivation of band-to-band tunneling current in Refs. [58], [59]

and [60], we will calculate the probability of tunneling within WKB approximation as follows:

$$D = exp[-2J(E_{||})] \tag{9.41}$$

where:

$$J(E_{||}) = \int_{x_1}^{x_2} \kappa(x)dx. \tag{9.42}$$

$\kappa(x)$ is the decay constant of the tunneling electron wave function, given by:

$$\hbar^2\kappa^2/2m^* = |E_{||} - U|, \tag{9.43}$$

where $E_{||}$, and $E_{\perp}$ are electron kinetic energies in directions along and perpendicular to the tunneling direction, respectively.

After some derivation that are explained in Ref. [60], we can obtain the tunneling probability as follows:

$$D = exp(-2J) = P_0 exp(-E_{\perp}/\underline{E}) \tag{9.44}$$

where:

$$J(E_{||}) = \int_{x_1}^{x_2} \left|(2m^*/\hbar^2)[(E_g/2)^2 - (\varepsilon_c)^2]/E_g + E_{\perp}\right|^{\frac{1}{2}} dx \tag{9.45}$$

and

$$P_0 = exp\left[\frac{\pi m^{*\frac{1}{2}}(E_g)^{3/2}}{2(2)^{\frac{1}{2}}qF\hbar}\right] = exp\left(-\frac{E_g}{4\underline{E}}\right) \tag{9.46}$$

$$\underline{E} = \frac{(2)^{\frac{1}{2}}qF\hbar}{2\pi m^{*\frac{1}{2}}(E_g)^{\frac{1}{2}}} \tag{9.47}$$

P_0 has a meaning of the tunneling probability with a zero perpendicular (to x-direction) momentum. $\underline{E}$ is a measure of significance of perpendicular momentum range in Eq. (9.45). In other words, if $\underline{E}$ is small, the only electrons that can tunnel through the bandgap barrier are the electrons with perpendicular momentum near zero. Typically $\underline{E}$ is in a range of 5-100 meV. The effective tunneling mass is defined here, as in Ref. [60]:

$$m^* = 2m_c m_v/(m_c + m_v) \tag{9.48}$$

9.3.2 Application to reverse biased p-n junction diode

Next formula for band-to-band tunneling will be applied to calculations of a tunneling current density for a reverse biased n^+ p^+ junction. Following derivations in [60], consider the situation of a reverse biased p-n junction as in Fig. 9.9.

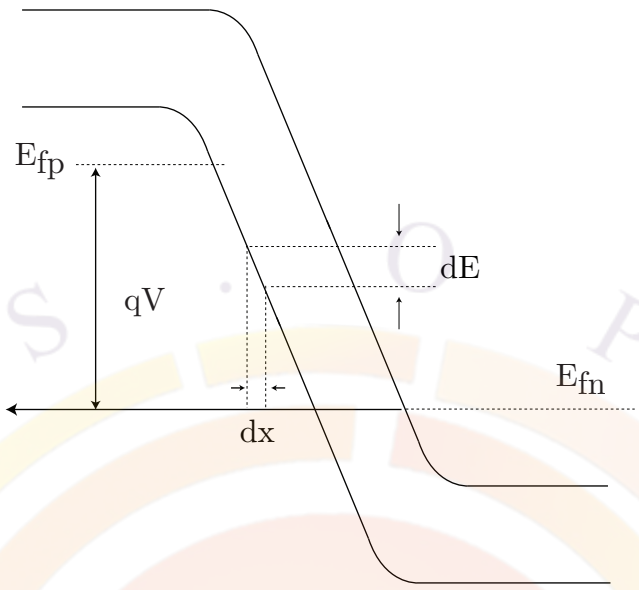


Figure 9.9: Schematic of a reverse biased Zener diode.

Let dE be an energy increment corresponding to incremental dx, then: $dE = qFdx$. If electron tunnels across the junction at energy E , the perpendicular motion kinetic energy must obey $E_\perp$ < E. The incident flux per unit volume with perpendicular momentum ranging from $k_\perp$ to ($k_\perp$ +$dk_\perp$) is:

$$\Phi = (q^2F/\hbar)\cdot(1/4\pi)^3\cdot(2\pi k_\perp dk_\perp)\cdot f_E(E) \tag{9.49}$$

where f_E(E) is a Fermi function. Using the relation $E_\perp$=$h^2\mathbf{k}_\perp{}^2/2m^*$ we can obtain:

$$\Phi = q^2Fm^*/(2\pi^2\hbar^3)f_E(E)dE_\perp \tag{9.50}$$

The tunneling flux must include also the necessity of final electron state being unoccupied, thus has to be multiplied by (1 - f'$_E$(E)) factor. As a result, the differential current density of a tunneling flow becomes:

$$dI/A = q^2Fm^*/(2\pi^2\hbar^3)P_0exp(-E_\perp/\underline{E})[f_E(E) - f'_E(E)]dE_\perp dx, \tag{9.51}$$

where dI/A = (incident flux) × (probability of tunneling) × (volume).

We can also modify the Eq. (9.51) using electrostatic force qF = dE/dx thus :

$$dI/A = qm^*/(2\pi^2\hbar^3)P_0exp(-E_\perp/\underline{E})[f_E(E) - f'_E(E)]dE_\perp dE \tag{9.52}$$

This current needs to be integrated with respect to $E_\perp$ in the limits of 0 < $E_\perp$ <E_{max}:

$$E_{max} = min(E_{vp} - E), (E_{cn} - E) \tag{9.53}$$

However, we simplify the integration of equation (9.52) by approximation of E_{max} to become an infinity, which is justified as long as E>> $\underline{E}$.

As typically $\underline{E} \approx$ 5-50 meV, the contribution to the tunneling process of those electrons with energy $E \approx \underline{E}$ can be neglected due to the small magnitude of an electric field near the edges of depletion layer of reverse biased p-n junction (See Fig. 9.9). Thus integration of equation (9.52) gives:

$$dI/A = qm^*/(2\pi^2\hbar^3)P_0\underline{E}[f_E(E) - f'_E(E)]dE \tag{9.54}$$

or using dE = qF dx,

$$dI/A = q^2Fm^*/(2\pi^2\hbar^3)P_0\underline{E}[f_E(E) - f'_E(E)]dx \tag{9.55}$$

or approximately:

$$dI/A = q^2Fm^*/(2\pi^2\hbar^3)P_0\underline{E}dx \tag{9.56}$$

where we used [f_E(E) - f'$_E$(E)] =1 for the range of reverse p-n junction.

Equation (9.56) can be rewritten in terms of e-h pair generation rate (in units: $m^{-3}s^{-1}$):

$$G_{Zen}(F) = dI/A/(qdx) = qFm^*/(2\pi^2\hbar^3)P_0\underline{E} \tag{9.57}$$

where

$$P_0 = exp\left[\frac{\pi m^{*\frac{1}{2}}(E_g)^{3/2}}{2(2)^{\frac{1}{2}}qF\hbar}\right] = exp\left(-\frac{E_g}{4\underline{E}}\right) \tag{9.58}$$

and

$$\underline{E} = \frac{(2)^{\frac{1}{2}}qF\hbar}{2\pi m^{*\frac{1}{2}}(E_g)^{\frac{1}{2}}} \tag{9.59}$$

This expression for a local field dependent generation rate can be subsequently substituted into the drift diffusion equation solver as a carrier generation term which can be activated using the command **zener**.

9.3.3 Non-local model and forward biased tunnel junction

The model we developed in previous subsections expressed was converted into a volume integral over the whole junction area for each mesh point, making it a local carrier generation term in the drift-diffusion equation. We find that such an implementation may suffer from numerical non-convergence near zero bias since the generation term does not have a zero-current offset mechanism. Basically, the high field within the junction causes a high generation rate even at zero bias. Without current flow to balance out the carrier generation, the program may run into convergence difficulties.

We have developed an alternative and more stable implementation of the TJ model based on a form of direct flow between two reference points on either side of the TJ. These reference points must be closely spaced (∼10 nm) so that the quantum effects are important but they must also be some distance away from the junction itself so that the band profile is almost flat. This minimizes the effect of the junction on the carrier densities used to compute the tunneling current. This model has been implemented in the command **tunnel_junc** and can be activated by setting the parameter **use_physical_model**.

The formulas for forward biased tunnel junction take the same form as those for the reverse one except that the approximation we made in Eq. (9.52) for infinite integration range for $dE_\perp$ is no longer in use. Furthermore, the Fermi occupancy factor $[f_E(E) - f'_E(E)]$ is much smaller than unity and must be carefully evaluated. The tunneling model for forward-biased junctions is also implemented using **tunnel_junc**.

An application note on forward TJ is warranted here. We find that the forward current behavior is strongly affected by the potential and current injection conditions surrounding the TJ area. For many cases of non-linear injection condition, the negative resistance (N-shape) expected for forward TJ does not appear. The reason is that non-linear injection (such as through an n-i junction or a Schottky barrier) may overshadow a narrow and weak N-shape peak. To produce the negative resistance behavior from the software, we should first start with a pure TJ without perturbation of external doping profile and contact effects. For example, we should start with a TJ where the n-side and p-side are reasonably uniform in doping and with ohmic contacts on both sides. Such ideal conditions almost always produce an N-shape in forward I-V curve given sufficient carrier population inversion across the junction and sufficiently small voltage bias steps. A more complete device can be built up from this TJ one step at time, maintaining the N-shape at each step.

Other tunneling models

Other interband tunneling models (e.g. phonon-assisted tunneling) have been implemented in the software; see **tunnel_junc** for a list of supported models. Trap-assisted tunneling can also be defined with **trap_assisted_tunneling**.

Chapter 10

FINER POINTS ABOUT QUANTUM WELLS

10.1 Band Offset In Strained Quantum Well

10.1.1 Bulk band alignment

Before discussing QW band offset, it is helpful to review how bulk band alignment works in Crosslight device simulation. Simply put, for bulk layers the **affinity** statement controls everything.

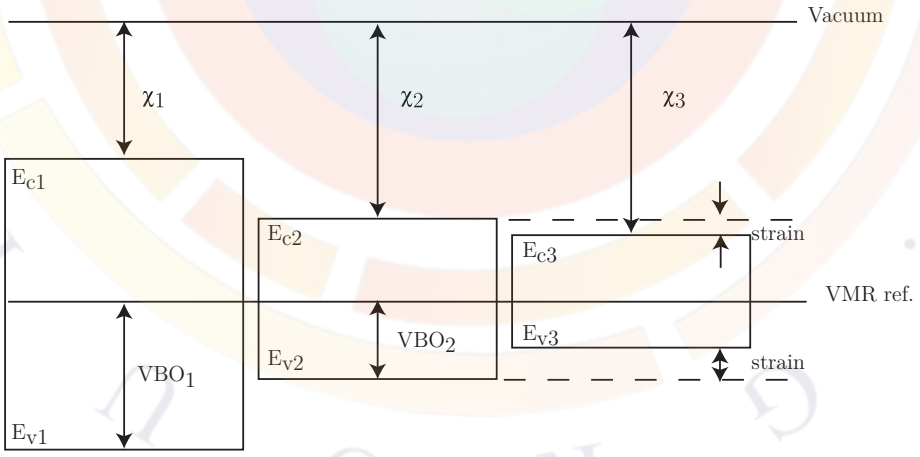


Figure 10.1: Schematic illustrating the alignment of bulk bands in Crosslight. The VMR line represents the energy reference value used in Ref. [61].

In Fig. 10.1 we show a schematic of the band alignment for 3 materials: materials #1 and #2 are unstrained while material #3 is a strained version of material #2. In the Crosslight system, the vacuum level is taken as an absolute energy reference and

the position of any bulk conduction band is fixed once the affinity χ is defined. If a bulk layer is strained, then it is the responsibility of the user to include the strain-induced shift in the affinity declaration for that layer (i.e. χ_3). It is important to note that this rule applies to both passive and optically active bulk materials, even if the latter uses an additional macro to define parameters needed for stand-alone gain calculations.

Readers experienced in device modeling may have noticed that many references that list semiconductor material parameters do not provide the affinity. This is the case for the excellent review article by Vurgaftman, Meyer and Ram-Mohan [61] which instead, defines the valence band offsets (VBO) for various unstrained materials. As Fig. 10.1 makes clear, since both the vacuum level and this reference energy are absolutes then conversion between the two conventions for any given group of materials is straightforward once a single affinity value (e.g. that of the substrate) has been fixed.

After the conduction band is set, Crosslight tools define the valence band position(s) by adding the bandgap to the affinity value. For passive layers, a single hole band with a reduced effective mass is used for the carriers. For bulk active layers, multiple band valleys may be used if they are needed to compute the gain and spontaneous emission spectra.

Note that if a bulk passive layer is strained, the strained bandgap may be required to correctly position the valence band; this is not supported in all material systems. See **wurtzite_offset_model** and **zincblende_offset_model** for more details.

10.1.2 Band offset in QW

How are band offsets used ?

Band offsets in quantum well regions are used in two ways. The first is for the Schrödinger solver: under flat band conditions, the barrier heights (confining potentials) for the electrons and holes are needed to solve the problem and the absolute band positions do not matter. However, as part of the finite difference discretization, a zero potential reference is defined at the first mesh point of the first quantum well inside the quantum-confined region and the offset is applied to the leftmost barrier.

The second way in which band offset is used is when the QW region is added to the full device simulation: in this case, the absolute position of the QW band edges is needed in order to do the alignment with the other layers. For this step, Crosslight tools use the relative barrier heights from this first stage and position the QW band edges using the bulk band edges of the outer barrier. This “cut & paste” operation preserves the barrier height from the Schrödinger solver in the full device simulation.

It is therefore convenient if the outer barrier of a MQW region is unstrained in

order to correctly define the absolute band position of the MQW. Otherwise, a strained bulk affinity should be used to correctly define both the barrier height and the absolute position of the band edges. This situation is known to occur in strain-balanced MQW regions (tensile barrier/compresive well) based on In(1-x-y)Ga(x)Al(y)As .

How is the band offset defined ?

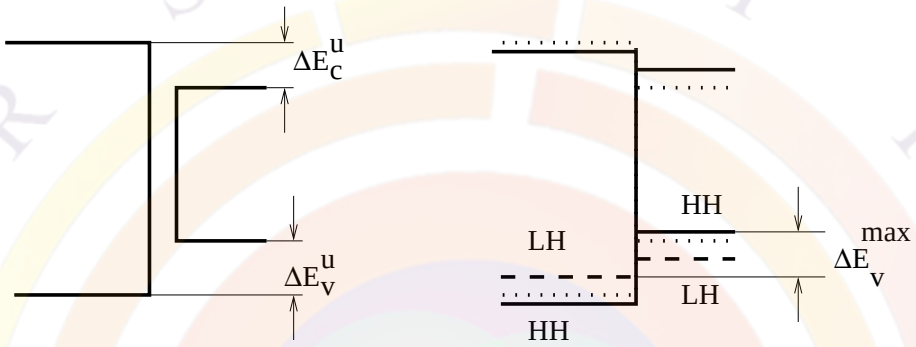


Figure 10.2: Schematics illustrating the definition of band offset.

In a quantum well without strain in the well or the barrier, the band offset is easily defined as the fraction of bandgap discontinuity in the conduction band:

$$O_{ff} = \Delta E_c/(\Delta E_c + \Delta E_v) \tag{10.1}$$

However when there is strain in the quantum well and barrier, there are many ways of defining the band offset. Three popular methods are explained below:

- 1) Use of unstrained bandgap discontinuity:

$$O_{ff} = \frac{\Delta E_c^u}{\Delta E_c^u + \Delta E_v^u} \tag{10.2}$$

  where superscript ^{u} is used to denote unstrained quantities (see Fig. 10.2). This convention is used by [15] and is most convenient for theoretical derivations, since the valence band potential can be written as one minus band offset times unstrained bandgap difference and then adding the additional strain terms.

  This band offset definition is also the one which most closely relates to the electron affinity of the material.

- 2) Use of HH valence band and the strained conduction band edge:

$$O_{ff} = \frac{\Delta E_c^{min}}{\Delta E_c^{min} + \Delta E_v^{HH}} \tag{10.3}$$

where ΔE_c^{min} is defined as (for zincblende):

$$\Delta E_c^{min} = Min(E_{cb}^{\Gamma}, E_{cb}^{L,X}) - Min(E_c^{\Gamma}, E_c^{L,X}) \tag{10.4}$$

and the subscript $_b$ has been used to denote quantities in the barrier.

For zincblende, this secondary band valley (L or X) may be important at high injection levels (inter-valley transfer). For wurtzite though, no additional conduction band is considered.

- 3) Use of the highest valence band and the strained conduction band edge:

$$O_{ff} = \frac{\Delta E_c^{min}}{\Delta E_c^{min} + \Delta E_v^{max}} \tag{10.5}$$

where for zincblende materials:

$$\Delta E_v^{max} = Max(E_v^{HH}, E_v^{LH}) - Max(E_{vb}^{HH}, E_{vb}^{LH}) \tag{10.6}$$

and for wurtzite materials:

$$\Delta E_v^{max} = Max(E_v^{HH}, E_v^{LH}, E_v^{CH}) - Max(E_{vb}^{HH}, E_{vb}^{LH}, E_{vb}^{CH}) \tag{10.7}$$

Here again, the subscript $_b$ has been used to denote quantities in the barrier.

By default, Crosslight tools always use method #3. The advantage of using this method is that the bands in question are the ones which contain most of the carriers; it is therefore believed that this value will best reflect experimental estimates of band offset. This default behavior can be changed with the **wurtzite_offset_model** and **zincblende_offset_model** commands. However, the above commands only change the bandgap value used to calculate the band offset: the strained conduction band edge is still used. Put in other words, the **band_offset** value provided to the simulation *should already include* any strain-induced shifts in the conduction band.

Users who prefer to use method #1 to define the band offset should explicitly define the strained conduction band position (χ_3) in the affinity statement as described above. Thus, the **affinity** statement should define both the unstrained band offset and the explicit shift in the conduction band. This strained conduction band position can then be enforced in the MQW region by invoking **use_bulk_affinity** which will override the **band_offset** statement.

How does band offset work in a complex well region ?

As shown in Fig. 8.1, the basic QW model consists of symmetric and isolated quantum wells. In such a case, the band offset used on the left side is also applied on the right.

For a single complex well region (using material parameter macros labeled with a "cx-" prefix) barrier symmetry is not guaranteed but we still use a single band offset fraction for that well. The resulting band discontinuity may therefore be asymmetric if the barrier bandgaps are different; this behavior can further be controlled by using the **band_discont** and **band_discont_right** statements to directly specify the band discontinuity on either side of the well.

If multiple wells are considered as part of the same quantum-confined region, the default behavior of Crosslight tools is to assume that the MQW profile consists of a barrier/well/barrier/well/...../barrier layout; that is, even-numbered layers are considered to be wells. The band offset rules for a single complex well are then applied on each well in the complex, going from left to right; for each well, the band discontinuity is calculated using the offset fraction for that well and the bandgap difference between well and barrier. This is true even if the **inner_bar_gain** statement is used: this statement only affects the gain profile and the partition of the carriers, not the band alignment.

So in this complex MQW case, we start by defining the position of the first well (layer #2) relative to position the first barrier (layer #1) and then the band discontinuity on the right defines the position of the second barrier (layer #3). The second well (layer #4) is then positioned relative to the previous barrier (layer #3) using the left band discontinuity for that well; we then fix the next barrier's position (layer #5) using the right band discontinuity. This process continues until we reach the end of the complex MQW.

The band offset/band discontinuity must therefore be defined in the macros for all well layers; these same parameters are ignored in barrier layers. Odd/unexpected profiles may therefore be created when mixing different materials or wells having different offset/discontinuity values; this system also creates complications when defining graded well profiles where "wells" are not in even-numbered positions.

How does band offset work in the SCXLIB system ?

As introduced in Sec. 3.5.1, the new simplified complex library system (SCXLIB) provides an alternate method of defining complex MQW regions. In this system, all of the layers tagged with **model**=*quantum_well* are considered to be wells and the un-tagged layers on either side serve as the barrier materials.

Starting with v.2016, a new band alignment system has been designed for this system: the position of the conduction band in each well is calculated using the band offset coefficient for that well and the bandgap difference between that well and the first barrier. Since all the inner layers of the MQW are considered wells in this system, this positions all of the wells using a common reference.

The only layer which has an ambiguity is the right barrier. By default, this is left un-

touched in the SCXLIB system and this layer is exempted from the "cut & paste" operation we previously mentioned. However, this behavior can be changed by issuing either the **barrier_correction_by_qw_model** or **force_last_barrier_offset** statements. In that case, the right barrier is either offset by a set amount or simply follows the same rules as for the other QW regions. Since the older macro syntax for both complex wells and basic QWs previously overrode the right barrier position, using one of these statements is recommended in most cases.

As always, users may also rely on the **use_bulk_affinity** statement to correctly position all of the layers inside a MQW region; this statement applies to both the old complex MQW macro syntax and the new library syntax.

10.1.3 Definition of strain terms

A full description of the strain-induced terms is beyond the scope of this section; it can easily be found in the literature and the exact form depends on the crystal structure (zincblende or wurtzite). Users who wish to read more on this topic are invited to read the many papers by S.L. Chuang [15, 33, 62–66].

For now, it suffices to say in zincblende materials the strain is described by both a hydrostatic term (δE_{hy}) and a shear term (δE_{sh}). The hydrostatic term is split between the conduction and valence bands while the shear term splits the HH and LH bands. This results in different bandgaps for the HH and LH valleys in strained layers[33]:

$$E_g^{HH} = E_g^u - \delta E_{hy} + \frac{1}{2}\delta E_{sh}, \tag{10.8}$$

$$E_g^{LH} = E_g^u - \delta E_{hy} - \frac{1}{2}\delta E_{sh} \tag{10.9}$$

For band alignment purposes, the exact split of the hydrostatic term is needed. In the software, this is set by a fraction which describes how much of the hydrostatic shift should be applied in the valence band. By default, this value is set as $\frac{a_v}{a} = \frac{1}{3}$ but in reality, this ratio depends on the material system; it should adjusted in either **active_reg** or **set_active_reg** for optimal simulation results.

In wurtzite materials the conduction and valence band deformation potentials are explicitly defined in the material macro (e.g. **ac_well**, **d1_well**, etc...). This topic shall be addressed in more detail in Chap. 13.

10.2 Intraband Relaxation Times and Gain Broadening

10.2.1 Introduction

The influence of line shape broadening on optical gain can be described by

$$G(\hbar\omega) = \int_{E_g'}^{\infty} g(E_{cv})L(\hbar\omega - E_{cv})dE_{cv} \tag{10.10}$$

here $G(\hbar\omega)$ is the gain coefficient as a function of photon energy $\hbar\omega$. $L(\hbar\omega - E_{cv})$ is the lineshape broadening function characterized by the intraband relaxation. This lineshape function is often represented as

$$L(\hbar\omega - E_{cv}) = \frac{1}{\pi}\frac{\hbar/\tau_{in}}{(\hbar\omega - E_{cv})^2 + (\hbar/\tau_{in})^2}, \tag{10.11}$$

where τ_{in} is called the intraband relaxation time, which is regarded as the reciprocal intraband scattering probability, and $\hbar/\tau_{in}$ is the broadening factor we will study in this section. Previously, we just assume τ_{in} to be a constant of the order of 10^{-13} seconds. The lack of accurate model for this parameter causes significant uncertainty in the optical gain.

Our theory here is limited to the case of intraband relaxation due to carrier-carrier and carrier-LO-phonon scattering in quantum wells. For details, please consult references [67] and [68].

We consider optical transition between subband energy $E_{vik_\parallel}$ in the valence band and $E_{cjk_\parallel}$ in conduction band. The energy is expressed as

$$E_{cik_\parallel} = E_{ci} + \hbar^2 k_\parallel^2/2m_c \tag{10.12}$$

$$E_{vjk_\parallel} = E_{vj} + \hbar^2 k_\parallel^2/2m_v \tag{10.13}$$

where the (E_{ci} and E_{vj}) are the quantized energy levels, m_c and m_v are the effective masses. $k_\parallel$ is the wave vector parallel to the well interface.

When discussing scattering mechanisms occurring in conduction band and valence band, it is convenient to introduce broadening factors Γ_c and Γ_v, respectively. According to theory of Green's function [67],[68], these are related to the decay rate of amplitude of the wave function. For example, the expectation value of the amplitude of the electron wave is approximated as $\psi_c \propto exp(-\Gamma_c t/\hbar)$ in the conduction band, including thermal and quantum- mechanical statics. Thus, the number of electrons at this energy level decays as $exp(-2\Gamma_c t/\hbar)$. The decay rate in the valence band is obtained similarly. Therefore, the broadening factors are related to the intraband

relaxation times (a measure of particle decay probability) in the conduction and valence bands (τ_c and τ_v) by

$$\hbar/\tau_c = 2\Gamma_c(E_{cik_\parallel}) \tag{10.14}$$
$$\hbar/\tau_v = 2\Gamma_v(E_{vjk_\parallel}) \tag{10.15}$$

In optical transition, the dipole moment between electron and hole is proportional to $\psi_c\psi_v^* \propto exp[-(\Gamma_c+\Gamma_v)t/\hbar]$. Thus, the intraband relaxation time determining the spectral broadening, which is the relaxation time of the dipole moment is obtained as

$$\begin{aligned}\hbar/\tau_{in} &= \Gamma_v + \Gamma_c \\ &= (\hbar/\tau_c + \hbar/\tau_v)/2 \end{aligned} \tag{10.16}$$

In the following subsections, broadening factors $\hbar/\tau_c$ and $\hbar/\tau_v$ determined by carrier-carrier and carrier-LO phonon scattering are calculated, which are dominant in lightly-doped compound semiconductors.

10.2.2 Carrier-Carrier Scattering

The intraband relaxation time in the conduction band (τ_c) due to carrier-carrier scattering is calculated from the imaginary part of self energy using the Green's function method[67]. In the present analysis, the calculation is based on second-order self energy. Only the product between two matrix elements with the same the momentum change is taken into account in various terms of the self energy.

In this treatment, $\hbar/\tau_c$ is calculated by Fermi's Golden Rule and the spectral broadening factor for scattering in conduction band is written as

$$\hbar/\tau_c = (2\pi)\sum_{k'_\parallel p'_\parallel}\sum_{ij} \mid V_{cn}(k_\parallel k'_\parallel, ii'jj') \mid^2 \cdot A_c(B_c + C_c) \tag{10.17}$$

with

$$A_c = \delta(E_{cik_\parallel} + E_{njp_\parallel} - E_{ci'k'_\parallel} - E_{nj'p'_\parallel}) \tag{10.18}$$
$$B_c = f_c(E_{ci'k'_\parallel}) f_n(E_{nj'p'_\parallel}) \left[1 - f_n(E_{njp_\parallel}\right] \tag{10.19}$$
$$C_c = \left[1 - f_c(E_{ci'k'_\parallel})\right] \left[1 - f_n(E_{nj'p'_\parallel})\right] f_n(E_{njp_\parallel}) \tag{10.20}$$

where the suffix n refers to the conduction ($n = c$) or valence ($n = v$) band, i, i', j, and j' are subband numbers, the δ-function represents the energy conservation, and V_{cn} is the matrix element of the interaction given below.

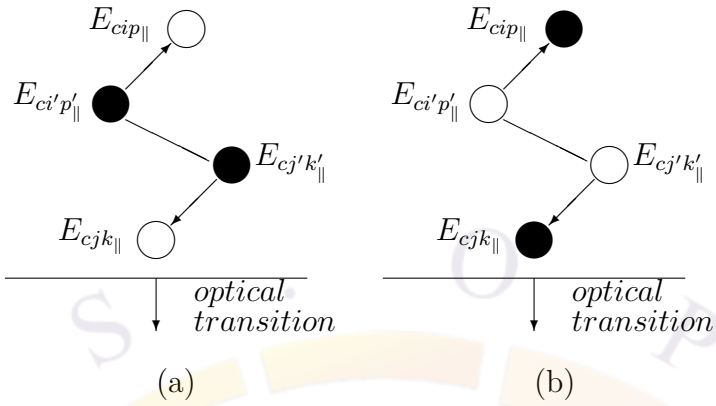


Figure 10.3: Schematic illustration of electron-electron scattering in the conduction band. (a) Formation of a hole at energy $E_{cjk_\parallel}$ under the optical transition is intercepted due to collision of two electrons at $E_{cj'k'_\parallel}$ and $E_{ci'p'_\parallel}$, resulting in spectral broadening. (b) Formation of an electron at energy $E_{cjk_\parallel}$ under the optical transition is scattered due to collision of two holes.

A simple illustration is possible for the above formula. The scattering processes are schematically shown in Figs. 10.3(a) and 10.3(b), for $n = c$.

In Fig. 10.3(a), two electrons at $E_{ci'p'_\parallel}$ and $E_{cj'k'_\parallel}$ collide with each other, and are scattered into two holes at $E_{cip_\parallel}$ and $E_{cjk_\parallel}$. Formation of a hole at $E_{cjk_\parallel}$ during optical transition is intercepted by this process, resulting in spectral broadening.

In Fig. 10.3(b), optical transition of an electron at $E_{cjk_\parallel}$ is intercepted by the collision of two holes at $E_{cj'k'_\parallel}$ and $E_{ci'p'_\parallel}$.

A process for $n = v$ in Eq. (10.17) is similarly given. Electron-electron scattering and electron-hole scattering correspond to $n = c$ and $n = v$, respectively. However, the formula for the spectral broadening factor caused by another scattering electron-LO phonon ($n = LO$) is radically different, which we will discuss in the next subsection.

For the interaction matrix element V_{cn} in (2.16), we use the screened Coulomb potential, which is the result of static and long wavelength limit of the many-body interaction. Deviation from the exact calculation is discussed below

$$V_{cn}(k_\parallel k'_\parallel, ii'jj') = \int\int \psi^*_{ci'k'_\parallel}(r_1)\psi^*_{nj'p'_\parallel}(r_2)\frac{e^2\exp(-\lambda_s r)}{4\pi\epsilon r} \cdot\psi_{cik_\parallel}(r_1)\psi_{njp_\parallel}(r_2)dr_1dr_2 \tag{10.21}$$

where e is the electron charge, ϵ is the static dielectric constant, $r = \mid r_1 - r_2 \mid$, and λ_s is the inverse screening length given below; ψ's are the electron wave functions, which are approximated as follows in the well potential:

$$\psi_{njk_\parallel}(r) \simeq u_{nk}(r)\Phi_{nj}(z)\exp(ik_\parallel \cdot r_\parallel)/\sqrt{S} \tag{10.22}$$

with

$$\Phi_{nj}(z) = \sqrt{2/L_{nj}}\sin(j\pi z/L_{nj}) \tag{10.23}$$

where the z axis is perpendicular to the well interface, S is the interface area of the sample, u is the periodic part of the bulk Bloch function, $k_\parallel$ and $r_\parallel$ are the components of the wave vector and position vector parallel to the interface, respectively, and L_{nj} is the effective well width given by

$$\begin{aligned} L_{nj} &= \pi/k_{nj\perp} \\ &= \pi\hbar/\sqrt{2m_n E_{nj}} \end{aligned} \tag{10.24}$$

where $k_{nj\perp}$ is the z component of the wave vector.

The inverse screening length λ_s is given as follows for the carrier injection case, where electrons and holes simultaneously exist:

$$\begin{aligned} \lambda_s^2 &= -\frac{e^2}{\epsilon}\left\{\int_{E_{c1}}^{\infty}\frac{\partial f_c}{\partial E_c}g_c dE_c + \int_{E_{v1}}^{\infty}\frac{\partial f_v}{\partial E_v}g_v dE_v\right\} \\ &= \frac{e^2}{\pi\hbar^2\epsilon}\sum_j\{m_c f_c(E_{cj})/L_{cj} + m_v f_v(E_{vj})/L_{vj}\} \end{aligned} \tag{10.25}$$

where g_c and g_v are the step-like density-of-states of the conduction and valence bands, respectively. We sum over all the subbands and both heavy hole and light hole.

Substituting Eq. (10.22) into Eq. (10.21) the matrix element is calculated as

$$\begin{aligned} V_{cn}(k_\parallel k'_\parallel, ii'jj') &= \frac{e^2}{2\epsilon S}\frac{\delta(k_\parallel - k'_\parallel, p'_\parallel - p_\parallel)}{\sqrt{g_\parallel^2 + \lambda_s^2}} \\ &\cdot\int\int\Phi^*_{ci'}(z_1)\Phi_{ci}(z_1)\Phi^*_{nj'}(z_2)\Phi_{nj}(z_2) \\ &\cdot\exp(-\mid z_1 - z_2\mid\sqrt{g_\parallel^2 + \lambda_s^2})dz_1 dz_2 \end{aligned} \tag{10.26}$$

where the δ-notation represents the momentum conservation within the plane parallel to the interface, and $g_\parallel = k_\parallel - k'_\parallel$.

Generally, the possibility of scattering in the same subband is much higher than in the different subbands, so we only consider one subband in either the conduction or valence bands ($i' = i, j = j'$). The absolute square of the matrix element is calculated as

$$\begin{aligned} \mid V_{cn}(k_\parallel k'_\parallel, iij'j')\mid^2 &= (e^2/2\epsilon S L_e)^2 \\ &\cdot\delta(k_\parallel - k'_\parallel, p'_\parallel - p_\parallel)T(g_\parallel, k_{ci\perp}, k_{nj'\perp}) \end{aligned} \tag{10.27}$$

where L_e is the minimum of the effective well widths of the four wave functions in (10.26), and

$$T(g_{\parallel}, k_{ci\perp}, k_{nj'\perp}) = \left[\frac{2}{\alpha} + \frac{\delta(k_{ci\perp}, k_{nj'\perp})}{\alpha + 4k_{ci\perp}^2} - \frac{2}{L_e\sqrt{\alpha}} A_t(B_t + C_t)\right]^2 \tag{10.28}$$

with

$$A_t = 1 - \exp(-L_e\sqrt{\alpha}) \tag{10.29}$$

$$B_t = \frac{1}{\alpha} + \frac{\alpha + 4k_{ci\perp} + 4k_{nj'\perp}^2}{(\alpha + 4k_{ci\perp}^2)(\alpha + 4k_{nj'\perp}^2)}) \tag{10.30}$$

$$C_t = \frac{L_e k_{ci\perp}(L_e k_{ci\perp} - \pi)(\alpha - 4k_{nj'\perp}^2)}{4(\alpha + 4k_{ci\perp}^2)(\alpha + 4k_{nj'\perp}^2)\left[\exp(L_e\sqrt{\alpha}) - 1\right]} \tag{10.31}$$

$$\alpha = g_{\parallel}^2 + \lambda_s^2 \tag{10.32}$$

The first and second terms within the square bracket of Eq. (10.28) are essentially the same as those in bulk semiconductors. These terms arise from the forward and backward waves in the standing wave along the z axis given by Eq. (10.23), the third term expressed by Eqs. (10.29)- (10.31), which approaches zero at the limit $L \to \infty$, is peculiar to the quantum-well structures, and results from the localization of the wave function. Equation (10.28) coincides with that for the perfect two-dimensional case at the limit $L \to 0$ due to the existence of the third term.

Replacing the summation for $k'_{\parallel}$ and $p'_{\parallel}$ in with the integral for the wave vector parallel to the well interface, the broadening factor $\hbar/\tau_{cn}$ is given as follows:

$$\hbar/\tau_{cn} = \frac{m_c e^4}{8\pi^5\hbar^2\epsilon^2}(\frac{m_n}{m_c})^2 \int_0^{2\pi} d\phi \int_0^{\infty} dk'_{\parallel} \int_0^{\infty} du A_\tau(B_\tau + C_\tau) \tag{10.33}$$

with

$$A_\tau = T(g_{\parallel}, k_{ci\perp}, k_{nj'\perp})(k_{ci\perp}^2 k'_{\parallel} \mid \beta \mid /g_{\parallel}^2) \tag{10.34}$$

$$B_\tau = f_c(E_{cik'_{\parallel}}) f_n(E_{nj'p'_{\parallel}}) \left[1 - f_n(E_{nj'p_{\parallel}})\right] \tag{10.35}$$

$$C_\tau = \left[1 - f_c(E_{cik'_{\parallel}})\right] \left[(1 - f_n(E_{nj'p'_{\parallel}})\right] f_n(E_{nj'p_{\parallel}}) \tag{10.36}$$

and

$$g_{\parallel}^2 = k_{\parallel}^2 + k_{\parallel}'^2 - 2k_{\parallel}k'_{\parallel}\cos\phi \tag{10.37}$$

$$\beta = k_{\parallel}'^2 - k'_{\parallel}k_{\parallel}\cos\phi + (m_c/m_n - 1)g_{\parallel}^2/2 \tag{10.38}$$

$$E_{cik'_{\parallel}} = (\hbar^2/2m_c)(k_{\parallel}'^2 + k_{ci\perp}^2) \tag{10.39}$$

$$E_{nj'p'_{\parallel}} = (\hbar^2/2m_n)[(m_n/m_c^2(u^2+1) + g_{\parallel}^2 - 2(m_n/m_c)\beta + k_{ci\perp}^2] \tag{10.40}$$

$$E_{nj'p_{\parallel}} = E_{cik'_{\parallel}} + E_{nj'p'_{\parallel}} - E_{cik_{\parallel}} \tag{10.41}$$

where the parabolic band structure is assumed, and the summation with respect to n corresponds to electron-electron scattering and electron-hole scattering for $n = c$ and $n = v$, respectively.

In the same manner, carrier-carrier scattering also occurs for holes in the valence band, corresponding to the processes in the conduction band shown in Figure 10.3. The spectral broadening factor for the valence band $\hbar/\tau_v$ is calculated by the same procedure as above, and obtained by replacing the suffix c with v.

10.2.3 Carrier-Phonon Scattering

The spectral broadening factor in the conduction band due to carrier-LO phonon scattering $\hbar/\tau_{cLO}$ is calculated in a manner similar to carrier-carrier scattering as

$$\hbar/\tau_{cLO} = (2\pi)\sum_{k'_\parallel}\sum_{j'} \mid V_{cLO}(k_\parallel k'_\parallel, jj') \mid^2 (A_q + B_q) \tag{10.42}$$

$$A_q = \delta(E_{cjk_\parallel} - E_{cj'k'_\parallel} \pm \hbar\omega_{LO}) f_c(E_{cj'k'_\parallel}) \tag{10.43}$$

$$B_q = \delta(E_{cj'k'_\parallel} - E_{cjk_\parallel} \pm \hbar\omega_{LO}) \left[1 - f_c(E_{cj'k'_\parallel})\right] \tag{10.44}$$

where $\hbar\omega_{LO}$ is the phonon energy, $+\hbar\omega_{LO}$ and $-\hbar\omega_{LO}$ correspond to emission and absorption of phonons, respectively, and V_{cLO} is the matrix element of the carrier-LO phonon scattering, and is given below.

The scattering processes given by (10.43) and (10.44) are schematically shown in Figures 10.4(a) and 10.4(b), respectively.

Assuming the electron wave function of (10.22), the square of the matrix element $\mid V_{cLO}(k_\parallel k'_\parallel, jj') \mid^2$ is given as follows, taking into account the screening effect:

$$\mid V_{cLO}(k_\parallel k'_\parallel, jj') \mid^2 = \sum_q (e^2\hbar\omega_{LO}/2V)(1/\epsilon_\infty - 1/\epsilon) C_q \cdot D_q \tag{10.45}$$

with

$$C_q = \left[q/(q^2 + \lambda_s^2)\right]^2 \left\{ \begin{array}{c} n_q + 1 \\ n_q \end{array} \right\} \delta(k_\parallel - k'_\parallel, q_\parallel) \tag{10.46}$$

$$D_q = \mid \int \Phi^*_{cj}(z)\Phi_{cj'}(z) e^{-iq_\perp z} dz \mid^2 \tag{10.47}$$

where q, $q_\parallel$, and $q_\perp$ are the phonon wave vector and its components parallel and perpendicular to the well interface, respectively. ϵ and ϵ_∞ are the static and optical dielectric constants, respectively, V is the volume of the system, and the factors $n_q + 1$ and n_q correspond to the emission and absorption of phonons, respectively, where n_q is the phonon number per mode q given by

$$n_q = 1/(\exp(\hbar\omega_{LO}/kT) - 1. \tag{10.48}$$

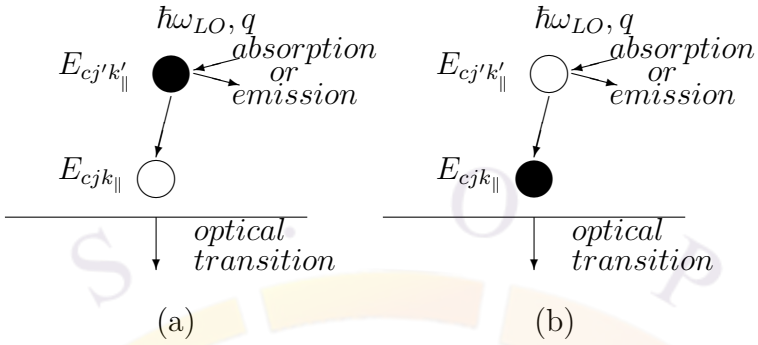


Figure 10.4: Schematic illustration of electron-LO phonon scattering in the conduction band. (a) Formation of a hole at energy $E_{cjk_\parallel}$ under the optical transition is intercepted by an electron at $E_{cj'k'_\parallel}$ which emits or absorbs a phonon. (b) Formation of an electron at energy $E_{cjk_\parallel}$ under the optical transition is scattered by a hole $E_{cj'k'_\parallel}$ which emits or absorbs a phonon.

The intensity of the phonon mode confined in the well is assumed to be much smaller than that of the bulk mode, and is neglected in (10.45)-(10.47).

Replacing the summation for $k'_\parallel$ and q in (10.42)-(10.47) with the integral for the wave vector, $\hbar/\tau_{cLO}$ is calculated for one subband case ($j = j' = i$) as

$$\hbar/\tau_{cLO} = (m_c e^2/2\pi^2\hbar^2)(1/\epsilon_\infty - 1/\epsilon)\hbar\omega_{LO}(A_o + B_o) \tag{10.49}$$

with

$$\begin{aligned} A_o &= \left[n_q + 1 - f_c(E_{cik_\parallel} - \hbar\omega_{LO})\right] \\ &\quad \cdot \int_0^{\phi_{max}} \frac{(I(a_1) + I(a_2))d\phi}{\sqrt{k_\parallel^2 \cos^2\phi - 2m_c\omega_{LO}/\hbar}} \end{aligned} \tag{10.50}$$

$$\begin{aligned} B_o &= \left[n_q + f_c(E_{cik_\parallel} + \hbar\omega_{LO})\right] \\ &\quad \cdot \int_0^{\pi} \frac{I(a_3)d\phi}{\sqrt{k_\parallel^2 \cos^2\phi + 2m_c\omega_{LO}/\hbar}} \end{aligned} \tag{10.51}$$

and

$$\phi_{max} = \cos^{-1}(\sqrt{2mc\omega_{LO}/\hbar k_{\parallel}^2}) \tag{10.52}$$

$$I(a) = \int_0^{\infty} \frac{ab(b^2x^2+a^2)\sin^2 x dx}{(b^2x^2+a^2+1)^2x^2(x^2/\pi^2-1)^2} \tag{10.53}$$

$$a_1 = \gamma - \sqrt{\gamma^2-\delta^2} \tag{10.54}$$

$$a_2 = \gamma + \sqrt{\gamma^2-\delta^2} \tag{10.55}$$

$$a_3 = \gamma + \sqrt{\gamma^2+\delta^2} \tag{10.56}$$

$$b = (k_{\parallel}/\lambda_s)\cos\phi \tag{10.57}$$

$$\delta = \sqrt{2m_c\omega_{LO}/\hbar\lambda_s^2} \tag{10.58}$$

The process of numerical calculation is similar to that for carrier-carrier scattering. In the same manner, carrier-phonon scattering also occur for holes in the valence band corresponding to the process in the conduction band shown in Figure 10.4. The spectral broadening factor in the valence band caused by the hole-LO phonon scattering is obtained by replacing the suffix c with v in the above equations. At the same time, the subband suffix together with the suffix v should be j, other than i.

Equations (10.33) and (10.49) are two basic formulas for calculating the broadening factors with respect to the carrier-carrier and carrier-phonon scattering, respectively.

Please note that two new parameters comes into the carrier-phonon scattering. One is the optical dielectric constant ϵ_∞ and the other is the longitudinal optical phonon energy $\hbar\omega - LO$. As an indication, we use $\hbar\omega_{LO} = 34.E-3$ eV and $\epsilon_\infty = 11.4$ for InGaAsP/InP system, and $\hbar\omega_{LO} = 46.E-3$ eV and $\epsilon_\infty = 10.4$ for GaAs/AlGaAs system. For accurate setting of these two parameters, please use the **tau_model** statement.

10.2.4 Nomenclature

We list the symbol definitions separately here due to the large number of new symbols in this section.

$D_c(E), D_v(E)$	Energy spectra of electron and hole, which have energy E.
E_{cj}, E_{vj}	Quantized energy levels in conduction and valence bands.
$E_{cjk_\parallel}, E_{vjk_\parallel}$	Energies of electron and hole under the optical transition at the subband j.
E, E_{cv}	Photon energy and energy difference between conduction and valence bands under the optical transition.
E_{fc}, E_{fv}	Quasi-Fermi levels of conduction and valence bands.
E_g	Bulk bandgap energy of semiconductor.
e	Electronic charge.
$\epsilon, \epsilon_\infty, \epsilon_o$	Static dielectric constant and optical dielectric constant and dielectric constant of conductor in vacuum.
$f_c(), f_v()$	Fermi distribution functions in conduction and valence bands.
g_c, g_v, g_{cv}	Step-like-densities-of-states of electron (c) and hole (v) and the step-like-density-of-states electron-hole pair.
$\Gamma_c, \Gamma_v, \Gamma_{in}$	Half broadening factors in conduction (c) and valence (v) bands, and average broadening factor, their relationship is $\Gamma_{in} = \Gamma_c + \Gamma_v$.
$\hbar$	Plank constant.
k	Boltzmann constant.
$k_\parallel, k_\perp$	Wave vectors parallel and perpendicular to the well interface.
$k_{f_\parallel}$	Component of the Fermi wave vector parallel to the well interface.
$L()$	Lorentzian line shape function for optical gain broadening.
L_w	Well width.

L_{cj}, L_{vj}, L_e	Effective well widths at the jth levels in conduction (c) and valence (v) bands, and the minimum of the four effective well widths.
λ_s	Inverse screening length.
m_0	Free electron mass.
m_c, m_v, m_{cv}	Effective masses of electron (c) and hole (v) and reduced effective mass between electron and hole, and their relationship is $1/m_{cv} = 1/m_c + 1/m_v$.
μ_o	Permeability of semiconductor of the vacuum.
n_r	Refractive index of semiconductor.
N	Injected electron density.
P	Injected hole density.
ω, ω_{LO}	Angular frequencies of light wave and longitudinal optical (LO) phonon wave.
$\psi' s$	Electron wave functions.
$\Phi' s$	Complex amplitudes of the electron wave functions.
$r, r_\parallel$	Position vector and the element of position vector parallel to well interface.
$< R_{cv}^2(E_{cv}) >$	Averaged square of electric dipole moment.
S	Interface area of sample.
T	Absolute temperature.
$\tau_c, \tau_v, \tau_{in}$	Intraband relaxation times in the conduction (c) and valence (v) bands and the total intraband relaxation time (in), and their relationship is $2/\tau_{in} = 1/\tau_c + 1/\tau_v$.
$\hbar/\tau_c, \hbar/\tau_v, \hbar/\tau_{in}$	Broadening factors caused by carrier-carrier and carrier-LO phonon scattering in conduction (c) and valence (v) bands, and the total broadening factor including summation over the conduction and the valence bands.
u	Periodic part of the bulk Bloch function.
V_{cn}	Matrix elements of the interaction in conduction (n=c) and valence (n=v) bands.
V_{in}	Injection potential. Obviously eV_{in} is injected carrier energy.

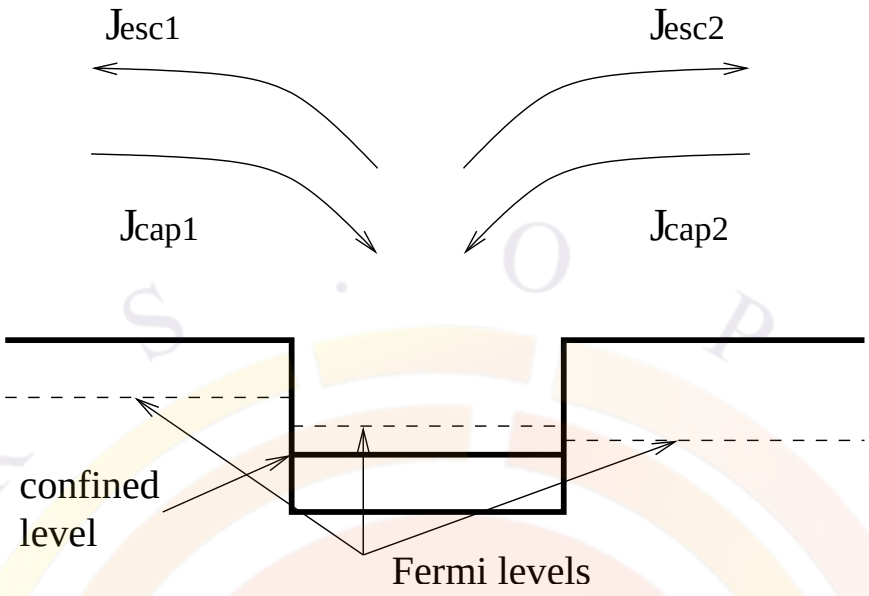


Figure 10.5: Schematics of capture and escape currents in real coordinate space.

10.3 Quantum Well Capture and Escape Processes

10.3.1 Carrier Capture and Escape in Real Space

As described in previous sections, the carrier transport across a heterojunction is via thermionic emission. For a quantum well, the carriers have to go across two heterojunctions (or two interfaces) via the same thermionic emission mechanism.

Some researches prefer to describe carrier transport across a quantum well in terms of "carrier capture" and "carrier escape" [69]. We shall also describe the carrier transport across a quantum well in the same technical terms.

In schematic in Fig. 10.5, we describe the events in real space (or in spatial coordinates). We denote the current components in term of capture and escape processes. In thermionic theory, the capture components I_{cap1} and I_{cap2} do not experience any scattering by the quantum well interfaces (in semi-classical approximation) and can be written as

$$J_{cap1} = \gamma_{12n} \overline{v}_{1n}^{therm} n_1, \tag{10.59}$$

$$J_{cap2} = \gamma_{23n} \overline{v}_{3n}^{therm} n_3, \tag{10.60}$$

where $\overline{v}_{1n}^{therm}$ is the thermal velocity for carriers in region 1 (left barrier) and n_1 is the carrier density there. γ_{12n} is a correction factor due to the semi-classical approximation (for neglecting quantum well potential scattering). Similar definition applies to the right barrier.

The computation of escape components are less straight forward because the carriers with energies less than the barrier are strongly reflected by the barrier. However,

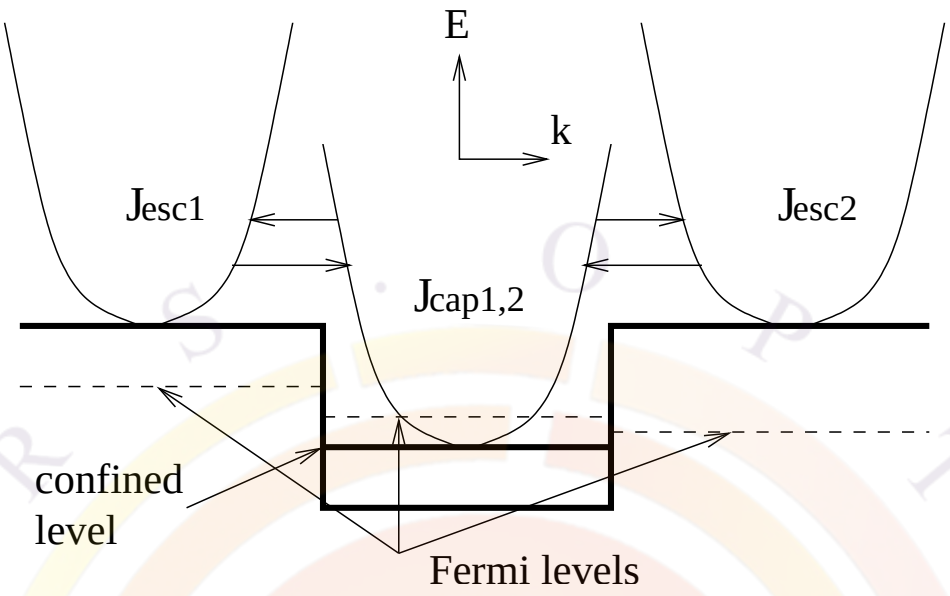


Figure 10.6: Schematics of capture and escape currents in energy-wave vector coordinate space assuming elastic scattering.

we make the observation that the escape component must balance out the capture component if Fermi levels on both sides are equal (such is the case in equilibrium). Thus, the escape components can be written as:

$$J_{esc1} = \gamma_{12n} \overline{v}_{1n}^{therm} n_{10}, \tag{10.61}$$

$$J_{esc2} = \gamma_{23n} \overline{v}_{3n}^{therm} n_{30}, \tag{10.62}$$

where n_{10} is the carrier density on the left barrier with Fermi level set to equal to that in the well (region 2). Similar arguments apply to J_{esc2}.

10.3.2 Carrier Capture and Escape in k-Space

The capture/escape event in the thermionic emission model above can also be described in energy space. The explanation of thermionic emission capture/escape model within the frame work of quasi-equilibrium (i.e., use of quasi-Fermi levels) is as follows. A carrier transfers with the same kinetic energy (elastic process) between confined and unconfined states with thermal velocity. After the transfer, it is instantly thermalized with local temperature and becomes part of the local carrier distribution described by the a quasi-Fermi level.

Similar to Ref. [69] we assume that transition of carriers between unconfined states and confined states have the same initial and final energies (see Fig. 10.6). The capture current can be written as follows in the notation of Ref [69].

$$J_{cap} \propto \int_{E_c^b}^{\infty} \rho_u(E) f(E, E_{fn}^u) \rho_c(E) [1 - f(E, E_{fn}^c)] dE \tag{10.63}$$

where u and c are for unconfined and confined states. E_c^b and E_{fn} are the energy at the barrier and Fermi level, respectively. $\rho(E)$ is the density of states.

Usually, the Fermi level is below barrier and we can safely set

$$[1 - f(E, E_{fn}^c)] \approx 1 \tag{10.64}$$

We also know that 2D density of states $\rho_c(E)$ is a constant in E. Thus the capture current can be written as

$$J_{cap} \propto \int_{E_c^b}^{\infty} \rho_u(E) f(E, E_{fn}^u) dE = n_u \tag{10.65}$$

which is just the unconfined carrier density at the barrier. This is consistent with Equations 10.59 and 10.60.

Once we have the expressions for the capture components, the escape components can be obtained using the argument that the total current flow must be zero if Fermi levels are flat.

In conclusion, the carrier capture/escape model in our simulator is that of elastic carrier scattering process followed by instant local thermalization which has been reported to agree with experiment (see Ref. [69]).

10.4 Dynamic Behavior of Quantum Capture and Escape

It is generally believed that the dynamic behavior (i.e., high speed modulation characteristics) of a laser diode depends on how fast carriers can be captured from unconfined states into confined states of the quantum well [70]. In a microscopic theory of quantum capture, energy distributions in both energy and space can be solved to give a complete description of the capturing process due to carrier-phonon interactions. The capturing process can be described by the following characteristic capture time τ_{cap} [70]:

$$\frac{1}{\tau_{cap}} = \int_{E_{final}} S_{3D,2D} \frac{g_{2D}}{L_{norm}} dE_{2D} \tag{10.66}$$

where E_{2D} is the 2D energy, E_{final} the energy of the final states, $S_{3D,2D}$ the scattering probability between 3D and 2D states, g_{2D} the density of state in the well, and L_{norm} the normalization distance of the confined wave function. Detailed carrier-photon interaction must be taken into account to evaluate $S_{3D,2D}$ above. Solving

the microscopic energy-space equations also produce the spectral hole burning effect which were thought to further limit the modulation band width.

We consider the dynamic effect associated with the quantum capture within the thermionic capture/escape model. We show that it is possible to compare the phenomenological carrier capturing theory in our simulation software with the more complex microscopic energy-space equation model. We shall derive the characteristic capturing time as follows.

Consider a simplified version of our carrier transport model for electrons in a quantum well. The simplified continuity equation with thermionic emission can be written as

$$\frac{dN_w}{dt} = \gamma \frac{v_{th}^{(b)}}{d_w} N_b - \gamma \frac{v_{th}^{(w)}}{d_w} N_w - \frac{N_w}{\tau_a} \tag{10.67}$$

where N_w and N_b are electron concentrations in the well and at the barrier, respectively. d_w is the quantum well width. $v_{th}^{(w)}$ and $v_{th}^{(b)}$ are the mass-dependent thermionic velocities in the well and at the barrier, respectively. τ_a is the interband recombination life time due to radiative, stimulated, SRH, and Auger recombination effects. γ is the heterojunction capture coefficient used to modify the thermal velocity.

Suppose we apply a pulse to the injection current which causes the barrier concentration N_b to behave like a step function. We wish to see how the quantum well carriers respond. Solving Eqn. (10.67) shows that the carriers in the well undergo the following transient behavior:

$$exp\{-[\gamma v_{th}^{(w)}/d_w + 1/\tau_a]t\} \tag{10.68}$$

Since the $1/\tau_a$ term is about five orders smaller and can be ignored, we find that the characteristic capturing time in our simulator can be expressed simply as

$$\frac{1}{\tau_{cap}} = \frac{\gamma v_{th}^{(w)}}{d_w} \tag{10.69}$$

Comparison of Equations (10.69) and (10.66) leads us to conclude that the the heterojunction capture coefficient γ is proportional to the microscopic scattering probability between 3D and 2D states, and thus can be used as a convenient parameter to measure the deviation of our thermionic model from the more complex microscopic energy-space model. The theoretical analysis of hole capturing is completely similar and shall be omitted here.

We consider a numerical example of idea thermionic emission with $\gamma = 1$. For a GaAs quantum well of 100 Å, we find that the electron and hole capturing times are 0.1 and 0.3 ps, respectively. These values seem to be reasonable compared to other estimates (see for example, [70]). As we pointed out in the last subsection that thermionic emission model involves the assumption of instant local thermalization,

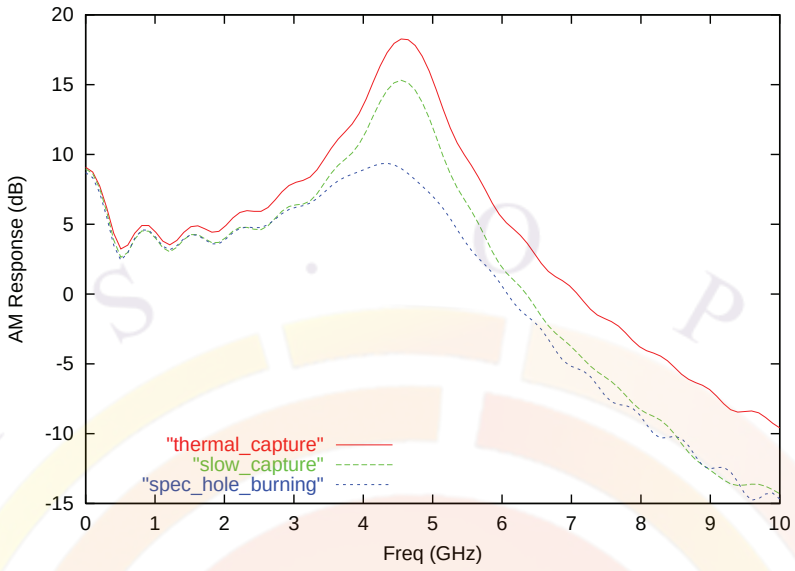


Figure 10.7: Numerical simulation of the modulation response of a 76 Åsingle QW GaAs/AlGaAs laser diode. The capturing speed is reduced by a factor of 0.01 and the non-linear gain constant is set to $5 \times 10^{-23}\ m^3$.

it is no surprise that these values are on the fast side. The actual capturing process should be smaller (with $\gamma < 1$).

Having resolved the carrier-capturing dynamics, we consider the spectral hole burning effect. In our simulator, this is expressed as a factor multiplied to the material optical gain:

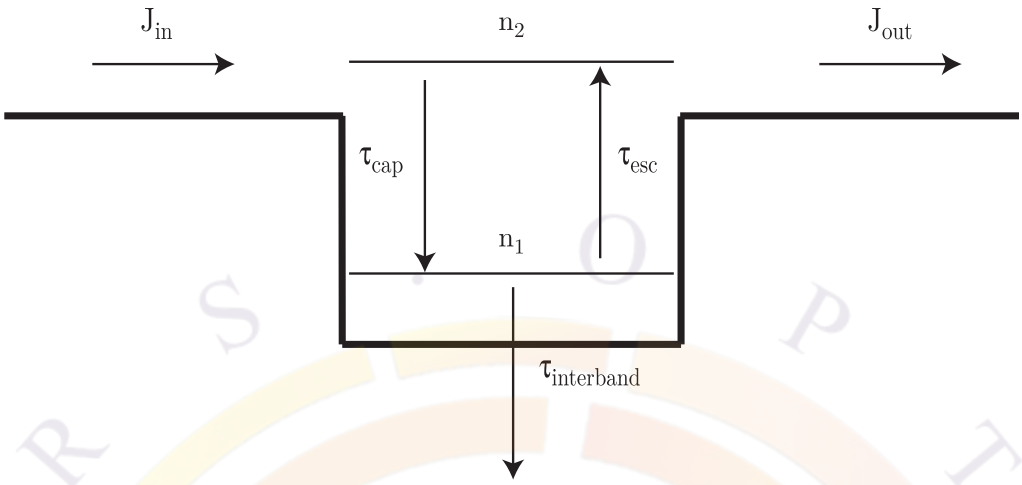
$$g(\omega, S) = \frac{g_0(\omega)}{1 + \epsilon_{nl} S} \tag{10.70}$$

where S is the bulk photon density and ϵ_{nl} is an empirical constant also called non-linear gain saturation constant.

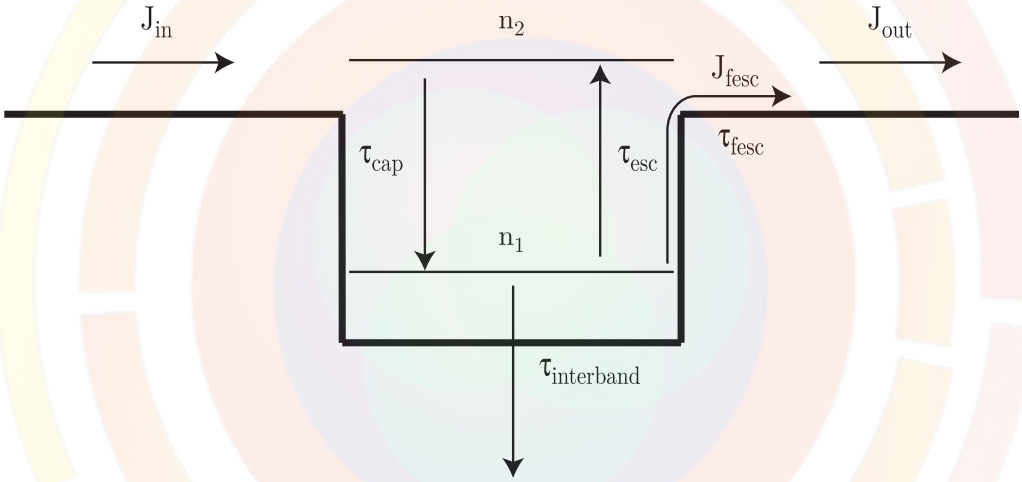
To show that both the capturing and spectral hole burning effects are important for the modulation band width, we use the Fourier transform of a small-step transient simulation to produce the modulation response in Figure 10.7. We find that spectral hole burning has similar effect as a slower capturing time.

10.4.1 Direct Flying-over Transport Model

When quantum wells (QW) or quantum dots (QDOT) are much smaller than the mean free path of the carriers (typically several nm), there is large chance the carriers directly fly over the QW or QDOT before being thermalized with carriers inside them. Given the mean free path λ_n for electron estimated from the conductivity (Drude model) near the QW/QD, we can write the probability of flying-over the QW/QD



(a) Quasi-equilibrium transport model



(b) Non-equilibrium transport model

Figure 10.8: Schematic of Quasi- and Non-equilibrium transport model

as $exp(-D_{qw}/\lambda_n)$. This model may be enabled using the **q_transport** command. Similar models can be enabled for holes.

10.4.2 Fractional Density Quantum Well/Dot Escape Theory

Crosslight also implements a new non-equilibrium carrier escape scheme to describe the phenomenon that the injected carriers escape from the quantum dots (QDOTs) or the quantum wells (QWs) before being thermalized with the local IMREF. A fraction of total carrier density in QWs or QDOTs is used to represent the density of hot carriers escaping from QWs or QDOTs. The theory is based on Ref.[71].

Assuming that the carrier capture/escape processes in QWs or QDOTs can be described by the following two-level quantum capture/escape model in the traditional quasi-equilibrium transport theory as shown in Fig.10.8(a), where n_1, n_2 are the density of carriers on the two levels. J_{in}, J_{out} are the injection and outflow density of carriers. τ_{cap}, τ_{esc}, τ_{interb} are the carrier lifetime of capture by the confined level, escape and interband recombination from the confined level, respectively. Then, the rate equation of carriers in QWs or QDOTs in the quasi-equilibrium transport model is

$$\frac{dn_t}{dt} = -\frac{n_t}{\tau_{interb}} + J_{in} + J_{out} \tag{10.71}$$

where $n_t = n_1 + n_2$. In our non-equilibrium carrier escape scheme, a fraction of the confined carriers are free to go directly via average conductivity with average local velocity: V_{fesc}. Then the carrier transport process can be shown as in Fig.10.8(b), the rate equation of carriers in QWs or QDOTs is

$$\frac{dn_t}{dt} = -\frac{n_t}{\tau_{interb}} + J_{in} + J_{out} - J_{fesc} \tag{10.72}$$

where J_{fesc} is the carrier density escaping from wells or dots before being thermalized, given by

$$J_{fesc} = \frac{F_{esc} n_t V_{fesc}}{d} \tag{10.73}$$

where F_{esc} is the fraction of total carrier density escaping from the wells or dots. d is the thickness of the wells or dots.

According to Ref.[72], the lifetime of carriers escaping before being thermalized can be determined by a similar form:

$$\tau_{fesc} = \frac{d}{V_{fesc} F_{esc}} \tag{10.74}$$

where $V_{fesc} = 2\pi\hbar/dm^*$, m^* is the effective mass. If $F_{esc} = 0.1$, $d = 3nm$, $m^* = 0.1$, then $\tau_{fesc} = 12fs$, which is close to the quantum well tunneling escape time as shown in Ref.[72]. This means that our fractional density escape scheme is equivalent to the traditional time constant escape scheme.

PARS OPTICS GROUP

Chapter 11

THERMAL EFFECTS IN SEMICONDUCTORS

This chapter deals with thermal effects in semiconductors. As we will show, there are a variety of heating sources in semiconductors: it is critical that these sources be accurately formulated when modeling devices where self-heating is important. Examples of such devices are high power transistors and laser diodes.

11.1 Temperature dependent parameters

A successful thermal simulation requires accurate temperature-dependent parameters. Typically, the most important parameters that need to vary with the temperature are bandgap, mobility, recombination coefficients and refractive index.

However, the software makes no assumption as to which parameter should be affected by the temperature. Instead, it allows all material parameters to be a function of the temperature. To do so, the parameter definition in the material macro should be a function of the reserved "temper" variable. This will tell the software to use the local temperature at any given mesh point to evaluate the function and get the parameter value.

Note that not all the macros in the database have thermal effects included. The user is advised to review the material parameters he plans to use and modify them as required. In particular, the refractive index is seldom defined as function of temperature in the default macros and this effect may be important for thermal lensing in lasers.

11.2 Isothermal temperature variation

Crosslight software allows the user to vary the isothermal temperature of a simulation. This is the temperature that is used during the initial equilibrium calculations and during the rest of the simulation if self-heating is neglected. Note that even with self-heating neglected, the material parameters will still be evaluated at the local mesh temperature (which is the same as the isothermal temperature).

11.3 Heat flow and temperature distribution

11.3.1 Expression of heat flux

It is well known that the heating effect is very important for semiconductor lasers in almost all applications. The heat generation often limits what a designer can do with a passively cooled device and the maximum power that can be reached.

From the point of view of simulation and modeling, the concern is two-fold. First, we must find out the temperature distribution from all possible heat sources. This involves a much larger simulation area than the small region near the p-n junction. We must consider how the heating power flows though the whole substrate as well as from any wire bonds. Second, we must consider how the heating affects the laser performance. This means we must accurately evaluate the degradation of power, efficiency and other parameters due to the the non-uniform temperature distribution. This is not a trivial task because virtually all variables and material parameters are temperature dependent. Our goal is to provide a thermal modeling environment so that all possible sources of temperature dependence can be taken into account.

We are concerned with the generation and flow of lattice heating power in a semiconductor. We first consider the heat flux and then the heat source. We introduce the thermal conductivity such that the heating power flux (in Watt/m^2) is given by:

$$J_h = -\kappa \nabla T \tag{11.1}$$

Conservation of energy requires that the temperature distribution satisfy the following basic thermal equation:

$$C_p \rho \frac{\partial T}{\partial t} = -\nabla \cdot J_h + H \tag{11.2}$$

or

$$C_p \rho \frac{\partial T}{\partial t} = \nabla \cdot \kappa \nabla T + H \tag{11.3}$$

where C_p is the specific heat and ρ is the density of the material. H is the heat source.

11.3.2 Thermoelectric power and thermal current

The temperature gradient induces a current of the form [73] [74] [75]:

$$-q\mu_n n P_n \nabla T \tag{11.4}$$

to the electron current and

$$-q\mu_p p P_p \nabla T \tag{11.5}$$

to the hole current.

The thermoelectric powers, P_n and P_p are given by:

$$P_n = \frac{k_B}{q}\left[-\frac{5}{2} - \nu + ln\left(\frac{n}{N_c}\right)\right] \tag{11.6}$$

and

$$P_p = \frac{k_B}{q}\left[-\frac{5}{2} - \nu + ln\left(\frac{p}{N_v}\right)\right] \tag{11.7}$$

where ν is the exponent used in the field-dependent relaxation time. For phonon scattering [76], $\nu = -1/2$

11.4 Heat sources

The heat source can be separated into contributions from Joule heat, generation/recombination heat and Thomson/Peltier heat. Following the suggestion of Ref. [73] we discuss these terms in more detail in the following subsections.

11.4.1 Joule/optical heat

There are two sources of Joule heating. One is from the steady state or low frequency part of the electrical field:

$$H_{Joule-dc} = \frac{J_n^2}{q\mu_n n} + \frac{J_p^2}{q\mu_p p} \tag{11.8}$$

The other part comes from the optical frequency (optical heat). When the optical wave passes a lossy semiconductor, the wave is absorbed by the lossy material. The absorbed energy can either generate electron hole pairs or be dissipated and become Joule heat. We also refer to this type of heat source as the optical part of the Joule heating.

The internal loss of a semiconductor laser is the cause of the optical part of the Joule heating while the band to band absorption is the source for electron-hole pair

generation in photo-detectors. A simple derivation of this term follows. The optical power dissipated per unit volume can be expressed as:

$$power\ loss = \sigma_{op} F_{op}^2 = \epsilon_2 \epsilon_0 \omega F_{op}^2. \tag{11.9}$$

where the optical field F_{op} is the root mean square of the oscillating field.

For convenience a constant γ is introduced such that the complex wave amplitude is related to the electric field by:

$$\mid F \mid^2 = \gamma \mid W \mid^2 . \tag{11.10}$$

To determine γ we use the following basic relation:

$$S\hbar\omega = \int \epsilon_0 \epsilon_1 \mid F \mid^2 dv = \epsilon_0 \gamma \int \epsilon_1 \mid W \mid^2 dv = \epsilon_0 \gamma < \epsilon_1 > . \tag{11.11}$$

The power loss becomes:

$$power\ loss = S\hbar\omega^2 \epsilon_2 \mid W \mid^2 / < \epsilon_1 > . \tag{11.12}$$

or in terms of material internal loss and local index:

$$\epsilon_2 = \overline{n}_1 \alpha_i / k_0 \tag{11.13}$$

$$H_{Joule-op} = S\hbar\omega^2 \overline{n_1} \alpha_i \mid W \mid^2 /(k_0 < \epsilon_1 >). \tag{11.14}$$

11.4.2 Recombination heat

When an electron-hole pair recombines, the energy either converts to a photon (radiative) or turns into heat (non-radiative). For the purposes of determining the locally generated heat, radiative recombinations are considered as a heat term but are partially canceled out by other terms (see below).

For each electron-hole pair recombined, the heat released is the difference between the quasi-Fermi levels:

$$H_{rec} = (R_{trap} + R_{Aug} + R_{spon} + R_{stim})(E_{fn} - E_{fp}) \tag{11.15}$$

An additional term is also needed to account for recombination events above the Fermi level. This contribution is related to the thermoelectric power by:

$$H_{rec2} = qR_{total} T (P_p - P_n) \tag{11.16}$$

where P_p and P_n are thermoelectric power for hole and electrons, respectively.

11.4.3 Radiative heat

Photons which escape the cavity remove heat from the device (radiative cooling); this defines a negative radiative heat term:

$$H_{rad} = -(R_{stim} + R_{spon})\hbar\omega \tag{11.17}$$

In the above, it is assumed that all photons which are emitted at a specific mesh point remove heat locally. This contribution is very close in magnitude to the radiative recombination terms in the recombination heat above since $E_{fn} - E_{fp} \approx \hbar\omega$.

Photons which are trapped in the cavity leave their energy elsewhere in the device; this is the optical heat term defined in previous sections. Therefore, the total amount of radiative cooling **is not** equal to the optical output power of the device.

In most real devices, some photons emitted by spontaneous emission do not couple into a mode but still manage to escape the cavity. This may be accounted for in future versions of the software by a scaling factor to R_{spon}.

11.4.4 Thomson/Peltier heat

The Thomson and Peltier heat terms are related to the spatial variation in the thermoelectric power:

$$H_{Pt-Th} = -T(J_n \nabla P_n + J_p \nabla P_p) \tag{11.18}$$

This heat source can be divided into its individual components by isolating the different contributions to the spatial gradient:

$$\nabla P_n = \frac{\partial P_n}{\partial T}\nabla T + \frac{\partial P_n}{\partial n}\nabla n \tag{11.19}$$

with a similar equation for the hole thermoelectric power.

The Thomson effect is thus due to the change in local temperature while the Peltier effect comes from the change in the local carrier concentration.[1]

11.5 Thermal boundary

We need to establish some realistic boundary conditions for the heat transport equation. In our simulator they are separately defined for electrical contacts and other parts of device.

[1] Prior to v.2016, the Thomson heat source was not correctly calculated and reported values that actually referred to H_{rec2}. Also prior to that version, the Peltier heat source referred to the combined Peltier-Thomson term.

For electrical contacts, they are defined in the **contact** statement:

- 1) Thermal contact type 1. The contact is a heat sink assumed to have a lattice temperature given by the parameter **lattice_temp**.
- 2) Thermal contact type 2. The contact is assumed to have an out-going heat power flow given by the parameter **heat_flow**. The lattice temperature of the contact is unknown and is to be determined by the thermal equation solver.
- 3) Thermal contact type 3. The contact is assumed to be connected to a thermal conductor with its conductance given by parameter **thermal_cond**. This thermal conductor is connected to a heat sink with a fixed temperature given by parameter **extern_temp**. Again the lattice temperature of the contact is unknown and is to be determined by the thermal equation solver.

  Please note that due to recent improvements in the external circuit model (**minipsice**, self-heating due to contact resistance is no longer handled automatically by the software. Instead, the heat flow due to the external resistor should defined using the **contact_heating** command.

If the user used a .layer file to facilitate the input, then contact statements may have been generated in the .mater file. As these default contacts will use a default boundary condition of Type 1 and a 300K lattice temperature, it is strongly recommended that all thermal simulations re-issue the contact declarations in the .sol to define the correct thermal boundaries.

For other parts of a simulated device, the heat flow follows the same rules as the current flow and no heat can flow across the mesh boundary. To override this and define non-contact thermal boundaries, use the **thermal_interf** statement: the same boundary types (1-3) apply.

Chapter 12

WAVEGUIDE OPTICAL MODES

12.1 Introduction

Many optoelectronic devices simulated by Crosslight software contain a waveguide. It is thus necessary to solve the lateral optical modes of a waveguide. Since the optical modes couple to the drift-diffusion model through carrier-dependent refractive index, an efficient solution of the wave equations is critical.

12.2 Scalar Wave Equation For Lateral Modes

Our basic assumption in dealing with 2D simulation of optical wave is that it propagates along the z dimension with a factor $exp(j\beta z) = exp(jk_0^2 n_{eff} z)$ where n_{eff} is the complex effective index. The wave equation can then be reduced in dimension by variable separation.

When the waveguide dimension of an edge type of semiconductor laser is comparable to or larger than the wavelength, the lateral optical modes can be described by the scalar wave equation as follows:

$$\nabla^2 W(x,y) + k_0^2(\varepsilon(x,y) - n_{eff}^2)W(x,y) = 0 \tag{12.1}$$

where $\varepsilon(x,y)$ is the optical dielectric constant (not to be confused with the static dielectric constant).

Our basic task is then to find the effective index and the related optical mode $W(x,y)$. There are several numerical methods we use to solve the optical modes:

- SOR iterative method. Convenient for simple single confined mode; useable but somewhat outdated.

- Effective index method (EIM): quasi-2D model which is effective for devices with index varying slowly in one direction.

  The EIM method solves a single mode 1D wave equation in the y-direction for a serious of y-cuts. It then uses the eigenvalues from these y-cuts as the index distribution along the x-direction and solves a multimode wave equations in the x-direction.

- Enhanced effective index method (EEIM) which is an extension of EIM. It can be used to model radiative boundary.

- Analytical approximation by Gaussian function.

- User defined data file.

- Arnoldi restarted method which is a highly sophisticated direction eigenvalue solution method useful for multiple lateral modes.

12.2.1 Multi-lateral mode model

A semiconductor laser can be driven into multi-lateral mode operation at high current injection levels if higher order lateral modes are supported by the waveguide structure. The 2D module solves the wave equation for the lateral modes at every current bias and solves, self-consistently, the photon densities of the modes. An additional photon rate equation for each additional lateral mode is included. Note that the lateral modes are coupled directly though the drift-diffusion equation via the stimulated recombination term and through the nonlinear gain effect as described later in this chapter.

12.3 Vectorial Helmholtz wave equation

12.3.1 Basic Vectorial Wave Equations

In this section, we derive the vectorial wave equation following Ref. [77]. For a harmonic guided wave propagating in the positive z direction, we consider the vector fields in the follow notation:

$$\overline{E}(x,y,z,t) = (E_x,E_y,E_z)exp[j(\omega t-\beta z)] \quad (12.2)$$

$$\overline{H}(x,y,z,t) = (E_x,E_y,E_z)exp[j(\omega t-\beta z)] \quad (12.3)$$

$$\overline{D}=\varepsilon\overline{E} \qquad \overline{B}=\mu\overline{H} \quad (12.4)$$

From Maxwell's equations for source-free regions:

$$\nabla \cdot \overline{D} = 0, \qquad \nabla \cdot \overline{B} = 0 \tag{12.5}$$

$$\nabla \times \overline{E} = -\frac{\partial \overline{B}}{\partial t} = -j\omega\mu\overline{H} \tag{12.6}$$

$$\nabla \times \overline{H} = \frac{\partial \overline{D}}{\partial t} = j\omega\varepsilon\overline{E} \tag{12.7}$$

we obtain:

$$\nabla \times \nabla \times \overline{E} = \nabla(\nabla \cdot \overline{E}) - \nabla^2\overline{E} = \omega^2\mu\varepsilon\overline{E} = k^2\overline{E} \tag{12.8}$$

where $k = 2\pi n/\lambda$ is the wave number.

Thus,

$$\nabla \cdot \overline{D} = 0 \text{ or } \nabla \cdot (k^2\overline{E}) = 0 \tag{12.9}$$

$$\overline{E} \cdot \nabla k^2 + k^2 \nabla \cdot \overline{E} = 0 \tag{12.10}$$

$$\nabla \cdot \overline{E} = -(1/k^2)\overline{E} \cdot \nabla k^2 \tag{12.11}$$

Using the Maxwell's equations above, we obtain the vectorial Helmholtz wave equation:

$$\nabla^2\overline{E} + k^2\overline{E} + \nabla\left(\frac{\overline{E} \cdot \nabla k^2}{k^2}\right) = 0 \tag{12.12}$$

Let us consider the transverse component of the above equation with the notation:

$$\overline{E}_T = (E_x, E_y, 0), \quad \nabla_T = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, 0\right) \tag{12.13}$$

We note that in most cases, it is reasonable to assume that $\frac{\partial}{\partial z}(k^2) = 0$. We then obtain:

$$\nabla_T^2\overline{E}_T + (k^2 - \beta^2)\overline{E}_T + \nabla_T\left(\frac{\overline{E}_T \cdot \nabla_T k^2}{k^2}\right) = 0 \tag{12.14}$$

which is the vectorial wave equation.

The x component of the above is as follows:

$$\frac{\partial^2 E_x}{\partial x^2} + \frac{\partial^2 E_x}{\partial y^2} + (k^2 - \beta^2)E_x + \frac{\partial}{\partial x}\left(\frac{E_x}{k^2}\frac{\partial k^2}{\partial x} + \frac{E_y}{k^2}\frac{\partial k^2}{\partial y}\right) = 0 \tag{12.15}$$

$$\frac{\partial}{\partial x}\left(\frac{\partial E_x}{\partial x} + \frac{E_x}{k^2}\frac{\partial k^2}{\partial x}\right) + \frac{\partial^2 E_x}{\partial y^2} + (k^2 - \beta^2)E_x + \frac{\partial}{\partial x}\left(\frac{E_y}{k^2}\frac{\partial k^2}{\partial y}\right) = 0 \tag{12.16}$$

Our final expression for the x component of the wave equation is as follows:

$$\frac{\partial}{\partial x}\left(\frac{1}{k^2}\frac{\partial}{\partial x}(k^2 E_x)\right) + \frac{\partial^2 E_x}{\partial y^2} + (k^2 - \beta^2)E_x + \frac{\partial}{\partial x}\left(\frac{E_y}{k^2}\frac{\partial k^2}{\partial y}\right) = 0 \tag{12.17}$$

Similarly, the y component can be written as follows:

$$\frac{\partial^2 E_y}{\partial x^2} + \frac{\partial}{\partial y}\left(\frac{1}{k^2}\frac{\partial}{\partial y}(k^2 E_y)\right) + (k^2 - \beta^2)E_y + \frac{\partial}{\partial y}\left(\frac{E_x}{k^2}\frac{\partial k^2}{\partial x}\right) = 0 \tag{12.18}$$

Once the transverse components are solved, we can obtain the z component from the zero-divergence as follows:

$$E_z = \left(\frac{1}{j\beta k^2}\right)\nabla_T \cdot (k^2 \overline{E}_T) \tag{12.19}$$

12.3.2 Interpretation of TE Vectorial Solutions

In this subsection, we attempt to provide a simple prediction or interpretation of the vectorial wave solution in comparison with the scalar solution. We concentrate on the TE mode $((E_x, 0, E_z))$ which is used in most device designs.

We start by rewriting the vectorial wave equation equation (12.16) using $E_y = 0$ as follows:

$$\frac{\partial^2 E_x}{\partial x^2} + \frac{\partial^2 E_x}{\partial y^2} + (k^2 - \beta^2)E_x + \frac{\partial}{\partial x}\left[\frac{E_x}{k^2}\frac{\partial(k^2)}{\partial x}\right] = 0 \tag{12.20}$$

We find that the vectorial wave equation can be written in the same form as the scalar one if we introduce a "vectorial effective index" $k_{ve} = 2\pi n_{ve}/\lambda$.

$$\frac{\partial^2 E_x}{\partial x^2} + \frac{\partial^2 E_x}{\partial y^2} + (k_{ve}^2 - \beta^2)E_x = 0 \tag{12.21}$$

$$k_{ve}^2 = k^2 + \frac{1}{E_x}\frac{\partial}{\partial x}\left[\frac{E_x}{k^2}\frac{\partial(k^2)}{\partial x}\right] \tag{12.22}$$

Therefore, interpretation of the vectorial wave equation can be made using the scalar equation and the "vectorial effective index". Let us consider qualitatively the effect of vectorial wave using the schematic in Fig. 12.1. We consider only the case of continuous index profile because the physical conclusion is the same for abrupt index profile.

Based on Fig. 12.1, we decide that $\frac{\partial(k^2)}{\partial x}$ is negative in positions with index change. Since $E_x(x)$ decays in the positive x-direction, we conclude that Eq. (12.22) yields increased effective index in locations of index change. The increased effective index in positions of index change can be used to interpret or predict the effect of vectorial wave equations as compared with the scalar equation.

To demonstrate the effect of the increased index, we set up a series of scalar wave simulations as shown in Fig. 12.2. We artificially put an increased index in three

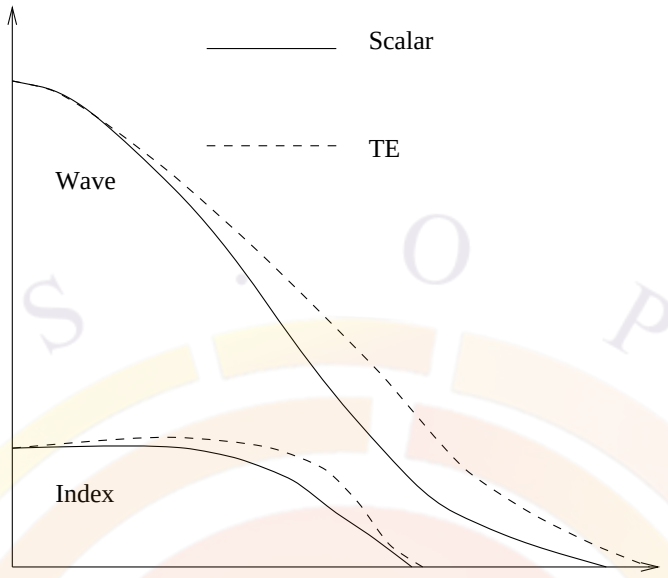


Figure 12.1: Schematics of the right half of the optical mode along the lateral direction. The solid lines are the refractive index and the scalar mode. The dashed lines are the "vectorial effective index" and the corresponding wave distribution.

different positions in the simulations. In a device with index change far from the center of the mode, the increased index of the vectorial term causes the wave to spread out more than scalar wave. If the index change is close to the mode center, more optical confinement is provided by the use of the vectorial wave.

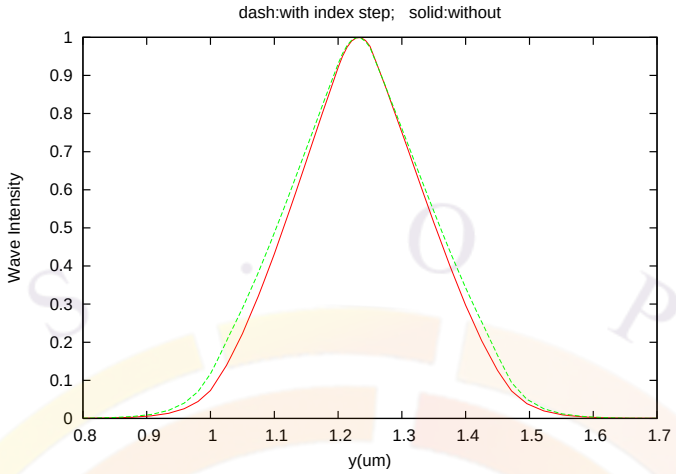
In most laser diode structures, the index change in the x-direction is usually located far from the mode center. For example, a typical buried heterostructure laser may uses a two micron wide active region width to provide the lateral electrical/optical confinement so the index change is about one micron away from the mode center. This means that the vectorial TE mode produces a mode which spreads out more in the x direction than the scalar mode.

We may therefore use the scalar mode solver in most situations: common laser designs operate in the TE mode and have a lateral index variation that is weak and/or far away from the mode center.

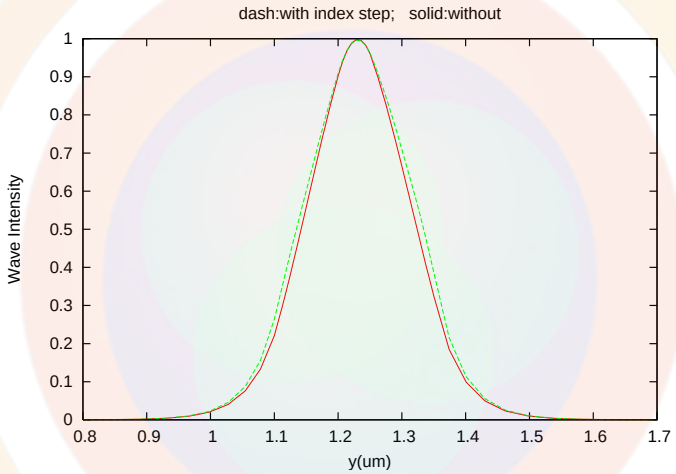
12.3.3 Interpretation of TM Vectorial Solutions

A similar derivation may be done for the TM $((0, E_y, E_z))$ vectorial mode by exchanging the x and y-axis. However, we must now consider the effect of index discontinuities in the y direction relative to the peak of E_y.

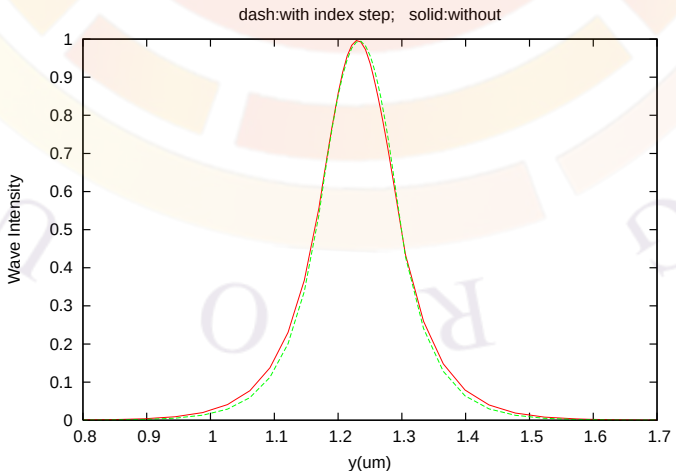
In most device designs, the optical mode is centered on a quantum well region which provides the optical gain: this means that the index discontinuity between the well



(a)

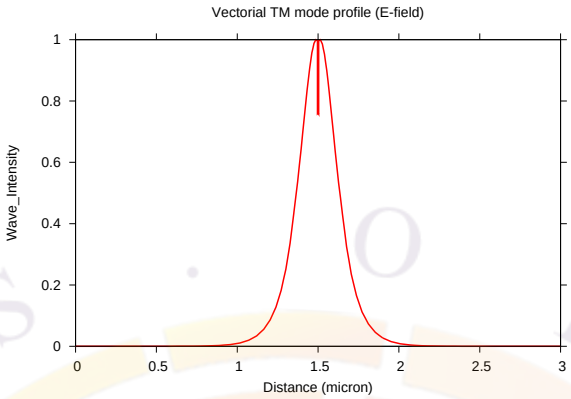


(b)

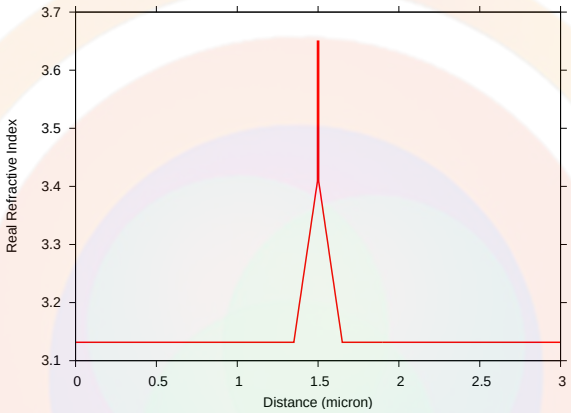


(c)

Figure 12.2: A simple laser structure used to understand the increased index caused by the vectorial term. The active layer is 0.05 μm GaAs. The wave guiding layer is 0.03 μm Al(0.2)Ga(0.8)As and the cladding layer is Al(0.8)Ga(0.1)As. The increased index step is placed at various distances from the active layer to simulate the effect of the vectorial term:a) 0.2 μm, b) 0.1 μm, c) 0.03 μm



(a)



(b)

Figure 12.3: A simple laser structure used to demonstrate the effect of the index discontinuity on the vectorial TM mode profile. The structure consists of a 7.6 nm GaAs QW with graded (71% to 33%) AlGaAs barriers.

and barrier is only a few nanometers away from the mode peak. This further means that the mode profile in E_y is discontinuous near the wave peak which produces a significantly lower mode confinement factor compared to a scalar wave solution, as can be seen in Fig. 12.3.

It is therefore recommended that all optical mode calculations for TM be done using the vectorial mode solver rather than the default scalar wave model.

We note in passing that many authors calculate the TM mode by using equations based on the magnetic field H. While this formulation produces a smooth wave profile due to the continuity of H_x at the QW boundary, it produces a different confinement factor and introduces additional terms in the calculations of the modal gain. In our model, we thus make the choice of using the simpler formulation based on E even though it produces discontinuous wave profiles for TM modes.

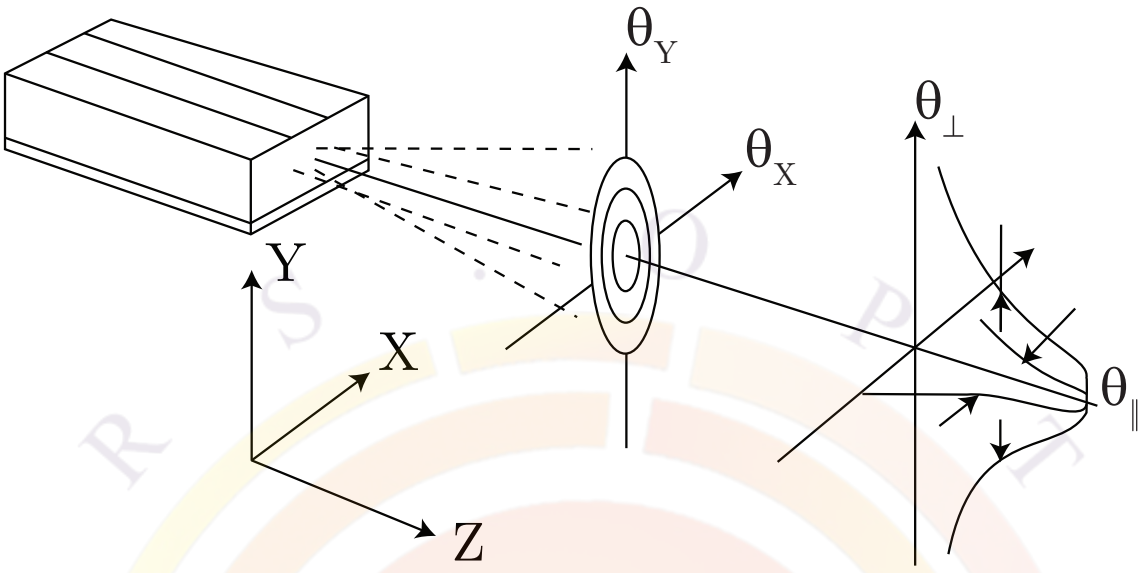


Figure 12.4: Schematic representation of the far-field emission of a semiconductor laser.

12.4 Far-field computation

After solving for the waveguide modes, it is straightforward to compute the far-field emission pattern. The problem is basically that of diffraction: *i.e.*, given the distribution of a propagating beam at a 2D cross section, calculate the distribution at a distance from this cross section. The formulation for this problem has been well established in classical electrodynamics and the user is referred to standard textbooks on the subject.

A semiconductor laser emits light in the form of a narrow spot of elliptical cross section. The spatial intensity distribution of the emitted light near the laser facet is known as the *near-field*. During its propagation, the spot grows in size due to beam divergence.

The dimensions of the elliptical spot and its divergence angles are important beam parameters associated with the laser mode. The *far-field* pattern is the angular distribution of beam intensity far from the laser facet. It is an important factor in the design of laser geometries since it directly affects the coupling of power from the laser source to the object being illuminated (e.g. an optical fiber).

Fig. 12.4 gives a schematic representation of the far-field emission of a semiconductor laser. The full angles at half power are $\theta_\perp$ and $\theta_\parallel$ in the directions perpendicular to and along the laser junction plane, respectively. Typically, $\theta_\parallel$ is of the order of 10°, whereas $\theta_\perp$ is considerably larger (35 to 65°), depending on the detailed near-field distribution.

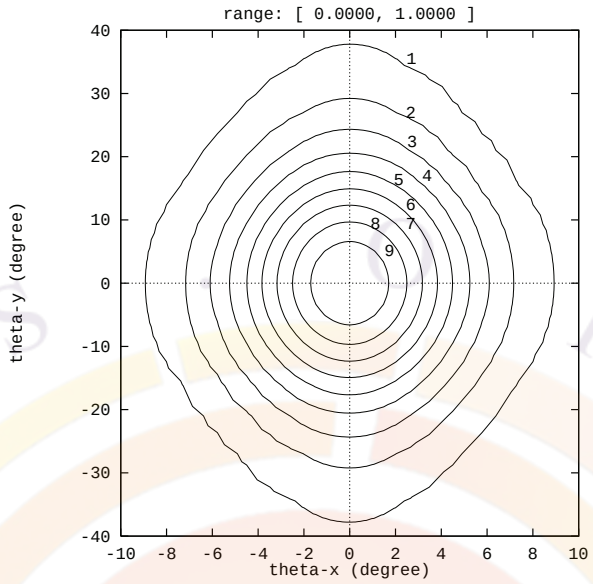


Figure 12.5: An example of the far-field emission pattern from a non-planar ridge waveguide GaAs/AlGaAs laser emitting at 0.78 μm.

The program solves the wave equation and the semiconductor equations self-consistently and yields accurate near-field distribution as part of the output data. The far-field model is given by an additional post-processing step by the software.

Refs. [1] and [34] give mathematical formulas for the far-field distribution based on the solution to the wave equation in free space. Since these mathematical expressions were derived for broad-area lasers only (*i.e.*, assuming the x-dimension is infinite), the program has extended their formulas to an arbitrary two-dimensional situation using the same solution approach as suggested in [1] and [34].

As an example, we consider a ridge waveguide GaAs/AlGaAs laser emitting at about 0.78 μm. The far-field pattern is computed based on the near-field solution and is presented in Fig. 12.5 as a contour plot of the far-field power. The contour lines are evenly spaced in power level (at 10% power intervals). The angles $\theta_\perp$ and $\theta_\parallel$ can be directly extracted from the contour plot.

12.5 Enhanced Effective Index Method

12.5.1 Introduction

In the standard effective index method (EIM), all boundaries are assumed to be zero or decay exponentially. As a result, radiative modes are excluded. In this section, we will describe an extension of the EIM which allows us to treat radiative modes

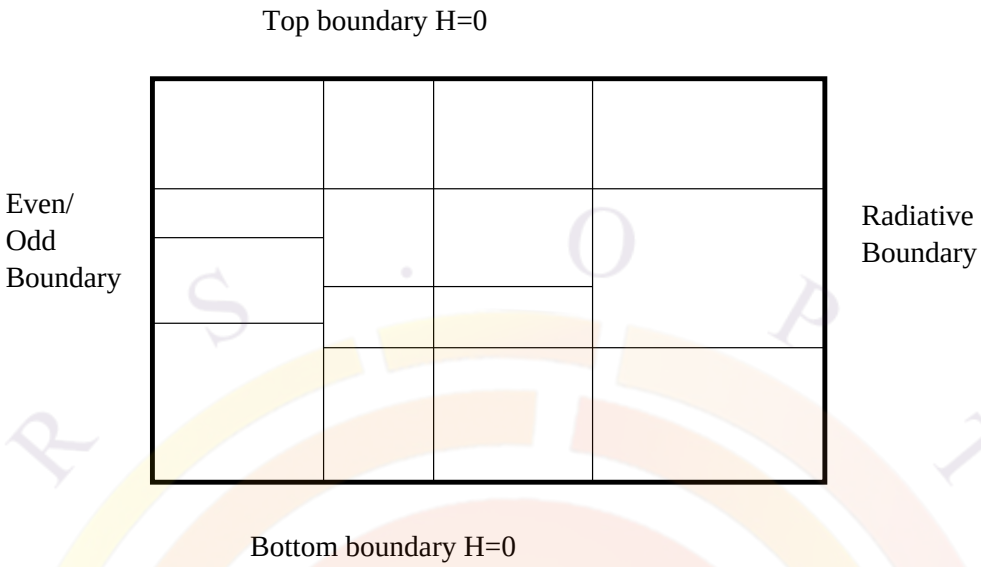


Figure 12.6: Boundary conditions and division of columns and rows in the enhanced effective index method (EEIM)

as well as confined modes.

The standard set up is as illustrated in Fig. 12.6. We follow the solution approach of Ref. [78] and assume the wave is TE-polarized. It is convenient to solve the H field in this case. The mode is assumed to be confined in y-direction, i.e., the top and bottom boundaries have zero field intensities. The left boundary is assumed to be either even (with derivative zero) or odd (with field zero). This is most suitable for device with symmetry axis place on the left of the simulation area.

The radiative boundary is assumed to be on the right side. This will be most convenient for cases such as ARROW waveguide type of lasers and other antiguiding lasers.

12.5.2 Rectangular model

For an arbitrary device structure, it is always possible to divide into columns and rows. The more divisions, the higher the accuracy that can be achieved (at the cost of computation time). The program is designed such that the user specifies the columns and the program determines the rows automatically. Usually, it is efficient to divide the structure into 2 or 3 columns according to the geometry of the device. For example, a ridge waveguide laser is best divided into two columns with boundary at the ridge wall.

According to Ref. [78], the basic wave equation for the TE mode can be represented

by the following magnetic field equation:

$$\nabla^2 H(x,y) = k_0^2(\overline{\varepsilon} - \varepsilon)H(x,y) \tag{12.23}$$

The horizontal boundary between row 1 and row 2 is:

$$H_1 = H_2; \frac{\partial H}{\partial y} \mid_1 = \frac{\partial H}{\partial y} \mid_2 \tag{12.24}$$

The vertical boundary between column 1 and column 2 is:

$$H_1 = H_2; \frac{1}{\varepsilon_1}\frac{\partial H}{\partial y} \mid_1 = \frac{1}{\varepsilon_2}\frac{\partial H}{\partial y} \mid_2 \tag{12.25}$$

In each vertical segment, only horizontal boundaries between dielectrics occur, and ε is now a step function of y. We separate the variable so that $H(x,y) = \alpha(x)\beta(y)$. The function β is a vertical eigenfunction for this vertical segment with eigenvalue δ satisfying the equation:

$$\frac{d^2\beta}{dy^2} = k_0^2(\delta - \varepsilon(y))\beta \tag{12.26}$$

α satisfies a similar equation:

$$\frac{d^2\alpha}{dx^2} = k_0^2(\overline{\varepsilon} - \delta)\alpha \tag{12.27}$$

Although there are, in general, an infinite number of eigenfunctions β for each vertical segment, we include only some finite number N of these to construct a general solution. The general solution in segment j can be written as:

$$H_j(x,y) = \sum_{i=1}^{N}[a_i exp(k_0\sqrt{\overline{\varepsilon} - \delta_i}x) + b_i exp(-k_0\sqrt{\overline{\varepsilon} - \delta_i}x)]\beta_i(y) \tag{12.28}$$

Our task is to determine the eigenvalue δ and function β in the first step, and $\overline{\varepsilon}$ and the coefficients of a_i and b_i in the second step.

12.5.3 Numerical approach

Our first step is to solve Eq. 12.26 within each column. The analytical expression for β is expressed in term of unknown δ. Assuming there are M layers, we have $2M$ number of unknown coefficients in this expression. The horizontal boundary condition gives a set of $2M$ equations that can be solved by setting the determinant to be zero. The zeros of the determinant is used to find the roots of δ. A singular

value decomposition (SVD) method is then used to determine the $2M$ coefficients. We shall call this procedure "searching for roots in y-direction" (y-root).

Zero boundary on top and bottom will produce a complete set of eigenvectors that can be used to construct any wave function. Thus, more β base functions result in a higher accuracy of the constructed wave function. However, the use of more base function in y-direction slows down the solution substantially. Here we will choose a limited number of based functions (β_i) where $i = 1, 2, ..., N$.

Once we have the β base functions, we can construct the final H-field as in Eq. 12.28. For each column j, the H-field consists of $2N$ coefficients of a_i and b_i. The vertical boundary conditions can be used to write down two equations for H-fields connecting H-field in j column with that in the $j + 1$ column. The boundary condition on the right boundary requires that traveling wave to the left be zero. This boundary condition can be used to model both radiating wave to the right or for confined wave with decaying wave.

If we multiply the two equations by $\beta_k(y)$ and integrate over y, we will get $2N$ equations relating columns j and $j + 1$. Using this technique for all columns will produce a sufficient number of equations to determine all the a_i and b_i coefficients and the modal eigensolution for $\bar{\varepsilon}$. We shall call this procedure "searching for modes in x-direction (x-root)" . This step generates the final results for the optical modes.

The modal optical dielectric constant can be used to determine the modal optical gain as follows:

$$g_{eeim} = -Im(\bar{\varepsilon})k0/n_{eff} \tag{12.29}$$

where $n_{eff} = Re(\sqrt{\bar{\varepsilon}})$ is the real part of modal effective index. We shall call this EEIM modal gain to differentiate it with the usual modal gain (approximated by averaging the local material gain with the weight of wave intensity). We shall refer to the latter as averaging modal gain.

The use of averaging modal gain has great numerical advantage of being stable against the Newton's method used by the drift-diffusion equations. But the averaging gain is inaccurate when there is radiative loss in the case of leaky modes. However, direct use of the EEIM gain will cause numerical problems near lasing condition.

As a compromise, we estimate the radiation loss at transparent condition by setting all imaginary parts of the dielectric constants to zeros and use the EEIM gain obtained this way as the radiation loss. We will use this estimated radiation loss to correct the averaging gain throughout the whole simulation.

In summary, we have been able to used the EEIM to estimate the radiation loss and use it to correct the averaging modal gain.

12.5.4 Estimate of radiative loss

Using the EEIM module, we are able to find the radiative modes and the effective index (or eigenvalue of the wave equation) associated with it. The modal loss/gain is represented by the imaginary part of the eigenvalue.

In many cases, we need to discuss device performance in terms of radiative loss and material loss separately. The issue here is how we can separate the radiative loss from the material loss since the imaginary part of the eigenvalue of the wave equation comes from both material and radiative losses.

We recall how we calculate the modal gain/loss in the absence of radiative loss. To a good approximation, we calculate the modal gain/loss by averaging the local material gain/loss using the wave intensity as the weight function. This approach simplifies and stabilizes the numerical procedures. Such method of wave intensity averaging cannot be extended to the case when there is radiative loss. The reason is that the wave extends to infinity where there is radiative loss.

In our simulator, we separate the radiative loss as the different between the total gain/loss and the material gain/loss from averaged local material gain/loss.

We use two methods to estimate the radiative loss. The first method of estimate is to set all material to be lossless (by setting imaginary part of indices to be zero). We then apply the EEIM method to compute the total loss with is just the imaginary part of the eigenvalue of wave solution. Since the material loss is artificially set to zero, the total loss from the EEIM is the radiative loss we needed. We refer to such a method of estimate as the "pure index estimate" since purely real indexes are used in the calculation.

An alternative method of radiative loss estimate is to compute at every bias both the total gain/loss from the EEIM and also the averaged local gain/loss. The difference between the two is the bias dependent radiative loss. Since the local complex index varies as a function of bias, the radiative loss also changes with bias. We shall refer to this method of estimate as the "bias-dependent estimate".

The "pure index estimate" is most convenient since we only get one value for each device. In most cases, it gives a good representation of the radiative losses in the same device when there are material losses.

The "bias-dependent estimate" is more realistic since the radiative loss depends also on local material gain/loss. Since the numerical fluctuation of EEIM as a function of bias can be quit large, the relative error from this method may also be substantial as compared with the pure index estimate.

To maintain numerical stability, we still use the method of averaging the local gain to obtain the modal material gain/loss. Then the radiative loss from either of the above estimate methods is subtracted from the material modal gain/loss to obtain the total modal gain/loss which is needed to compute the lasing characteristics.

We use the pure index method as our default setting the simulator. It is the most stable since we only have one number to subtract from the material gain. The numerical fluctuation of the bias-dependent method may be a problem when calculating the light vs. current characteristics so we freeze the value of the radiative loss when the total gain near threshold. In both methods, we freeze the optical wave function from the EEIM when the device is biased near the threshold to ensure a smooth light intensity as a function of bias.

12.5.5 Substrate radiative loss

A common simulation issue is to study the radiative loss caused by high refractive index substrate. So the direction of radiative wave is downwards in -y direction. Since our EEIM model assumes confined y-modes and radiative x-modes, we must first rotate our device counter-clockwise by 90 degrees. This may be achieved by use of GeoEditor GUI program or in the layer file.

12.5.6 Choice of optical window in y-direction

If the material high index in y-direction is always well centered, it does not matter what size of solution region you choose in y-direction (optical boundary in init_wave statement). However, there are cases where some columns have no optical confinements in y-direction at all.

For example, a ridge waveguide laser rotated by -90 degrees will have the substrate region with no optical confinement. In such as case, the base modes in y-direction depends very much on the choice of optical window. If the window size in y is too small, it obviously will affect the y-modes. If the size is too large, many y-modes will be needed to match the base modes from other columns with optical confinements in y. With a proper choice of optical window in y-direction, it is possible get reasonable modal solutions using only 3-5 y-base modes.

12.5.7 Run-time messages

The EEIM model is invoked at every bias step to ensure the variation of modes as a function of bias is taken into account. When using the EEIM model, the simulator prints out messages stating the status of the execution of the EEIM model procedures. Here we list the Run-time messages when EEIM is invoked for the first time as lines with > sign. Our comments are also given in the listing:

```
> -------Start EEIM Model--------
> With initial root search (this may take a while).
```

```
Without any previous knowledge of the roots (or modes),  we have
to search the complex delta (for y-modes) and epsilon (for x-modes)
using a point by point approach.   This may take a while.

> ---Estimate of radiation loss (pure index)---

As a first time use of EEIM in the program, we estimate
the radiation loss.  This is done by setting all imaginary parts
of the local index to zero.  The imaginary part of
epsilon (eigenvalue) is used to estimate the radiation modal
loss.

> Save base in "tmpb101.dat"
> Save base in "tmpb102.dat"
> Save base in "tmpb103.dat"
> Save base in "tmpb201.dat"
> Save base in "tmpb202.dat"
> Save base in "tmpb203.dat"
...
> Save base in "tmpb701.dat"
> Save base in "tmpb702.dat"
> Save base in "tmpb703.dat"

The y-base modes are computed and stored in "tmpbijk.dat",  where
i, j, k are integers for ith column and y-base mode number jk.   You may
use the following gnuplot command to view the y-mode:
plot  "tmpbijk.dat" w l.

Start to search for roots for the final optical field.

> Initial x-search found roots=           5

Found 5 possible roots for x-direction.

> Solve wave in x direction.
>   Find the following possible roots.
>No./Iter/Error=  1   80  0.7779E-02
>   root= (10.2363356873790,1.027566003195660E-003)
>     EEIM gain[1/m]=  -2374.09673169544
>   Save wave in "tmploss01.dat"
>No./Iter/Error=  2   80  0.6136E-03
>   root= (10.2118941158888,1.060170706117816E-003)
```

```
>     EEIM gain[1/m]=  -2452.35642688367
>   Save wave in "tmploss02.dat"
>No./Iter/Error=  3   80  0.1955E-04
>   root= (10.1705711295711,1.119772469334270E-003)
>     EEIM gain[1/m]=  -2595.48221708482
>   Save wave in "tmploss03.dat"

We have found the above 3 modes which are saved in "tmplossij.dat".
You may use the following gnuplot commands to plot modes
to see if these are physical:

set parametric
splot "tmplossij.dat" w l.

> Radiation loss for pure index model:
> Mode        Loss(1/m)
>          1   2374.09673169544

We estimate the radiation loss for the requested first mode as above.
This loss will be used to correct the material gain
to get the actual total modal gain.


> ---End of estimate of radiation loss---

Once we finished the "pure index estimate",  we proceed to
use EEIM model for optical modes are each bias with run-time
message similar to the above.

> Save base in "tmpb101.dat"
> Save base in "tmpb102.dat"
> Save base in "tmpb103.dat"
...

> -------End of EEIM Model--------
```

When the EEIM model is invoked in the next bias step, the roots found in the previous bias are used as initial guess. If the initial guess does not produce stable solution, the program will re-invoke the EEIM model again with initial root search. The initial root search usually takes up most of the time.

Please note that our simulator always gives the pure index estimate the first time EEIM is invoked. Then, the modes are solved at every bias steps. If pure index estimate is not used, radiative loss at each bias step is calculated and printed at

different bias steps.

12.6 Perfectly Matched Layer Boundary

The finite element (FEM) discretization of the wave equation results in an eigenvalue matrix problem of the order of the number of mesh points (for scalar equations). The matrix problem can easily be set up if the modes are well confined within a finite region and their intensities decay as a function of distance from the mode centers. We can obtain a well defined eigenmatrix by deleting the nodes of the boundary assuming zero wave intensities there. The situation for leaky modes is more complicated for finite element analysis: there is no simple way to truncate the mesh to establish a well defined eigenmatrix.

The idea of perfectly match layer (PML) is to create an artificial layer to absorb the radiating wave without reflecting it back. Since the wave decays to zero intensity within the PML, we have converted the problem into one of confined modes if our mesh extends to the outer boundary of the PML.

We have implemented a model of anisotropic PML due to Sacks and others [79]. In this model, the artificial PML region has a complex anisotropic dielectric constant. If this constant is chosen properly, the PML method should enable us to apply FEM to radiative mode solutions which are more accurate and powerful than EEIM.

We define an artificial layer with an anisotropic medium tensor as follows:

$$\frac{[\varepsilon]}{\varepsilon} = \frac{[\mu]}{\mu} = [\Lambda] = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \tag{12.30}$$

where ε and μ are permittivities of the isotropic medium to be matched.

Without loss of generality, we consider the situation as indicated in Fig. 12.7. As proved by Sacks et. al., the reflection from the PML may be eliminated if the permittivity tensor components follow the relationship $c = b = 1/a$.

As we show in Appendix A, the wave equation in the PML can be written as follows:

$$\left(\frac{1}{c^2}\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\overrightarrow{E} + k^2\overrightarrow{E} = 0 \tag{12.31}$$

Consider a scalar field being having the components:

$$E(x, y, z) = W(x, y)exp(j\beta z) \tag{12.32}$$

$$\left(\frac{1}{c^2}\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)W(x, y) + (k^2 - \beta^2)W(x, y) = 0 \tag{12.33}$$

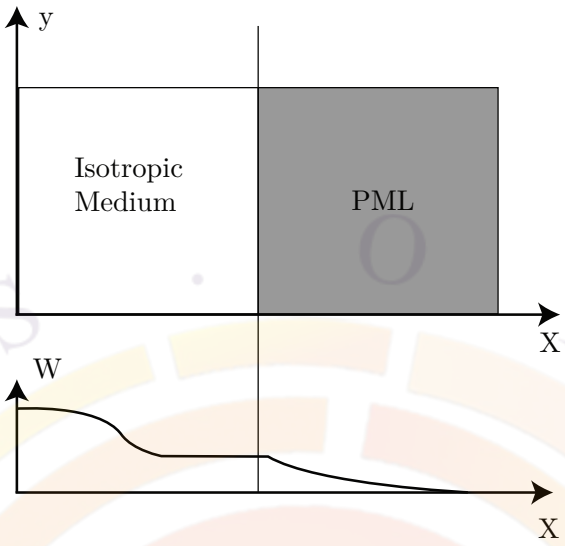


Figure 12.7: Schematics of the perfectly matched layer the decay of the wave within it.

Let us study the asymptotic behavior of the wave in the PML. Usually, the radiative wave occurs when the index near the PML is higher than the effective index, or :$Re(k^2 - \beta^2) > 0$. The wave behaves as:

$$W \sim exp(jc\sqrt{k^2 - \beta^2}) \tag{12.34}$$

Thus, the imaginary part of c will cause the wave to decay within the PML so that we can truncate our eigenmatrix on the outer boundary of the PML.

The criteria for setting the permittivity parameter c is to attenuate the wave without causing reflection. Theoretically, any value of c can be used without causing reflection. However in a numerical simulation, realization of a theory can only be approximate because of discretization using finite number of mesh points. Thus, to minimize reflection, we recommend using unity as the real value of c (i.e. the same real reflective index in the PML as in the isotropic medium).

When choosing the imaginary part of c, please note the two extremes. If $Im(c)$ is too large, the substantial difference between the index of the isotropic medium and the PML may cause significant reflection and you may observe standing wave ripples in the isotropic medium near the PML. On the other hand, if $Im(c)$ is too small, the attenuation of the wave within the PML is so weak that it requires a thick PML layer and many mesh points within the PML. If the thickness of the PML is not large enough, the truncation of the eigenmatrix within the PML will cause reflection within the PML which may propagate all the way back to the isotropic medium. Again, we may observe standing wave ripples in the isotropic medium. Therefore the PML should be adjusted in such a way that standing wave ripples do not appear

in the isotropic medium.

Based on some of our initial tests, a value of 0.02 for the imaginary part of c goes well with a PML layer thickness of 5 microns. 50 mesh points may be allocated for such a PML layer.

12.7 Optical Pumping and Incident Light

The program allows the use of an external light input to the device. It will be treated as a bias component in the same way as electrode bias. For APSYS, this is a crucial part of modeling photodetectors and solar cells while PICS3D uses the same idea to model SOAs and modulators.

When the device being modeled has an active region, there will be two different wavelengths involved in the simulation: a shorter wavelength controlling the optical generation (i.e. the optical pump) and a longer wavelength resulting from the radiative/stimulated emission. In LASTIP and PICS3D, this is used for optically pumped lasers; similarly, APSYS can model optically pumped LEDs.

Fig. 12.8 shows that for a typical active region under a specific carrier injection condition, it is possible for the material to exhibit interband absorption (or negative gain, thus e-h pair generation) at shorter wavelength while yielding optical gain at the peak gain wavelength. Our accurate optical gain spectrum model is evaluated at every mesh point using an active macro (not just the active region) and will thus respond to different wavelengths by providing either gain or absorption. The transmission of the light through a multi-layer device is modeled using a plane wave transfer matrix approach. The local absorption of the pump generates carriers which, in turn, may generate light at a different wavelength.

The optical pump may be defined as a single wavelength (i.e. pumping by an external laser source) or by using an input spectrum (i.e. a flash lamp or solar spectrum illumination). If a spectrum is provided, then the transfer matrix is calculated for each individual wavelength and the optical generation term is integrated over the whole spectrum.

As in the key emission wavelength, the incident wave can be also be treated as a guided wave and thus all previous discussion of lateral optical mode may be applied without modification.

For more information about how to define an incident light source, refer to the reference section and the **light_power** statement.

To control the input light in the simulation, the user should remember that the equilibrium calculations define an initial state without any external bias. As the input light is one kind of bias, it needs to be "turned on" with the **scan** statement. The scan variable "light" starts off with a value of zero and is used as a multiplier

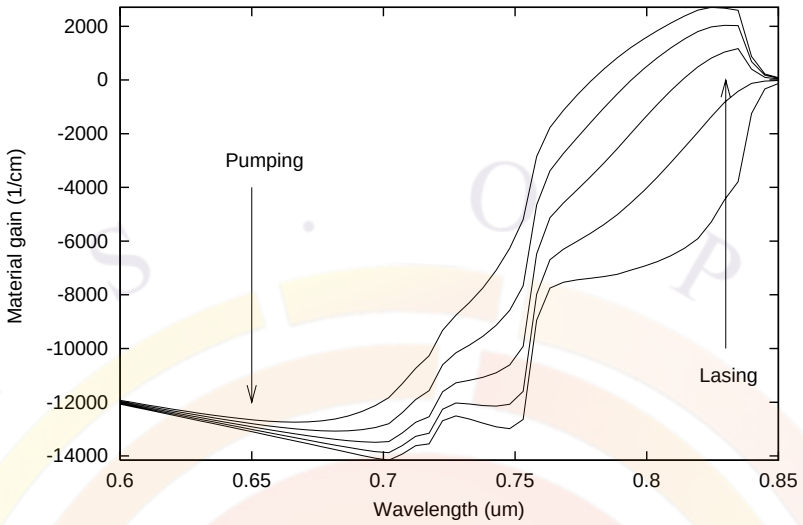


Figure 12.8: Optical gain spectrum and indication of pumping and lasing wavelengths.

to the input light defined in **light_power**. Setting the value to 1 in a scan fully turns on the input light but larger values can be used; for example, one may wish to multiply an input light spectrum for a multi-sun solar cell simulation.

Chapter 13

WURTZITE STRAINED MQW

13.1 Introduction

Recently, wurtzite strained quantum wells have been studied intensively because of applications in GaN blue green lasers and LEDs. However, the previous band structure models are only applicable for zincblende compounds. In this section, we will describe the MQW band structure and optical transition models for wurtzite crystals. We follow the work of S.L. Chuang [62] closely here.

We must point out a couple of major differences between zincblende and wurtzite crystals. The first issue is the base lattice. For zincblende material system, the basic lattice constant is usually well-known (e.g. GaAs or InP) and the strain is well defined in each layer once we know the substrate lattice. In the wurtzite system, a bulk GaN-based material layer is sometimes grown on a thick buffer layer. Through dislocations and defects, the buffer layer may relax and it is not clear what the base lattice constant of the MQW should be.

Although the default lattice base constant in our software is that of GaN, it may not be true for many systems. For example, if a thick bulk layer of AlGaN is grown on a sapphire substrate before the MQW is placed on top, the base lattice constant should be that of bulk AlGaN instead of GaN. For this reason, our active layer macros for wurtzite MQW provide a statement to describe the base lattice so that the users can adjust it according to individual material systems.

Another important issue is material parameters. As we will see below, there are many times more band structure and strain parameters for wurtzite than for the zincblende material system. Since wurtzite semiconductor band structures and optical properties are less understood and many bandŋstructure parameters are not available, our expectation of the accuracy of the simulation should be lower than that for zincblende material system.

On a final note, other wurtzite compounds such as ZnO have become a popular field

of study in recent years. Most of the discussions from this chapter will apply but care must be taken to use the correct material parameters and base lattices since strain terms will differ.

13.2 Bulk Band Structure

Consider a strained wurtzite crystal pseudomorphically grown along the c-axis (z axis) on another thick wurtzite layer. The base lattice constant is a_0 and the original lattice constant of the layer under consideration is a. The strain tensor in the well region has the following elements:

$$\begin{aligned} \varepsilon_{xx} &= \varepsilon_{yy} = \frac{a_0 - a}{a} \\ \varepsilon_{zz} &= -2\frac{C_{13}}{C_{33}}\varepsilon_{xx} \\ \varepsilon_{xy} &= \varepsilon_{yz} = \varepsilon_{zx} \end{aligned} \tag{13.1}$$

The conduction bands can be characterized by a parabolic band model with different electron effective masses m_e^t and m_e^z perpendicular and parallel to the c-growth direction, respectively. The hydrostatic energy shift in the conduction band can be written as:

$$P_{ce} = a_{cz}\varepsilon_{zz} + a_{ct}\varepsilon_{xx} + \varepsilon_{yy} \tag{13.2}$$

The band structure for the valence band is more complicated. Using the k.p method Chuang and Chang [63] have derived a 6 by 6 Hamiltonian which has been block diagonalized into the following upper and lower Hamiltonians:

$$H^v_{6\times 6}(k) = \begin{bmatrix} H^U_{3\times 3}(k) & 0 \\ 0 & H^L_{3\times 3}(k) \end{bmatrix} \tag{13.3}$$

where:

$$H^U = \begin{bmatrix} F & K_t & -iH_t \\ K_t & G & \Delta - iH_t \\ iH_t & \Delta + iH_t & \lambda \end{bmatrix} \tag{13.4}$$

$$H^L = \begin{bmatrix} F & K_t & iH_t \\ K_t & G & \Delta + iH_t \\ -iH_t & \Delta - iH_t & \lambda \end{bmatrix}$$

When there is strain, the energy bands of shift in a complicated manner. We need to use a common reference energy level when writing down the Hamiltonian. However, we wish to stress that this reference is *only* for the purposes of solving the Hamiltonian and obtain the bandgap, energy dispersion curves, etc...: in a full device simulation, band alignment follows the rules set out in Sec. 10.1.

Following the above convention for the wurtzite structure, if E_c^0 is the unstrained conduction band edge, the reference valence band level is as follows:

$$E_v^0 = E_c^0 - (E_g + \Delta_1 + \Delta_2) \tag{13.5}$$

When there is strain, the conduction band is shifted by an amount P_{ce} and the new reference level is:

$$E_v^0 = E_c - (E_g + \Delta_1 + \Delta_2 + P_{ce}) \tag{13.6}$$

Using the above reference energy, the matrix elements are defined as follows:

$$\begin{aligned}
F &= \Delta_1 + \Delta_2 + \lambda + \theta \\
G &= \Delta_1 - \Delta_2 + \lambda + \theta \\
\lambda &= \lambda_k + \lambda_\varepsilon \\
\lambda_k &= \frac{\hbar^2}{2m_0}(A_1 k_z^2 + A_2 k_t^2) \\
\lambda_\varepsilon &= D_1 \varepsilon_{zz} + D_2(\varepsilon_{xx} + \varepsilon_{yy}) \\
\theta &= \theta_k + \theta_\varepsilon \\
\theta_k &= \frac{\hbar^2}{2m_0}(A_3 k_z^2 + A_4 k_t^2) \\
\theta_\varepsilon &= D_3 \varepsilon_{zz} + D_4(\varepsilon_{xx} + \varepsilon_{yy}) \\
K_t &= \frac{\hbar^2}{2m_0} A_5 k_t^2 \\
H_t &= \frac{\hbar^2}{2m_0} A_6 k_t k_z \\
\Delta &= \sqrt{2}\Delta_3
\end{aligned} \tag{13.7}$$

Please note that the base vectors $|1>$ to $|6>$ of the above can be expressed by spherical harmonics $Y_{lm}(l = 1)$ [62]. For the upper Hamiltonian, $|1>$, $|2>$, $|3>$ corresponds to heavy hole (HH), light hole (LH) and crystal field split hole (CH), respectively. At the zone center (k=0), the HH subbands are decoupled from the LH and CH subbands while there is always a coupling between the LH and CH bands.

In our simulation software, the above Hamiltonian is solved to generate the bulk band structure from which the density of states are evaluated for bulk macros. To achieve maximum efficiency, the bulk valence band structure is fitted to a parabolic band for each of HH, LH and CH bands under any strain condition. Such fitted parabolic bands are used for modeling the non-active bulk regions. For MQW active region, a more rigorous approach is used which we shall describe below.

13.2.1 MQW Model - Effective Mass Approximation

In general, the dispersion of bulk valence bands is non-parabolic and anisotropic with strong mixing or anti-crossing behavior in the direction perpendicular (or transverse) to the c-axis. We need an effective mass model for the computation of density of states and for a simplified MQW model.

An efficient approximation is to take the effective masses fitted from the bulk band structure. This approach takes into account the anti-crossing behavior but the quality of the fit can be poor near some range for the transverse direction (see Fig. 13.1 and Fig. 13.2).

Based on Ref. [63], we have also implemented the following analytical effective mass model for the valence band:

- 1) Within a range of small k (a situation when the valence band is lightly populated by holes), the following effective masses hold:

$$\begin{aligned} m0/m^z_{hh} &= -(A_1 + A_3) \\ m0/m^z_{lh} &= -\left[A_1 + \left(\frac{E^0_2 - \lambda_e}{E^0_2 - E^0_3}\right) A_3\right] \\ m0/m^z_{ch} &= -\left[A_1 + \left(\frac{E^0_3 - \lambda_e}{E^0_3 - E^0_2}\right) A_3\right] \end{aligned} \tag{13.8}$$

$$\begin{aligned} m0/m^t_{hh} &= -(A_2 + A_4) \\ m0/m^t_{lh} &= -\left[A_2 + \left(\frac{E^0_2 - \lambda_e}{E^0_2 - E^0_3}\right) A_4\right] \\ m0/m^t_{ch} &= -\left[A_2 + \left(\frac{E^0_3 - \lambda_e}{E^0_3 - E^0_2}\right) A_4\right] \end{aligned} \tag{13.9}$$

  where $E^0_i (i = 1, 2, 3)$ are the valence band edges at k=0. The parabolic bands of this model are shown in Figs. 13.3 and 13.4.

- 2) For a large range of k (a situation when the valence band is heavily populated by holes), the following effective masses formulas are valid:

$$\begin{aligned} m0/m^z_{hh} &= -(A_1 + A_3) \\ m0/m^z_{lh} &= -(A_1 + A_3) \\ m0/m^z_{ch} &= -A_1 \end{aligned} \tag{13.10}$$

$$\begin{aligned} m0/m^t_{hh} &= -(A_2 + A_4 - A_5) \\ m0/m^t_{lh} &= -(A_2 + A_4 + A_5) \\ m0/m^t_{ch} &= -A_2 \end{aligned} \tag{13.11}$$

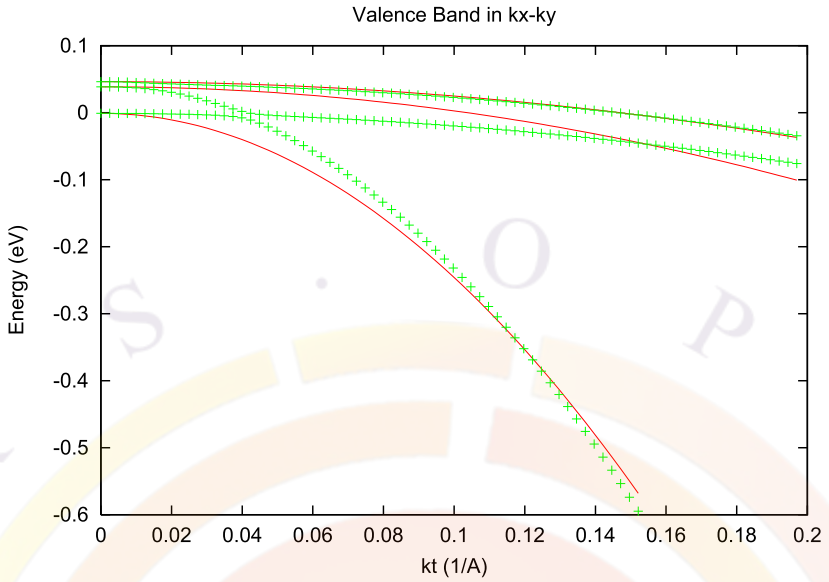


Figure 13.1: Transverse valence bands (points) fitted to effective mass model.

The parabolic bands of this model are shown in Figs. 13.5 and 13.6.

- 3) A compromise of the above two models is to average them. The parabolic bands with average masses are shown in Figs. 13.7 and 13.8.

The default setting in Crosslight software is 3) above. The user may adjust the setting of the effective mass model through the statement **modify_wurtzite**. The choice of model is based on how heavily the valence bands are populated by holes (i.e. hole carrier density).

13.3 Wurtzite dipole moments

Since the crystal symmetry of wurtzite is different from that of zincblende, the dipole moment in effective mass approximation for zincblende can not be used here and we need to use more accurate models (e.g. k.p theory) to compute the dipole moment for optical transitions. However, study of results in k-dependent dipole moment from k.p theory for a large number of structures leads us to propose the following simplified dipole moment enhancement factors:

$$\begin{aligned} A_{hh} &= 3/2 \\ A_{lh} &= \frac{3cos^2(\theta_e)}{2} \\ A_{ch} &= 0 \end{aligned} \tag{13.12}$$

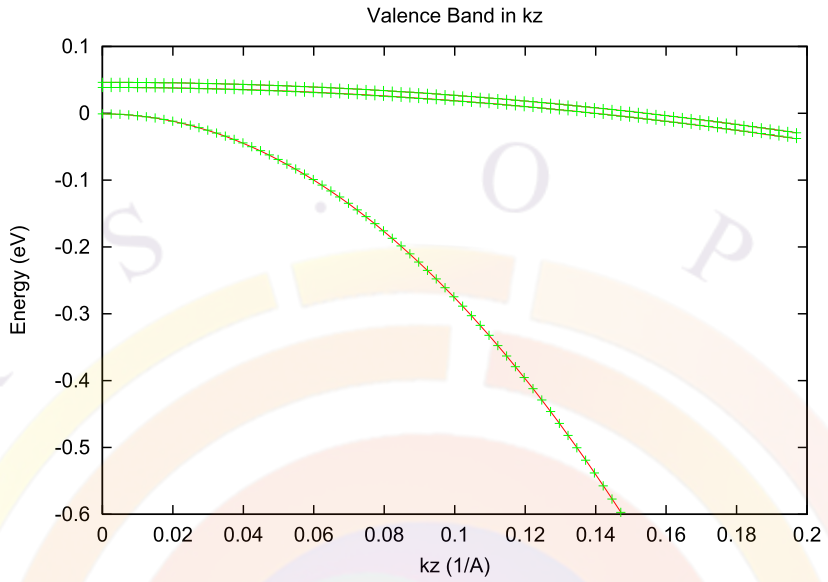


Figure 13.2: Valence bands along c-axis (points) fitted to effective mass model.

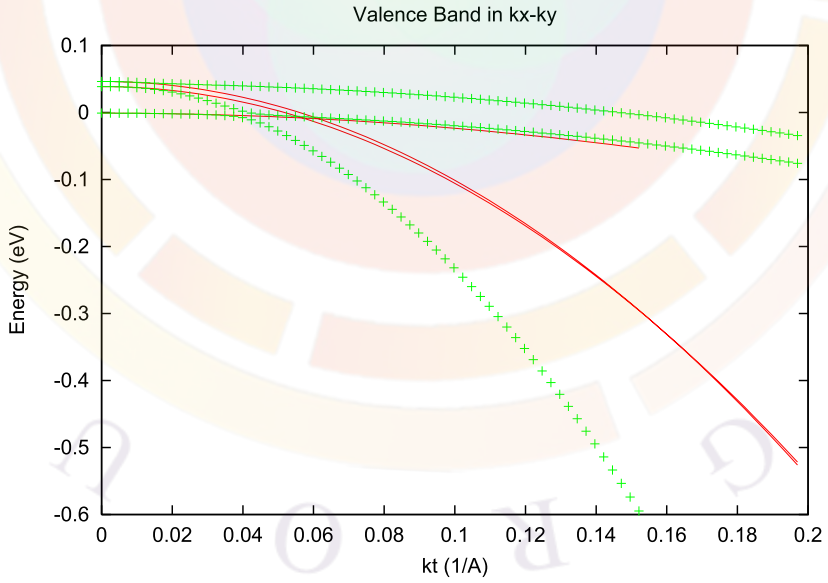


Figure 13.3: Transverse valence bands (points) as compared with analytical effective mass model of small k-range.

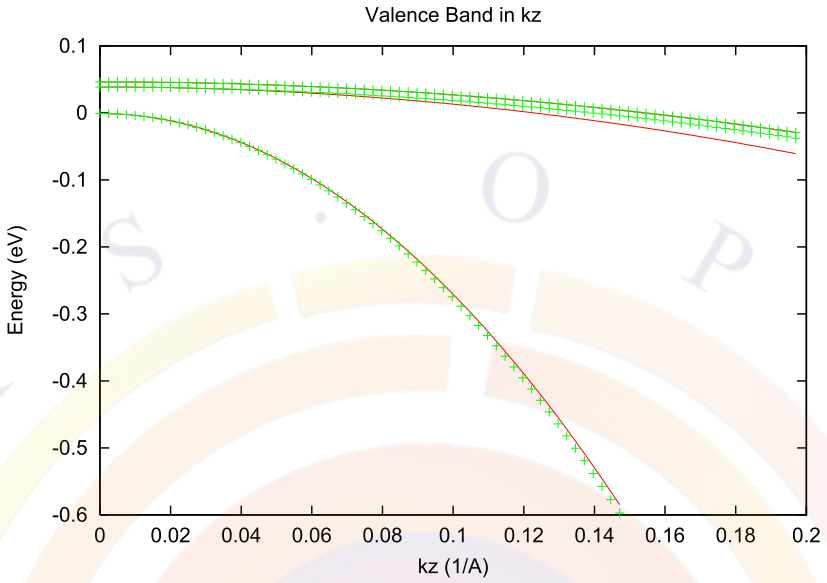


Figure 13.4: c-axis valence bands (points) as compared with analytical effective mass model of small k-range.

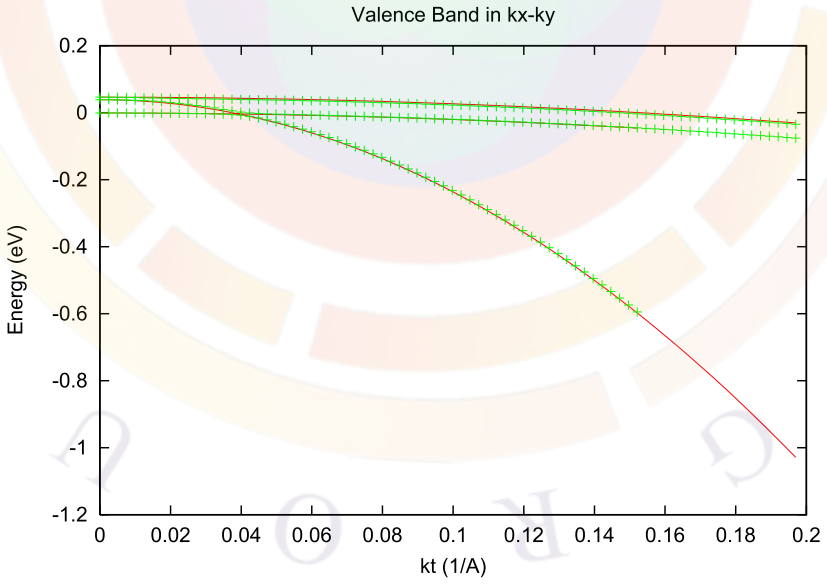


Figure 13.5: Transverse valence bands (points) as compared with analytical effective mass model of large k-range.

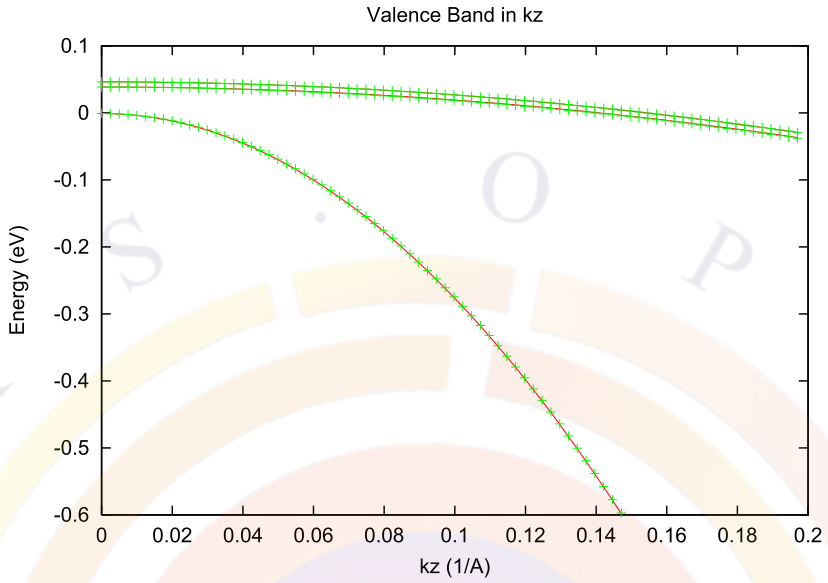


Figure 13.6: c-axis valence bands (points) as compared with analytical effective mass model of large k-range.

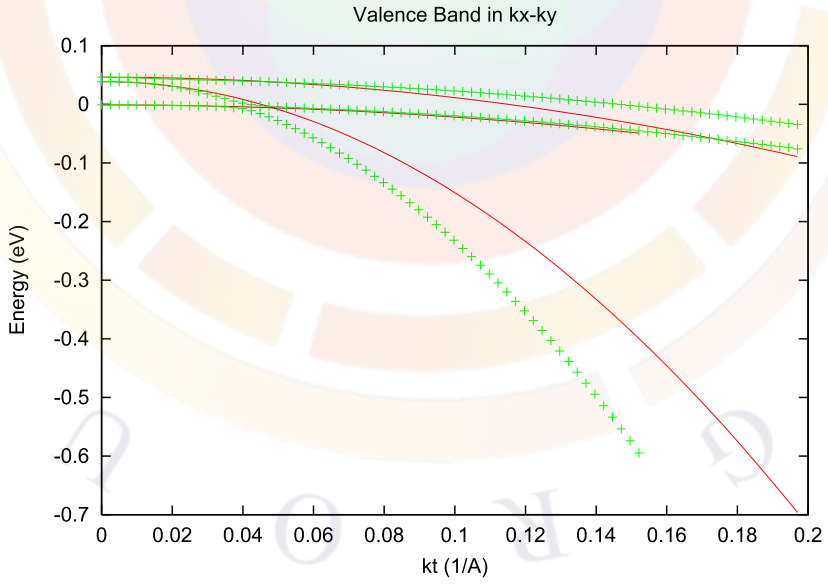


Figure 13.7: Transverse valence bands (points) as compared with analytical effective mass model of averaged masses.

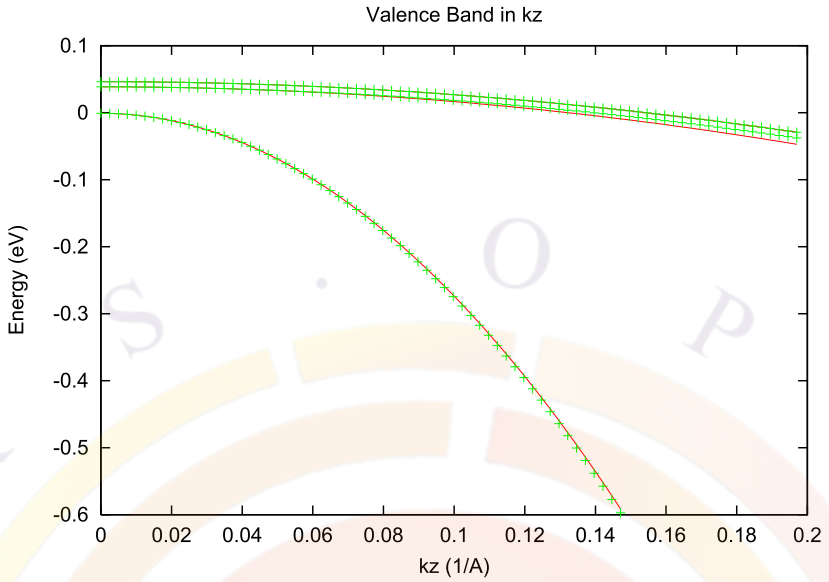


Figure 13.8: c-axis valence bands (points) as compared with analytical effective mass model of averaged masses.

for TE and

$$\begin{aligned} A_{hh} &= 0 \\ A_{lh} &= 3 - 3cos^2(\theta_e) \\ A_{ch} &= 3 \end{aligned} \tag{13.13}$$

for TM.

The $cos^2(\theta_e)$ term above is defined in the same way as in zincblende. Please note that we formulate the above factors in such a way that the following conservation rule is obeyed:

$$2 \times A_h^{TE} + A_h^{TM} = 3, (h = hh, lh, \text{ or } ch) \tag{13.14}$$

The dipole moment is expressed as:

$$\begin{aligned} M_{hh} &= A_{hh} O_{ij} M_b \\ M_{lh} &= A_{lh} O_{ij} M_b \\ M_{ch} &= A_{ch} O_{ij} M_b \end{aligned} \tag{13.15}$$

where O_{ij} is wave function overlap integral and M_b is bulk dipole moment given by [62] as follows:

$$\begin{aligned} M_b^{TM} &= \frac{m_0}{6} E_{pz} \\ E_{pz} &= \left(\frac{1}{m_e^z} - 1 \right) \frac{(E_g + \Delta_1 + \Delta_2)(E_g + 2\Delta_2) - 2\Delta_3^2}{E_g + 2\Delta_2} \end{aligned} \tag{13.16}$$

$$\begin{aligned} M_b^{TE} &= \frac{m_0}{6} E_{px} \\ E_{px} &= \left(\frac{1}{m_e^t} - 1\right) \frac{E_g[(E_g + \Delta_1 + \Delta_2)(E_g + 2\Delta_2) - 2\Delta_3^2]}{(E_g + \Delta_1 + \Delta_2)(E_g + \Delta_2) - \Delta_3^2} \end{aligned} \tag{13.17}$$

13.4 MQW Model – Valence Mixing

As in zincblende, off-diagonal elements in the Hamiltonian imply mixing of the valence band states. Within the envelope function approximation, we write the wave function as follows:

$$\Psi_m^U(z;k_t) = \frac{e^{i\mathbf{k_t}\cdot\mathbf{r_t}}}{\sqrt{A}} \left(g_m^{(1)}(z;k_t)|1> + g_m^{(2)}(z;k_t)|2> + g_m^{(3)}(z;k_t)|3>\right) \tag{13.18}$$

$$\Psi_m^L(z;k_t) = \frac{e^{i\mathbf{k_t}\cdot\mathbf{r_t}}}{\sqrt{A}} \left(g_m^(4)(z;k_t)|4> + g_m^{(5)}(z;k_t)|5> + g_m^{(6)}(z;k_t)|6>\right) \tag{13.19}$$

The valence subbands are determined by the following coupled differential equations:

$$\sum_{j=1}^{3} \left(H_{ij}^U \left(k_z = -i\frac{\partial}{\partial z}\right) + \delta_{ij} E_v^0(z)\right) g_m^{(j)}(z;k_t) = E_m^U(k_t) g_m^{(i)}(z;k_t) \tag{13.20}$$

The valence band discontinuity is represented by discontinuity in $E_v^0(z)$ (the reference energy). Similar equations can be written down for the lower Hamiltonian. It can be shown that MQW with reflection symmetry $E_v^0(z) = E_v^0(-z)$, the upper and lower Hamiltonians have the same band structures. In the software, we use finite difference method to solve the coupled differential equations. This gives us the valence mixing subbands.

The dipole moments, taking into account the upper and lower Hamiltonian degeneracy, are given as follows:

$$M_{nm}^{TE} = M_b \frac{3}{4} \left[< \phi_n | g_m^{(1)} >^2 + < \phi_n | g_m^{(2)} >^2\right] \tag{13.21}$$

$$M_{nm}^{TM} = M_b \frac{3}{2} < \phi_n | g_m^{(3)} >^2 \tag{13.22}$$

13.5 Polarization and interface charges

One of the most troublesome properties of the wurtzite material system is that its most common compounds have spontaneous and strain-induced polarization terms.

This manifests itself as a fixed interface charges at heterojunction interfaces. In quantum wells, these charges create a local field which separates carriers (i.e. Quantum-Confined Stark Effect). This affects the wave functions and the resulting gain calculations.

To model this effect properly, the user must use the **self_consistent** statement: this will force the solver to iterate between the Poisson/drift-diffusion equations and the Schrōdinger solver so a self-consistent solution can be found. Without this, the Schrōdinger solver will assume a flat band profile.

Since the local field can be different from well to well in a MQW region, it is recommended that the calculations for each well be done independently. To do this, a different material number must be assigned to each well: the **independent_mqw** statement in the layer file can automate this. If the same material number is used for multiple wells, the Schrōdinger and gain calculations will only be done once.

Interface charges can be defined manually with the **interface** statement. For the InGaN and AlGaN material system, this process has been automated in the layer file with the **set_polarization** statement.

13.6 Non-Polar and Semi-Polar materials

The polarization effects discussed in the previous section occur along the most natural growth orientation of wurtzite crystals: c-plane. However, it is possible to grow the material in other orientations where these effects can be reduced or even eliminated [65].

In this case, a full 6x6 k.p method must be used to find the correct masses for this orientation. This is done with the **modify_wurtzite** and **modify_qw** statements.

The gain and spontaneous emission integrals must also be done differently in this situation and an explicit averaging of the results over the direction of the in-plane wave vector must be used. This can be time consuming so it is recommended to run the gain calculations separately and use tabulated values in the main device simulation.

Crystal plane orientations are given using the Bravais-Miller indices $(hkil)$ where $i = -h - k$: the redundant index is useful in highlighting equivalent symmetric directions. The plane normal vector is specified using the rotation $(x \rightarrow x')$ angles ϕ and θ. To take into account the anisotropic refractive index in mode calculations, the direction of TE mode (x'') or the waveguide propagation direction (y'') must also be specified with the in-plane angle ψ. These angles are illustrated in Fig. 13.9.

Some common plane orientations are shown in Table 13.1. Note that because of rotational symmetry, multiple ϕ values can be used for the same plane orientation. However, the in-plane waveguide direction may need to be adjusted.

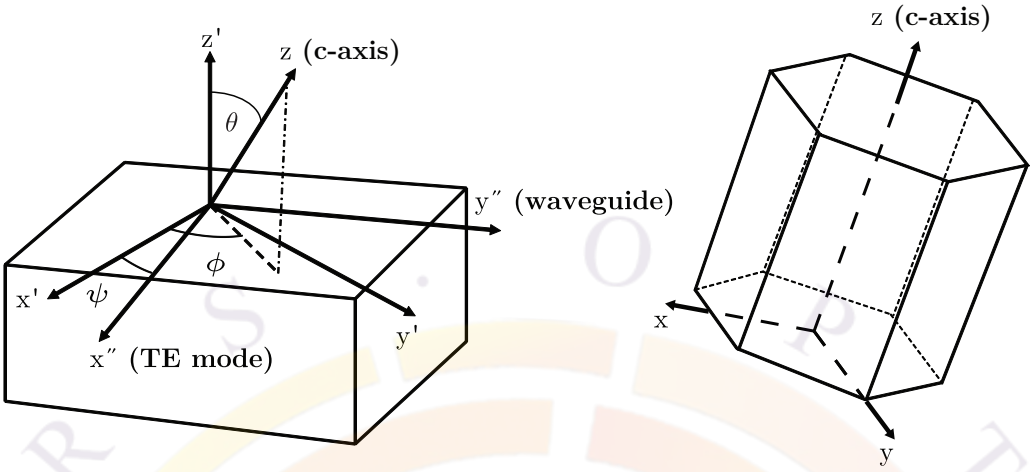


Figure 13.9: Rotation angles for non-polar and semi-polar wurtzite plane orientations.

Orientation	**Bravais-Miller Indices**	**Classification**	ϕ	θ
c-plane	(0001)	polar	0	0
m-plane	$\left(1\bar{1}00\right)$	non-polar	0	90
a-plane	$\left(11\bar{2}0\right)$	non-polar	30	90
r-plane	$\left(1\bar{1}02\right)$	semi-polar	0	$atan\left(\frac{c}{2a}\right)$
?	$\left(10\bar{1}1\right)$	semi-polar	0	$atan\left(\frac{c}{a}\right)$
?	$\left(11\bar{2}2\right)$	semi-polar	30	$atan\left(\frac{c}{2a}\right)$

Table 13.1: Common plane orientations in wurtzite crystals .

13.7 Nomenclature

We list some critical symbol definitions here because these are specific to wurtzite band structure.

a, c	Lattice constants of Hexagonal structure.
E_g	Bandgap.
E_g	Bandgap.
$\Delta_1, \Delta_2, \Delta_3$	Energy parameters.
Δ_{so}	Spin-orbit coupling energy.
m_e^z, m_e^t	Conduction band effective masses along c-axis (z) and transverse (x-y) direction, respective.
$A_i, (i = 1, ..., 6)$	Valence band effective mass parameters.
a_h, a_c, a_v	Hydrostatic deformation potentials of total, conduction, and valence parts, respectively.
$D_i, (i = 1, ..., 4)$	Valence band shear deformation potentials.
C_{13}, C_{33}	Elastic stiffness constants.

PARS OPTICS GROUP

Part III

PRODUCT SPECIFIC MODELS

PARS OPTICS GROUP

Chapter 14

APSYS SPECIFIC MODELS

14.1 Time-Dependent Traveling Wave Equations

As of the 2012 version of APSYS, this model is considered obsolete: please use PICS3D to model optical amplifiers (SOA), modulators (EAM) and superluminescent diodes (SLED). The new model is superior in a number of ways, including a full 3D model that couples longitudinal (wave propagation) and lateral effects in the same sparse solver.

14.2 Theories of Light Emitting Diodes

14.2.1 Introduction

A light emitting diode (LED) is different from a laser diode (LD) and deserves special treatment for the following reasons:

- 1) LED operates well below lasing threshold while theories of lasers almost always assumes lasing condition
- 2) Absence of simulated recombination in LED
- 3) Continuous emission spectrum must be considered for LED while longitudinal modes of laser only requires a limited number of lasing wavelengths
- 4) The randomness of spontaneous recombination in LED in contrast to the coherence of laser modes which affects the near and far-field models.

In this section, we attempt to establish the basic theories for LED emission in one direction: perpendicular to the semiconductor surface). This allows us to use a

simple 1D model to formulate our theory; once the basic formulas are established, we will extend it to emission in other directions.

Many of the theories here were initially formulated by Henry for laser diodes and semiconductor optical amplifiers. We find that the same theories are also applicable for LEDs. Interested readers are encouraged to consult Ref. [80].

The model described here is implemented in the **led_control** statement. A simplified version of the model is available through the **led_simple** statement: this is useful when used in conjunction with the ray tracing model of Section 14.5.

14.2.2 Fabry-Perot Cavity and LED

A LED may be regarded as a special case of a Fabry-Perot laser. Without loss of generality, we consider the following device configuration: the z-direction has a waveguiding structure which supports confined lateral/traverse modes. We later relax this condition to include the case with no optical confinement. Please note that in LED, we must treat as many modes as possible since they all contribute to the emission power because of the non-coherent nature of spontaneous emission. In the case of a LD, only a limited number of highly coherent and highly powered modes are considered. For simplicity, we assume throughout this section that the LED does not have any optical coatings so that light emits directly from the semiconductor to the air.

The theoretical bases regarding the wave function, the Wronskian and the "Diffusion" coefficient and its relation to semiconductor spontaneous recombination are covered in Appendix D. We shall only use the results here.

We will consider a waveguide with well confined lateral/transverse modes (**xy-mode** model). Later, we will relax this constrain and consider a case only confined transverse modes in y direction exist (we call it **y-mode** model). Finally, we consider a case where there are no confined modes (we call it **uniform model**).

For a Fabry-Perot cavity from 0 to L, the two basic wave functions Z_1 and Z_2 are as follows:

$$Z_1(z) = r_1 exp(jkz) + exp(-jkz) \tag{14.1}$$

$$Z_2(z) = exp(jkz) + r_2 exp(2jkL - jkz)] \tag{14.2}$$

We construct the following Wronskian:

$$\begin{aligned} W &= Z_1(z)Z_2'(z) - Z_2(z)Z_1'(z) \\ &= [r_1 exp(jkz) + exp(-jkz)] \\ &\quad \times [jkexp(jkz) - jkr_2 exp(2jkL - jkz)] \\ &\quad - [exp(jkz) + r_2 exp(2jkL - jkz)] \\ &\quad \times [jkr_1 exp(jkz) - jkexp(-jkz)] \\ &= -r_1 exp(jkz) jkr_2 exp(2jkL - jkz) \\ &\quad + exp(-jkz)]jkr_1 exp(jkz) \\ &\quad + r_1 exp(jkz) jkexp(-jkz) \\ &\quad - r_2 exp(2jkL - jkz) jkr_1 exp(jkz) \\ &= 2jk[1 - r_1 r_2 exp(2jkL)] \end{aligned} \tag{14.3}$$

Consider the solution of the field in terms of the Green's function:

$$g(z, z_s)W_n = Z_1(z)Z_2(z_s)\theta(z_s - z) + Z_2(z)Z_1(z_s)\theta(z - z_s) \tag{14.4}$$

$$\begin{aligned} E(z)W_n &= \int dz_s f(z_s)[Z_1(z)Z_2(z_s)\theta(z_s - z) + Z_2(z)Z_1(z_s)\theta(z - z_s)] \\ &= Z_1(z)\int_z^L Z_2(z_s)dz_s f(z_s) + Z_2(z)\int_0^z Z_1(z_s)dz_s f(z_s) \end{aligned} \tag{14.5}$$

where $f(z_s)$ is the noise term due to spontaneous emission the details of which can be found in Appendix D.

Consider the squared ensemble average

$$\begin{aligned} < |E(z)W_n|^2 > &= < [Z_1(z)\int_z^L Z_2(z_s)dz_s f(z_s) \\ &+ Z_2(z)\int_0^z Z_1(z_s)dz_s f(z_s)] \\ &\quad [Z_1^*(z)\int_z^L Z_2^*(z_t)dz_t f^*(z_t) \\ &+ Z_2^*(z)\int_0^z Z_1^*(z_t)dz_t f^*(z_t)] > \end{aligned} \tag{14.6}$$

We note that the cross terms of the above four terms are zero since noise sources are uncorrelated:

$$\begin{aligned} < |E(z)W_n|^2 > &= |Z_1(z)|^2 \int_z^L |Z_2(z_s)|^2 dz_s f_0(z_s) \\ &+ |Z_2(z)|^2 \int_0^z |Z_1(z_s)|^2 dz_s f_0(z_s) \\ &= |Z_1(z)|^2 \int_z^L |Z_2(z_s)|^2 dz_s 2D_{FF}(z_s) \\ &+ |Z_2(z)|^2 \int_0^z |Z_1(z_s)|^2 dz_s 2D_{FF}(z_s) \end{aligned} \tag{14.7}$$

As usual, we ignore the interference terms when we consider energy flow so that

$$|Z_1(z)|^2 \approx r_1^2 exp(-2k''z) + exp(2k''z) \tag{14.8}$$

$$|Z_2(z)|^2 \approx exp(-2k''z) + r_2^2 exp(-2k''(2L-z)] \tag{14.9}$$

We note that $|W_n|^2$ varies between $4k^2(1-r_1r_2)^2$ and $4k^2(1+r_1r_2)^2$ rapidly. To a good approximation, we take $|W_n|^2 = 4k^2$ to simplify matters.

The linear photon density (in units of 1/m) is given by

$$S(z)\hbar\omega\Delta\omega = \sum_n 2\varepsilon_0 n n_g < |E(z)|^2 > \tag{14.10}$$

where the summation n is over different lateral modes.

The power emission spectrum may be calculated from left and right facets as follows:

$$P_L(\omega) Delta\omega = \frac{1-r_1^2}{1+r_1^2} \sum_n v_g 2\varepsilon_0 n n_g < |E(0)|^2 > \tag{14.11}$$

where v_g and n_g are the group velocity and group index, respectively.

At the left facet:

$$< |E(0)|^2 >= |Z_1(0)|^2/|W_n|^2 \int_0^L |Z_2(z_s)|^2 dz_s 2D_{FF}(z_s) \tag{14.12}$$

$$P_L(\omega) = \frac{1-r_1^2}{1+r_1^2} v_g 2\varepsilon_0 n n_g \frac{(1+r_1^2)}{4k^2} \int_0^L |Z_2(z_s)|^2 dz_s 2D_{FF}(z_s) \tag{14.13}$$

As we derived in Appendix D that the diffusion coefficient

$$2D_{FF} = \frac{\pi\hbar}{\varepsilon_0 v_g n^2} \hbar\omega < n|r_{sp}(\omega)|n > . \tag{14.14}$$

where r_{sp} is the spontaneous emission rate at a photon frequency. $|n>$ is used to denote the wave function integral for the nth lateral mode.

$$P_L(\omega) = \sum_n \frac{1-r_1^2}{1+r_1^2} v_g 2\varepsilon_0 n n_g \frac{(1+r_1^2)}{4k^2} \frac{\pi\hbar}{\varepsilon_0 v_g n^2} \int_0^L |Z_2(z_s)|^2 dz_s \hbar\omega < n|r_{sp}(\omega)|n > . \tag{14.15}$$

Our key result for spontaneous emission power for an ω interval is as follows:

$$P_L(\omega)\Delta\omega = \sum_n (1-r_1^2) \frac{2n_g\pi}{n} \frac{1}{4k^2} \int_0^L |Z_2(z_s)|^2 dz_s (\hbar\omega) < n|r_{sp}(E)|n > \Delta E. \tag{14.16}$$

The above formula is applicable for a waveguide with a limited number of lateral / transverse modes (the xy-mode model). It can be adapted for different circumstances as follows.

We consider a device cross section with active region large enough to accommodate many different lateral modes in both x and y direction (the plane wave model or uniform model). Let us now count the number of allowable lateral modes at frequency ω.

If we assume a large rectangular device active cross section:

$$(\omega/c)^2 = k_z^2 + k_x^2 + k_y^2 \tag{14.17}$$

At a fixed emission frequency, if we consider all emissions in $\pm z$ direction, the allowable lateral states are confined within a circle of

$$k_x^2 + k_y^2 < (\omega n/c)^2 \tag{14.18}$$

Thus,

$$\sum_n = A\pi k^2/(4\pi^2) = Ak^2/(4\pi) \tag{14.19}$$

where A is the active region cross section. We assume all the modes overlaps with a uniform active region perfectly (uniform model) so that $< n|n >= 1$.

So far, all of our models based on the Green's function are one-dimensional. Thus the wave number k should be understood as k_z if the actual propagation direction is not in z-direction. We make a further approximation to replace k_z by a certain average $< k_z >$ and remove the summation altogether:

$$P_L(\omega)\Delta\omega = (1 - r_1^2)\frac{n_g}{2n}\frac{k^2}{4 < k_z >^2} A \int_0^L |Z_2(z_s)|^2 dz_s(\hbar\omega) r_{sp}(E)\Delta E. \tag{14.20}$$

The internal emission power from spontaneous emission is

$$P_{spon}(\omega)\Delta\omega = A\int_0^L dz_s(\hbar\omega r_{sp}(E))\Delta E. \tag{14.21}$$

The ratio of $\zeta_{ext} = P_L/P_{spon}$ gives us the LED external efficiency at a frequency in the left direction for such a highly multimode device.

If we assume that r_{sp} is reasonably uniform in z-direction, the external efficiency in one direction can be written as:

$$\zeta_{ext} = (1 - r_1^2)\frac{n_g}{2n}\frac{k^2}{4 < k_z >^2}\int_0^L |Z_2(z_s)|^2 dz_s/L \tag{14.22}$$

We choose $k^2/ < k_z >^2 = 4/3$ so that in the limit of zero facet reflectivity and transparent material, the external efficiency for one facet is 1/6, as it should be if the device is in a cubic shape.

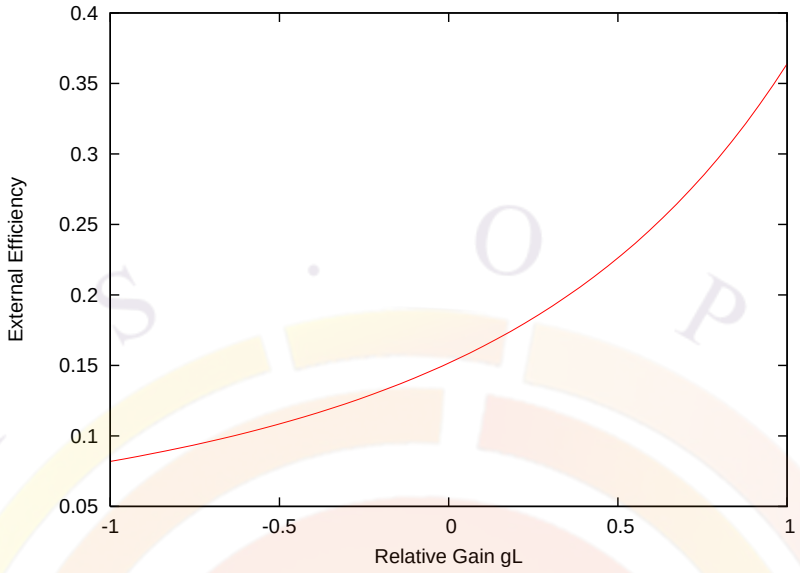


Figure 14.1: External efficiency versus relative gain (gL) of an LED with large active region cross section supporting many lateral modes.

Thus:

$$P_L(\omega)\Delta\omega = (1-r_1^2)\frac{n_g}{6n}A\int_0^L |Z_2(z_s)|^2 dz_s(\hbar\omega)r_{sp}(E)\Delta E. \tag{14.23}$$

and the external efficiency for a case of uniform spontaneous emission medium is:

$$\zeta_{ext} = (1-r_1^2)\frac{n_g}{6n}\int_0^L |Z_2(z_s)|^2 dz_s/L \tag{14.24}$$

The integral is evaluated as follows:

$$\begin{aligned}\int_0^L |Z_2(z)|^2 dz/L &= \int_0^L dz/L[exp(-2k''z) + r_2^2 exp(-2k''(2L-z)] \\ &= \frac{1}{2k''L}[1-exp(-2k''L)] \\ &- \frac{1}{2k''L}r_2^2 exp(-4k''L)[1-exp(2k''L)]\end{aligned} \tag{14.25}$$

To gain some insight into how the external efficiency of a LED depends on the gain/loss of the material, we plot the single facet external efficiency of an LED with large active region cross section supporting many lateral modes (uniform model). Parameters are chosen as follows: $r1^2 = r_2^2 = 0.3$, $n = n_g = 3.4$. The result is shown in Fig. 14.1. As expected, the efficiency goes up as gain is increased. At transparency, the external efficiency amounts to 1/6 for a single facet which accounts for 100 percent of light emission in one direction.

Similar expression can be written for traveling waves in other directions. For example:

$$P_R(\omega)\Delta\omega = \sum_n (1-r_1^2)/(1+r_2^2)\frac{2n_g\pi}{n}\frac{1}{4k^2}Z_2(L)\int_0^L |Z_1(z_s)|^2 dz_s(\hbar\omega) < n|r_{sp}(E)|n > \Delta E. \tag{14.26}$$

The integral is evaluated as follows:

$$\begin{aligned}\int_0^L |Z_1(z)|^2 dz/L &= \int_0^L dz/L[r_1^2 exp(-2k''z) + exp(2k''z)] \\ &= \frac{1}{2k''L}r_1^2[1-exp(-2k''L)] \\ &- \frac{1}{2k''L}[1-exp(2k''L)]\end{aligned} \tag{14.27}$$

$$P_R(\omega)\Delta\omega = \sum_n (1-r_2^2)\frac{2n_g\pi}{n}\frac{1}{4k^2}exp(-2k''L)\int_0^L |Z_1(z_s)|^2 dz_s(\hbar\omega) < n|r_{sp}(E)|n > \Delta E. \tag{14.28}$$

$$\begin{aligned}exp(-2k''L)\int_0^L |Z_1(z)|^2 dz/L &= \frac{1}{2k''L}r_1^2 exp(-2k''L)[1-exp(-2k''L)] \\ &+ \frac{1}{2k''L}[1-exp(-2k''L)] \\ &= \frac{1}{2k''L}[1-exp(-2k''L)] \\ &- \frac{1}{2k''L}r_1^2 exp(-4k''L)[1-exp(2k''L)]\end{aligned} \tag{14.29}$$

which is completely consistent with our result for the left facet. Thus we will only consider emission in one direction without loss of generality.

14.2.3 Internal reflection and LED emission power

The result in Eq. 14.23 is obtained under rather ideal condition and assumes that all emission in $\pm z$ can escape from the LED. The result is useful for qualitative analysis regarding the dependence of gain/loss. However, the efficiency from this result is overestimated because in reality, only a small fraction of the light in $\pm z$ can escape from the LED simply because of the internal total reflection angle is much smaller than 90 degrees. For a more accurate theory, we will re-evaluate Eq. 14.16 taking into account the internal reflection angle.

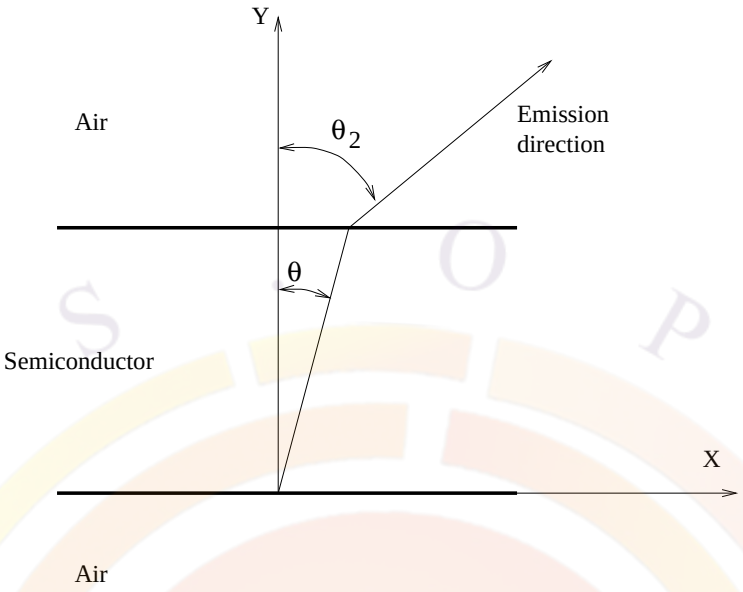


Figure 14.2: Schematics showing the angle of refraction for an LED.

If the effective refractive index of the material is n_{e1}, the maximum internal reflection angle θ_{max} (see also Fig. 14.2) is:

$$sin\theta_{max} = 1/n_{e1} \tag{14.30}$$

For the uniform model, we should only consider the wave vectors within this angle:

$$\sum_n = \int_0^{k/n_{e1}} A dk_r^2/(4\pi) \tag{14.31}$$

where

$$k_r^2 = k_x^2 + k_y^2 \tag{14.32}$$

Please note that the wave number in Eq. 14.16 is to be understood as k_z. We need to evaluate the numerical factor as follows:

$$\begin{aligned} f_{cone} &= \\ &= \frac{1}{8}\int_0^{k/n_{e1}} dk_r^2/(k^2 - k_z^2) \\ &= \frac{1}{8} ln\left(\frac{n_{e1}^2}{n_{e1}^2 - 1}\right) \end{aligned} \tag{14.33}$$

The power emission from facet 1 (left) is

$$P_L(\omega)\Delta\omega = (1 - r_1^2)\frac{An_g}{8n} ln\left(\frac{n_{e1}^2}{n_{e1}^2 - 1}\right)\int_0^L |Z_2(z_s)|^2 dz_s(\hbar\omega) r_{sp}(E)\Delta E. \tag{14.34}$$

Please note that the above emission power is much smaller than the ideal case in Eq. 14.23 because the new numerical factor is much smaller:

$$\frac{1}{8} ln \left(\frac{n_{e1}^2}{n_{e1}^2 - 1} \right) < 1/6 \tag{14.35}$$

It is clear that for an LED with simple configuration, the main power limitation is set by the internal reflection angle. Most of the emission power is lost due to internal facet reflection and absorption. Only a small fraction (around 10 percent) escapes through a small cone vertical to the LED surface.

The reduction factor in Eq. (14.33) has a problem of consistency with the factor of 1/6 because the former does not reduce to the latter if the index of refraction tends to unity. Therefore, we should use the same approximation of $< k_z^2 > /k^2 = 3/4$ and obtain:

$$\begin{aligned} f_{cone} &= \\ &= \frac{1}{8 < k_z^2 >} \int_0^{k/n_{e1}} dk_r^2 \\ &= \frac{1}{6 n_{e1}^2} \end{aligned} \tag{14.36}$$

The above formula of cone reduction factor is preferred because it reduces to the ideal case of no internal reflection if the index tends to unity. For most applications in LED emission to the top surface, the size of the active region facing the top is large enough to use the uniform model and thus Eq. 14.36 is our model of choice. The simulation program has a switch to allow the user to include or exclude the effect of total internal reflection.

By default, the idea case of no internal reflection is used so that the power emission is calculated by setting index to unity in Eq. 14.36. This represents an overestimate of emission in one direction. On the other hand, use of total reflection to assume that all lights being totally reflected are lost is an under-estimate because modern LED's often have special geometric design to enable the lights being totally reflected to change directions and to escape the device.

In some cases, such as emitting from the sides of a thin device with single mode confining waveguide, it is useful to consider a case where only one direction (say y-direction) has confined modes (y-mode model) while the x-direction is unconfined with a large device dimension L_x. We can simplify mode summation in the x-direction as follows.

Recall our basic result:

$$P_L(\omega)\Delta\omega = \sum_{nx} \sum_{ny} (1 - r_1^2) \frac{2 n_g \pi}{n} \frac{1}{4k^2} \int_0^L |Z_2(z_s)|^2 dz_s (\hbar\omega) < n|r_{sp}(E)|n > \Delta E. \tag{14.37}$$

The number of available modes in the x-direction is simply:

$$\sum_{nx} = \int_0^{k/n_{e1}} L_x dk_r/(2\pi) \tag{14.38}$$

Thus, we need to evaluate the integral:

$$\begin{aligned} g_l = & \frac{1}{4}\int_0^{k/n_{e1}} dk_r/(k^2 - k_r^2) \\ = & \frac{1}{8} ln\left(\frac{n_{e1}+1}{n_{e1}-1}\right) \end{aligned} \tag{14.39}$$

We then obtain our result for y-mode model:

$$P_L(\omega)\Delta\omega = \sum_{ny}(1-r_1^2)\frac{n_g}{n}\frac{L_x g_l}{k}\int_0^L |Z_2(z_s)|^2 dz_s(\hbar\omega) < n|r_{sp}(E)|n > \Delta E. \tag{14.40}$$

14.2.4 Far field distribution

We use a rather simple model to evaluate the far field distribution. We consider three effects:

- 1.) The optical path is different (longer) when the emission angle is not vertical. The longer the optical path, the more power absorption occurs.
- 2.) The power reflection r_1^2 is a function of the emission angle.
- 3.) Light direction change due to refraction causes a solid angle within the semiconductor to be magnified upon refraction. More details are discussed as follows.

It is reasonable to assume that the spontaneous emission is random and has uniform intensity in all directions within the semiconductor. Consider emission within a solid angle $\Delta\theta\Delta\phi$. The θ angle is determined by the following equation (see also Fig. 14.2):

$$n_{e1} cos\theta\Delta\theta = cos\theta_2\Delta\theta_2 \tag{14.41}$$

Thus the solid angle is amplified and the light intensity is reduced by a factor

$$\left(\frac{cos\theta_2}{n_{e1}cos\theta}\right)\left(\frac{sin\theta}{sin\theta_2}\right) \tag{14.42}$$

where the 2nd factor is due to angular integration in $\Delta\phi$.

14.3 Resonant Cavity Light Emitting Diode Model

The models in Section 14.2 were based on the Green's function approach and much effort was put into deriving a total power extraction formula that sums over all possible modes and over all wavelengths. In this section, we briefly outline the theory we use for the resonant cavity light emitting diode (RCLED). Instead of ignoring the interference or resonance effects of the optical waves, we explicitly consider them. This model is turned on with the **rcled_model** statement.

The task is to start from a continuous dipole source as a function of wavelength and find the emission power spectrum at different angles. If we ignore the micro cavity interference effects, the results should reduce to those in Section 14.2. Since the basic theories have been detailed in Appendix D, we will only highlight a few key points.

RCLED power emission strongly depends on the overlap of the optical field with the dipole source. Existence of the dipole source does not necessarily mean radiative recombination will occur. For example, the simple term Bn^2 only means that if carriers exist in the active layer of a device, a dipole source is available there. It does not guarantee the generation of electron-hole pairs. For that to happen, the optical field should exist so that the dipole moment (displacement times the field) can be formed to complete the energy conversion from current to photon. Therefore, in RCLED model, the simple term Bn^2 (and other form of QW recombination source) must be scaled by an interference enhancement factor (or the g-factor as referred in the simulator print out). In locations of constructive interference, the radiative recombination is enhanced. The opposite holds for locations of destructive interference.

It is interesting to note that the Green's function theory itself is fully consistent with the above view of power emission dependence on location of the dipole source. The Green's function theory states that a point dipole source can cause a field profile (definition of Green's function). However, one can show that the intensity of the field (and thus the emission power) strongly depends on whether the point source is placed at location of constructive or destructive interference. When using APSYS to simulate RCLED, such dipole source and power relation should be kept in mind.

To check whether the active region will be subject to constructive or destructive interference, the simulator will print out a .stw file during the equilibrium calculations. This a text file containing the index profile and standing wave pattern: they can easily be plotted with Gnuplot or other tools.

Another important point to note is that photon recycling effect is automatically taken into account. In the RCLED model, the active layers are responsive to light generated by the dipole source within the same device as well as to external incident light. Due to the photon recycling effect, the total spontaneous emission rate can greatly exceed the electrical pumping rate. The reason is that photons emitted get reflected back to the cavity and cause self optical pumping. Therefore, the correct

way to compute the IQE is to subtract the photon absorption rate from the total spontaneous emission rate before dividing the injection current. This has been done when we plot the IQE for the RCLED model.

14.4 2D Photonic Crystal Power Extraction Model

Our purpose here is to derive the total emission power from a photonic crystal LED. For simplicity, we assume the following general purpose configurations: a 2D photonic crystal (PhC) consisting of holes of low index (such as air) are on top of an LED which we consider as a waveguide with a cross section on the xy-plane.

The basic idea is to regard a 2D photonic crystal (PhC) as a 2nd order DFB grating for which the first order radiative loss has been well understood and formulated [81]. There are major differences between a 2D PhC and a DFB grating which we will identify and handle accordingly.

Without loss of generality, we assume the 2D PhC orientation is arranged such that the shortest period appears in the z-direction. Later on, we shall sum up power emission in other directions. In other words, we only consider power coupling due to the shortest periods in such a structure.

As a reference model, we use the Green's function method to compute the optical power from such a structure. We first cut up the structure into vertical divisions so that emission power can be calculated as a function of such divisions, or a function of position x. Within each division, the problem is simplified into a 1D Green's function in y.

$$P_G(y) = \int G(y, y_1) r_{sp}(y_1) dy_1 \tag{14.43}$$

where G() is a Green's function which is constructed from the products of the optical wave functions (to get the power from wave amplitude). For more details on Green's function approach, please see Appendix D.

It is clear from the form of the Green's function that the amount of power carried by an optical mode is proportional to the overlap between the mode intensity and the spontaneous dipole source profile. A dipole source will not emit power unless the optical mode already existed there (recall dipole moment is product of field and charge displacement). Therefore, we obtain the power distribution fraction for each mode (normalized):

$$P_r(j) = \Gamma_{qw}(j)/sum_j[\Gamma_{qw}(j)] \tag{14.44}$$

In the following discussion, we‘give the formulas for the power coupling, or power emission coefficient for each confined waveguide mode. Detailed derivations may be found in the section on second order grating for PICS3D and also in Ref. [81].

In the major PhC periodic direction, we expand the refractive index in the following form:

$$n(z) = a_0 + a_1 cos(k_b z) + b_1 sin(k_b z) + a_2 cos(2k_b z) + b_2 sin(2k_b z) \tag{14.45}$$

where $k_b = 2\pi/L_b$ and L_b is the period of the major lattice constant of the PhC.

The fraction of surface power emission is given by

$$F_{surf} = \frac{\pi^2 \Delta\varepsilon \zeta^2 Q_f}{n_{mode}^2 \lambda_0} \tag{14.46}$$

where n_{mode} is the modal effective index within the semiconductor wave guide.

Other terms are defined as follows:

$$\zeta = \frac{\sqrt{a_1^2 + b_1^2}}{2} \tag{14.47}$$

$$Q_f = \frac{|\int_{PhC} w(y) exp[-jk_y(y - y_c)]dy|^2}{\int w(y)^2 dy} \tag{14.48}$$

where $\int_{PhC}$ means integration over the PhC air hole layer. y_c is the center point of the air hole layer.

$$k_y = k_0 n_y \tag{14.49}$$

$$n_y = n_{insul} f_{insul} + n_{mode}(1 - f_{insul}) \tag{14.50}$$

where n_{insul} is the index of the insulator (air hole) and f_{insul} is the area fraction of the air hole.

$$\Delta\varepsilon = 4(\Delta n_r^2 + \Delta n_i^2) n_{mode}^2 \tag{14.51}$$

Δn_r and Δn_i are the real and imaginary index differences between the semiconductor and the insulating air hole.

Finally, the difference between a 2nd order DFB grating and a 2D PhC is that the latter has than one major scattering direction. Also, the air hole has a limited width. For these two factors we introduce an optical confinement factor:

$$\Gamma_\phi = M_{dir} W_{insul} / W_{cell} \tag{14.52}$$

where M_{dir} is the number of major scattering directions. For square lattice it takes 2, and for triangle lattice a value of 3 is used. W_{insul} is the width of the insulating air hole (diameter) and W_{cell} is the cell size which we take as the wavelength of interests in the semiconductor medium.

The final emission power for a particular lateral mode j of the LED waveguide is obtained from the power emission from the Green's function method in Equation (14.43) times the modal power fraction in Equation (14.44) times the surface power coupling fraction in Equation (14.46) and times the angular optical confinement factor in Equation (14.52) as follows:

$$P_{PhC,j} = P_G \times P_r(j) \times F_{surf} \times \Gamma_\phi \tag{14.53}$$

14.5 Ray Tracing Simulation

14.5.1 Introduction

Ray tracing (RT) is a modeling technique based on treating light waves as an ensemble of geometrical rays reflected and refracted off material boundaries. This technique is most suitable for optical systems where geometrical nature of the optics is more important than the wave nature. A good example is light emitting diode where the light source is spontaneous emission with randomized phase information while the coherent effect is not significant since the dimension of the cavity is much larger than the wavelength. In this section, we shall describe the theoretical backgrounds of the RT technique in the APSYS software.

A unique feature of our ray tracing technique is the geometrical treatment combined with a wave optics approach, taking into account refraction, reflection, absorption and interference effects. Our model is able to simulate in 2/3 dimensions the angular distribution of the transmitted power of an LED.

14.5.2 Basic ideas

The basic assumption in RT is that a ray of light travels in a straight line within a uniform medium. For simplicity, consider a simple cubic crystal. Let us put in the center a point source of light which emits rays of light in the form of many straight lines. The following can happen to a ray:

- It returns back into crystal completely due to total internal reflection and no fraction of light power goes outside.
- It splits into reflected and transmitted rays.
- It transmits to the outside completely.

Depending on which of these outcomes occurs, there may be a reflected ray and/or a refracted ray which needs to be dealt with (Fig. 14.3). The RT method is based on following the reflected ray until it decays down to a certain numerical precision or escapes from the device completely. Every time a secondary refracted ray is generated, it is put inside a queue for later processing and the program continues until the queue is empty or a maximum number of secondary rays has been generated.

The generation (or not) of a secondary ray is based on the incident angle of the ray, the refractive index on both sides of the interface as well as the particular nature of the material interface. Also, due to the wave nature of light there is a dependence on polarization of the electric and magnetic field vectors with respect to the plane

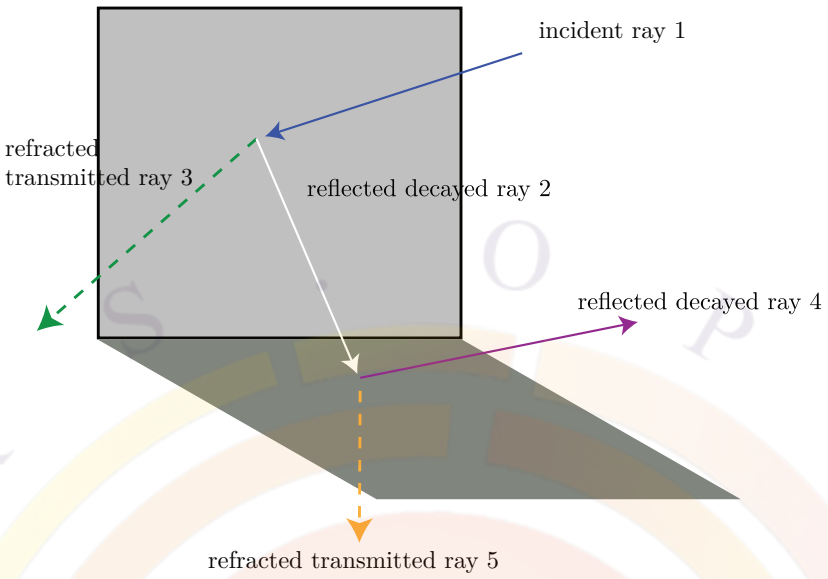


Figure 14.3: Ray tracing in simple cubic crystal

of incidence. If the electric field vector is normal to the plane of incidence, the ray is said to have transverse electric (TE) polarization. On the other hand, if the magnetic field vector is normal to the plane of incidence, the ray is said to have transverse magnetic (TM) polarization.

For example, if our crystal has a refractive index of 3.5 (typical for GaAs), the reflection coefficient (amplitude ratio) of the TE- and TM- polarized light as a function of the angle of incidence can be plotted as Figure 14.4. As you can see if light impinges a surface at an angle greater then around 16.5 degrees in our case, it is totally reflected back. Such an angle is called critical or total internal reflection angle and often limits the amount of light that can be extracted from LEDs.

For a light source from spontaneous emission (as is the case in an LED), the light wave does not have a definite polarization: a ray emitted from such a source should be regarded as having random polarization. To simulate such a source, two approaches can the used: the first is to treat a ray as having random polarization, the second is to treat it as having a 50% TE-portion and a 50% TM-portion. The latter approach was implemented because both of them produce the same results.

14.5.3 Ray tracing geometry

Recall that we started with the idea of a single cubic crystal. Real optoelectronic devices are far more complex and usually consist of many layers and columns. The ray tracing code handles this by defining of the concept of a “box”: the single crystal is the simplest cubic box we can make.

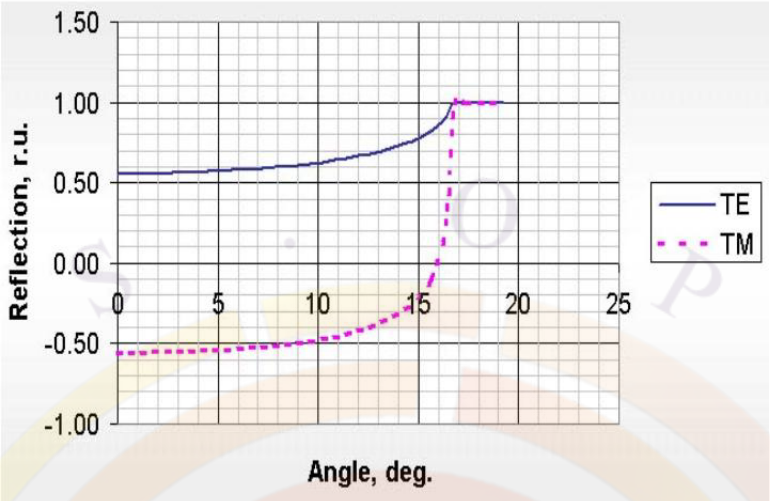


Figure 14.4: The reflection coefficient (amplitude ratio) TE- and TM- polarized light as a function of angle of incidence

APSYS will build boxes for the RT program based on the polygons defined in the .layer and .geo files. If this is a 3D simulation, then these polygons will already have depth information to form 3D boxes. If not, then the 2D polygons can be extruded to a fictitious depth of 1 μm to form 3D objects: some care will have to be taken to respect the original 2D nature of the problem even though we operate inside a 3D volume. Within each box, material properties are averaged to form a uniform propagation medium.

It is difficult to even imagine how a single ray of light undergoes multiple reflection and refraction in a multi-box structure. The situation is more complicated if multiple rays from multiple source points (see Figure 14.5) must be considered. The RT model tracks down all those rays within the multi-box structure until they are either absorbed by the material (down to numerical precision) or emitted outside of the device.

14.5.4 Coherent and incoherent light

As explained in the last section, the ray tracing approach is based on geometric optics and usually ignores coherence and interference effects. However in some situations, this approximation is not valid: we need to identify these situations in order to correct the model and know when it is safe to use it.

We shall start with some discussion about the coherence of light. Based on Fourier transform theories in the analysis related to the WienerŰKhinchin theorem[82], we

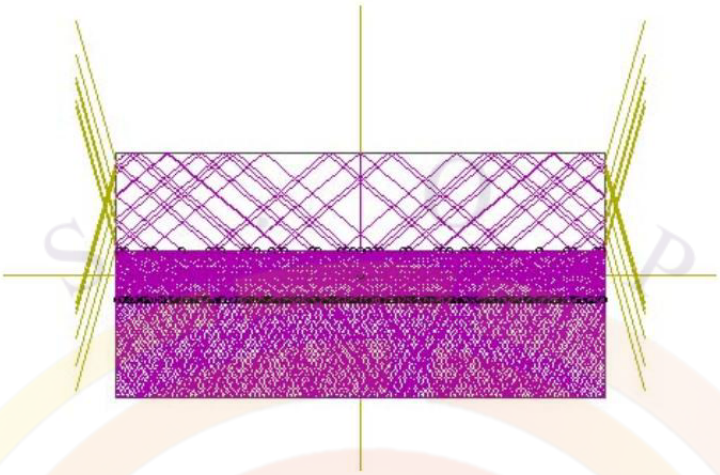


Figure 14.5: The rays undergo multiple reflections/refractions within a device until some of them will are absorbed and others emitted outside.

can express the coherence length of the wave as follows:

$$L = \frac{c}{\Delta f} = -\frac{\lambda^2}{\Delta\lambda} \tag{14.54}$$

where Δf (and thus $\Delta\lambda$) is the spectral line-width of the quasi-monochromatic radiation. The coherence length may be used to judge the suitability of geometric or wave approaches of the simulation.

Take the example of a GaAs LED emitting at 0.85 μm. A typical line-width of the spontaneous emission of the LED is 0.05 μm. The coherence length using the above formula works out to be about 14 μm, which is smaller than the dimension of a typical LED. We thus conclude that geometrical optics approach treating rays as independent and non-interfering waves is valid.

For smaller device features, we need to take into account the polarization of the input field (TE/TM) and the phase effects due to propagation in thin layers. In order to combine this with the geometric approach, such features are implemented as special boundary conditions to boxes. The resultant amplitudes of transmitted waves are converted back into units of power [83] and used for new geometric rays.

For uniform thin film layers, we use a transfer matrix approach for monochromatic plane waves (Ref. [82]) to get an effective reflection and transmission coefficient: this is used to define optical coatings (which are often $\frac{\approx\lambda}{4}$) as well as semi-transparent contacts.

For textured interfaces with large features, we can adjust the geometry of the boxes

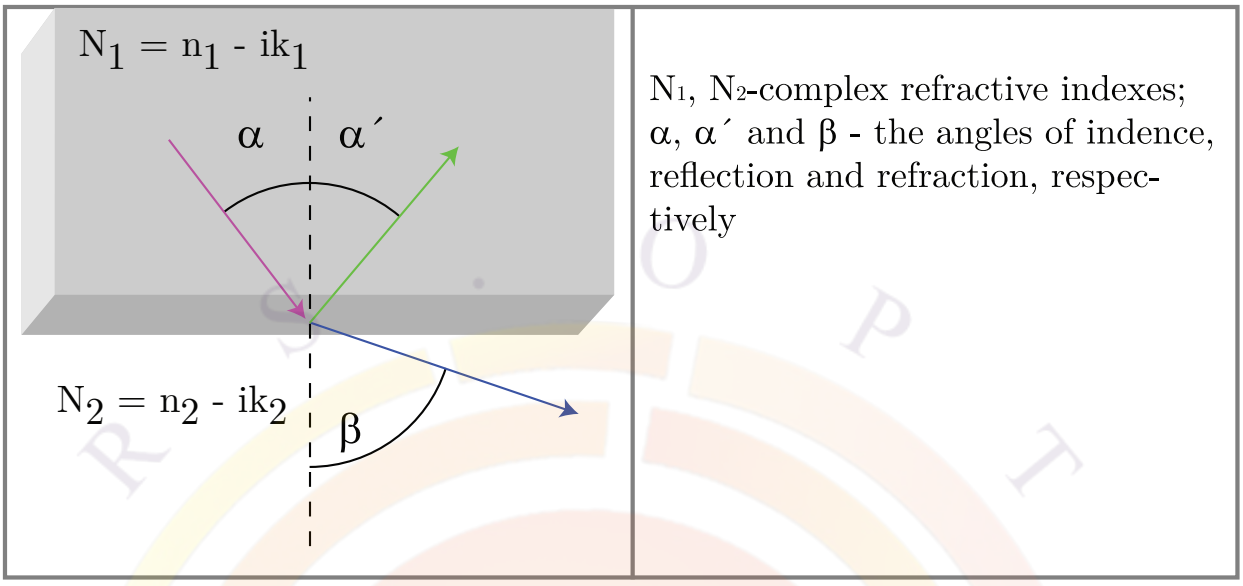


Figure 14.6: Propagation of a light ray from a dense to a rarer medium ($n1 > n2$)

to have the correct shape and work within the geometric optics approximation. However, when the texture is much smaller, we need to rely on other approaches such as FDTD or on experimental scattering results. The effective reflection and transmission coefficient of the interface can then be imported to the ray tracing simulations.

14.5.5 Optical absorption

The optical power carried by a ray traveling in a semiconductor decays due to the absorption by the material. In a thick box, its intensity can be written as a function of distance Z as follows:

$$I = I_0 \exp(-4\pi k Z/\lambda) \tag{14.55}$$

where I_0 is the wave intensity of the light source, k is the extinction coefficient (imaginary part of the complex refractive index) and λ is the wavelength of the light in free space.

14.5.6 Reflection and refraction

Let us consider simple dielectric boundary (Figure 14.6):

The following is Fresnel amplitude reflection and transmission coefficient at oblique incidence for TE-polarized wave [83]:

$$r_e = \frac{N_1 \cos\alpha - N_2 \cos\beta}{N_1 \cos\alpha + N_2 cos\beta} \tag{14.56}$$

$$t_e = \frac{2N_1 \cos\alpha}{N_1 \cos\alpha + N_2 cos\beta} \tag{14.57}$$

for TM-polarized wave:

$$r_m = \frac{N_1 \cos\beta - N_2 \cos\alpha}{N_1 \cos\beta + N_2 cos\alpha} \tag{14.58}$$

$$t_m = \frac{2N_1 \cos\beta}{N_1 \cos\beta + N_2 cos\alpha} \tag{14.59}$$

The resultant amplitude transmission and reflection coefficient t and r are converted into transmittance and reflectance using the expressions:

$$R_e = R_m = rr^* \tag{14.60}$$

$$T_e = \frac{tt^* Re(N_2)\cos\beta}{N_1^* \cos\alpha} \tag{14.61}$$

$$T_m = \frac{tt^* Re(N_2)\cos\alpha}{N_1^* \cos\beta} \tag{14.62}$$

where * denotes complex conjugate, Re - real part of complex numbers. The above power transmittance and reflectance here are used in both approaches mentioned above.

14.5.7 Emission models

A key point to modeling the emission in a LED is that the spontaneous emission is assumed to be isotropic: this means that every emission point must use many rays at various different angles in order to approximate this effect. Every ray is assumed set up to carry a portion of the emission power from the emission point and the rays are divided into equal solid angle regions.

However, the emission is also spread out all over the active region. The most accurate model would be for every point in the active region to emit light in all directions. However, having too many emission points with too many emission rays would pose a great computational burden for the RT program.

As a compromise, the user can either limit the number or rays per emission point or use an emission model that concentrates the emission power into fewer points. These models are detailed in the reference section for **do_raytrace_3d**.

14.5.8 Application to detectors

In previous subsections, we have assumed that the RT method is used to model a LED. However, the model works just the same for solar cells and detectors: the only difference is that original source ray is outside the device rather than inside. This means that in addition to power being lost and transmitted through the device, some it is also reflected from the top surface.

14.5.9 Limitations of the ray tracing method

As mentioned before, using the geometric/ray optics is only adequate where the wave or quantum nature of light can be neglected. Smaller features require some simplifying assumptions or input from other modeling methods.

In the RT method, we also assume that the optical properties are mostly controlled by a single wavelength: the default is to use the peak wavelength of the spontaneous emission spectrum in a LED. Multiple-wavelength simulation can be done by looping over all wavelengths of the spectrum but is very time-consuming.

A second problem occurs when the absorption is too small. Since we follow the reflected ray until it decays to a certain level, this can generate a huge number of secondary rays: unless the reflection coefficient cuts down on the power sufficiently, the reflected ray can keep bouncing around in its box almost indefinitely. To solve this, we do two things:

- A small minimum absorption coefficient is introduced to force rays to decay more quickly in transparent boxes.
- We put a limit on the number of secondary rays that can be generated.

Any power that is lost either to the minimum absorption or by giving up on a ray before it decays is assumed to be "numerically" lost. This kind of artificial loss can also occur due to numerical precision at the interfaces. To correct for this numerical loss, we re-scale all transmitted and absorbed power in the device so that they sum up to 100% of the original source power. If the numerical loss is small, then we can consider the other loss terms to be representative of the physical mechanisms in the device. Please keep this in mind since the re-scaling may be inappropriate if the numerical loss term is large.

Because of this re-scaling process, the RT method cannot be used in devices where there is optical gain: it would violate a key assumption that the transmitted and absorbed power sums up to 100% of the original source term. Attempting to do so would result in various oddities such as a negative % of light absorption in the semiconductor.

The RT code also includes the ability to model encapsulation in a plastic dome (i.e. a cylinder with a hemispherical cap). This is often done to maximize the amount of light that can be extracted by using a material with a refractive index that minimizes the reflection coefficient at the air-dome and semiconductor-dome interfaces. The geometry can also be useful to direct the light emission in a certain direction.

However, this model also suffers from a simple limitation: rays that exit the LED and enter the dome are assumed to stay within the dome and never enter the LED region again. Since the dome is usually much bigger than the LED, this is simply an approximation to save time: it can be time-consuming to see if a ray intercepts any of the LED facets every time there is a reflected ray within the dome.

14.5.10 Conclusions

The ray tracing method is the best way to estimate the extraction efficiency of a LED and its emission pattern, especially for complicated structures that do not behave like a waveguide (as in previous sections). It is best used to:

- Design the LED structures with a certain angular distribution of the emitting power
- Optimize structures to get the most efficient output of the emitting power for LEDs
- Get the most sensitive photodetectors
- Obtain accurate results for structures with complicated geometry
- Discover important physical properties of the optoelectronic devices
- Investigate new optoelectronic devices

14.6 Strained Si or Si/SiGe Quantum Well Model

14.6.1 Introduction

MOSFET designs based on strained Si or Si/SiGe quantum well in the conduction channel have been demonstrated to provide enhanced mobility and thus extend the performance of existing MOS technology[84]. The silicon channel may be strained using a relaxed SiGe material or through stress induced by thin films deposited on the MOS device. The mobility of the strained MOS has been found to depend on the orientation of the strain tensor[85][86]. It is thus necessary to establish a detailed anisotropic band structure model and associate it to the mobility enhancement effect.

This section is dedicated to the discussion of how Crosslight's simulator takes into account the strain dependence in the band structure and mobility of a quantum MOS device.

14.6.2 Band structure and strain orientation

Let us consider the symmetry of silicon crystal when under stress. A common configuration of strained silicon is to use lattice constant difference between SiGe and Si to apply biaxial tensile stress in the plane of $< 001 >$ (x-y plane in real space). This will cause the lower and upper conduction band valleys to split from the other four valleys. We shall label these two valleys as Δ_2 and other four as Δ_4 with the subscript denoting the degeneracy. Such a situation is represented by Fig. 14.7.

It is also common to apply uniaxial stress in the plane of MOS interface grown in (001) direction. Then, the two valleys along the direction of the strain will have different energy than the others and we shall label these two valleys as Δ_2. The other four valleys lying on a plane perpendicular to the direction of uniaxial stress are labeled Δ_4 (see Fig. 14.8).

In the linear approximation, a biaxial strain induces an uniaxial strain of opposite sign and vice versa[87][88]. For uniaxial in the (001) direction, the relationship is:

$$e_b = -D_{001} e_u; D_{001} = 0.771 \tag{14.63}$$

Therefore, if data are available for biaxial strain, it can be converted to those for uniaxial strain.

If the silicon crystal is not gown in (001) direction or if the uniaxial strain is not applied to directions of high symmetry, the six valleys may split into more than two energies. Crosslight's simulator allows a maximum of three different valleys for the conduction band.

The strained band structure parameters and mobilities are less well known for the valence band. In general, the valence band under strain is highly non-parabolic and this prevents a simple definition of curvature mass[87]. Based on a recent theory, the valence band is also anisotropic[89]. It was found that the higher valence band in (001) direction is light hole while the effective mass appeared to be larger in other directions. There was confusion in the literature whether the higher valence band under tensile biaxial strain should be light hole (LH) or heavy hole (HH). For example Richard [89] and Fischetti and Laux [87] labeled the upper valence band as LH while other papers did it otherwise (see for example, [90] and [91]).

At this time, we prefer to treat the strain dependence of the valence band mass as a fitting parameter so that the hole mobility behavior can be understand within the frame work of our mobility model as detailed in later subsections.

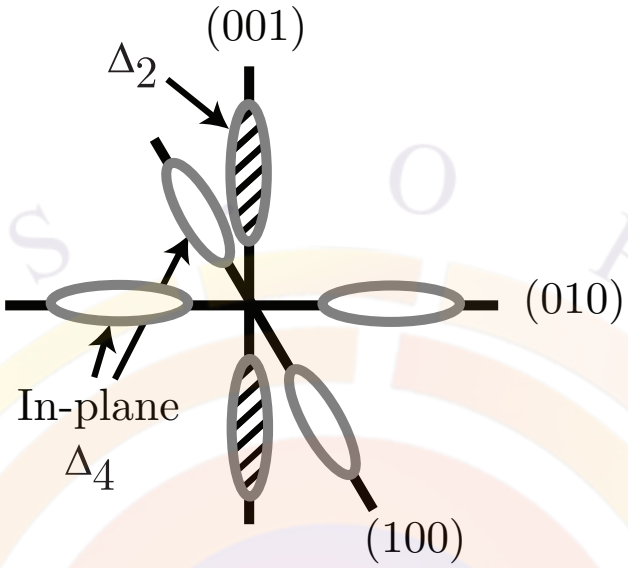


Figure 14.7: Schematics of conduction band valleys of silicon under biaxial in-plane stress.

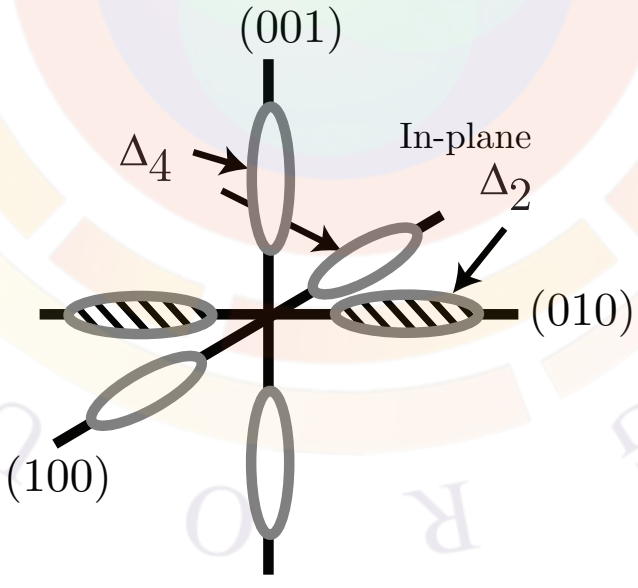


Figure 14.8: Schematics of conduction band valleys of silicon under uniaxial in-plane stress.

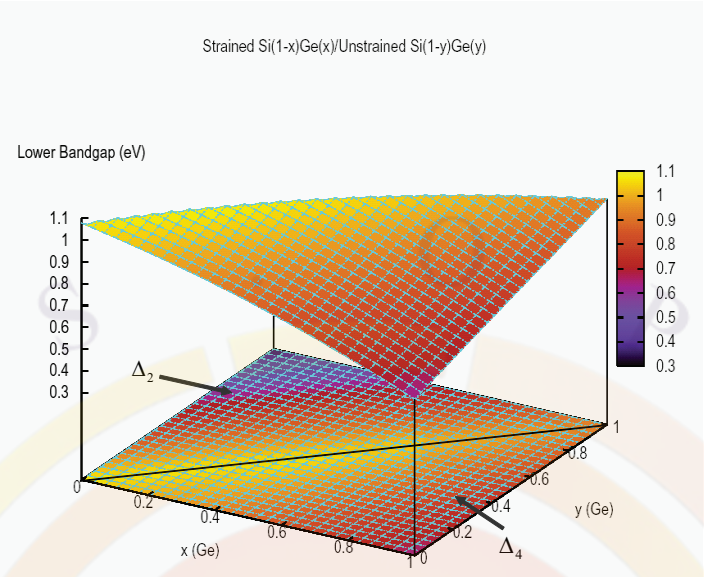


Figure 14.9: Lower bandgap parameterized over a full range of composition for Si(1-x)Ge(x)/relaxed-Si(1-y)Ge(y).

14.6.3 Band structure parameterization

As compared with its strained counterparts in zincblende and wurtzite crystals, the band structure of strained silicon is more difficult to model within the k.p theory[89]. Therefore, we rely on band theories described in the literature and parameterize the band structure as a function of strain or composition over a large range.

A special type of material macro, called the **general complex strain** macro, or **general_cx_strain** macro has been created to provide a tool to describe the strained silicon bands within the effective mass theory. The strained silicon macro allows the specification of up to three conduction band and valence band valleys with anisotropic effective masses.

For strained silicon grown on unstrained SiGe, we mostly depend on the work of Rieger and Vogl [92] for band edges and electron effective masses. Ref. [92] had already parameterize its results using second order polynomial which can easily be incorporated into our macros. The lower and higher bandgaps based on parameterization by Rieger and Vogl[92] are plotted in Figs. 14.10 and 14.10 over a full range of composition of SiGe/relaxed-SiGe. Thus the current model for Si/relaxed-SiGe is just a special case and can easily be extended to research on strained SiGe/relaxed-SiGe. For silicon under uniaxial strain, we mostly rely on theoretical calculation of Fischetti and Laux [87].

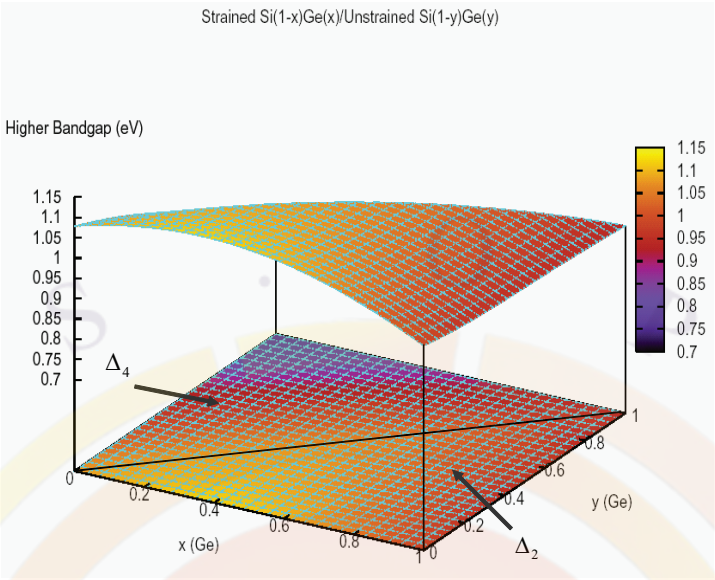


Figure 14.10: Lower bandgap parameterized over a full range of composition for Si(1-x)Ge(x)/relaxed-Si(1-y)Ge(y).

14.6.4 Mobility enhancement

Interest in mobility enhancement by strain has been the driving force behind research in strained silicon. However, the mobility model is also the most complex especially when quantum confinement effect is taken into account. It has been found that for electron mobility in a quantum well (as formed by the potential of the inversion layer in MOS), the mobility can be separated into two parts, one due to intra-valley acoustic phonon scattering and the other due to inter-valley optical phonon scattering[93]. Both parts are complicated functions of subband levels and energy split of the valleys. The complexity of a rigorous theoretical model makes it difficult for experimental data analysis and for CAD purposes.

To reduce the complexity but still maintain the physical intuition needed for understanding mobility enhancement, we propose the following phenomenological model. We start with bulk silicon mobility due to acoustic intra-valley acoustic phonon scattering:

$$\mu_{ac} = \frac{2^{3/2}\sqrt{\pi}\hbar^4\rho\omega_s^2}{3m^{*5/2}D_{ac}^2(k_BT)^{3/2}} \tag{14.64}$$

and that due to inter-valley optical phonon scattering:

$$\mu_{op} = \frac{4\sqrt{2\pi}e\hbar^2\rho sqrt\hbar\omega_0}{3m_d^{*3/2}m^*D_{op}^2}F(x_0) \tag{14.65}$$

where D_{ac} and D_{op} are acoustic and optical phonon deformation potentials, respectively. ρ is crystal density and ω_s and ω_0 are phonon frequencies. F() represents a numerical integral. m_d is the density of state mass and m^* is the conduction mass. For background materials related to the above formulas, please refer to Refs. [94] to [95].

The scattering time constant can be expressed as two terms due to acoustic and optical phonons:

$$1/\tau = 1/\tau_{ac} + 1/\tau_{op} \tag{14.66}$$

and the mobility can be written using the Drude model[96]:

$$\mu = q\tau/m^* \tag{14.67}$$

It is important to notice that the acoustic term and optical term have different dependence on conduction mass. It is commonly believed that mobility enhancement comes from change of conduction mass and suppression of inter-valley optical phonon scattering due to split of valley energies. From energy conservation point of view, carrier transition from one valley to the other via optical phonon scattering will be more difficult as energy difference between valleys are increased. Therefore, we propose the following valley dependent mobility model:

$$1/\mu_k = A_{ac} m_k^{*2.5} + B_{op} m_k^* exp(-\gamma E_{spt}) \tag{14.68}$$

where k is the valley index (denoting one of the six valleys or the HH or LH valleys). A_{ac}, B_{op} and γ are valley independent fitting parameters assume to be independent of the strain. E_{spt} is the valley split between the first and second valleys.

The above formula states that mobility enhancement due to strain comes from two terms: an acoustic term depending on conduction effective mass only and another term depending on effective mass and valley splitting. We shall refer to the above phenomenological model as Crosslight's 3-parameter mobility model.

The 3-parameter model in Eq. (14.68) can be conveniently written as:

$$1/\mu_k = A_{ac}[m_k^{*2.5} + r_{ba} m_k^* exp(-\gamma E_{spt})] \tag{14.69}$$

where $r_{ba} = B_{op}/A_{ac}$ is a ratio representing the relative importance of acoustic and optical terms in the silicon system.

Using our quantum-MOS model, we solve for the distribution of carriers in each valley k as n_k (or p_k for holes) and the total mobility can be expressed as the following:

$$\mu = \frac{\sum_k n_k \mu_k}{\sum_k n_k} \tag{14.70}$$

To set up the 3-parameter model in a simulation, we need to calibrate the fitting parameters using the following procedure:

- 1) Start with a guess of r_{ba} and γ (both of the order 1).
- 2) Set up an unstrained quantum well subband structure preview (quick simulation requiring no mesh) simulation assuming a typical quantum well carrier concentration (say $1 \times 10^{18} cm^{-3}$).
- 3) Compute the valley occupancy and use formula Eq. (14.69) to determine A_{ac}
- 4) Compare with experimental strain dependence of mobility.
- 5) Repeat 1) to 4) until good agreement with experiment.

For hole mobility, until a reliable anisotropic effective mass parameterization is achieved, we need to calibrate the mass-strain dependence in addition to the above procedure.

Such a procedure has been demonstrated and documented in the example chapter related to strained silicon. The calculated mobility enhancement as a function strain shows good agreement with experiment without much fitting effort (see Figs. 14.11 and 14.12). Experimental data here are based on Refs. [97] and [84].

Once a reasonable set of parameters are established for strained silicon, the 3-parameter model can easily be extended to include temperature dependence effect since acoustic and optical phonon terms have different temperature behaviors. Also, as more sophisticated mobility models are available, the simple model can be extended to include subband dependence, say between the first two subbands which are more populated than higher subbands.

14.7 Conduction and Recombination in OLED

A detailed description of conduction carriers in organic semiconductor requires the knowledge of molecular orbitals which can be difficult due to the complexity of the atomic configurations of large organic molecules. However, it is possible to use a simplified energy picture to consider only the highest occupied molecular orbitals (HOMO) describing the electron carriers and the lowest unoccupied molecular orbitals (LUMO) describing the hole carriers.

Similarly, we can introduce the concept of density of states for the HOMO and LUMO so that quasi-Fermi levels and Fermi statistics can be used. Having introduced quasi-Fermi levels, the drift-diffusion equation can be established in a similar way as for conventional semiconductors.

The mechanism of carrier conduction is different than in conventional semiconductor in that conduction is based on a hopping model. In spite of this difference, we can

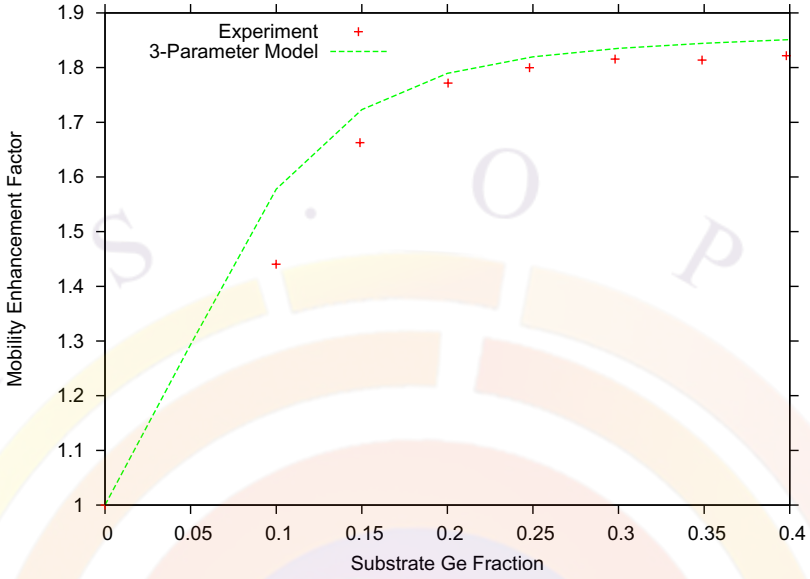


Figure 14.11: Comparison with experiment for Crosslight's 3-parameter mobility model averaged over valley densities in a typical n-MOSFET quantum well.

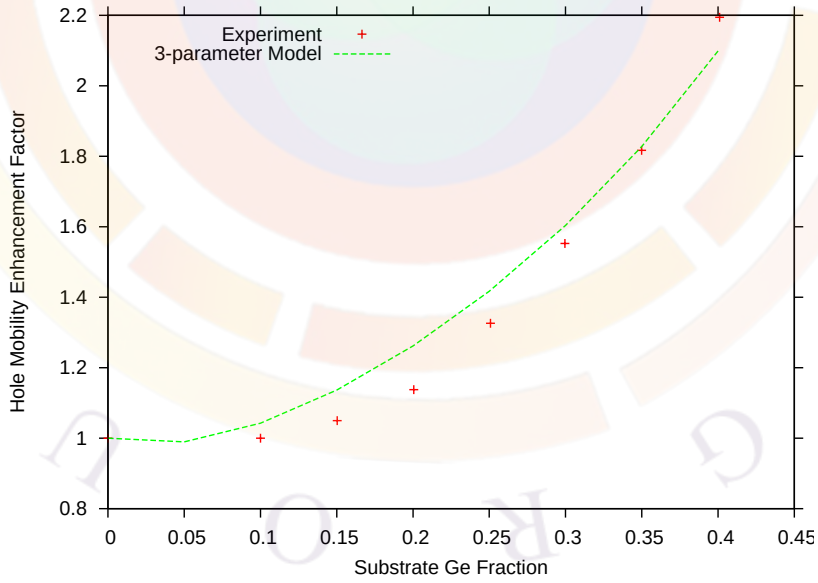


Figure 14.12: Comparison with experiment for Crosslight's 3-parameter mobility model averaged over valley densities in a typical p-MOSFET quantum well.

still approximate the transport using the drift-diffusion equation if we choose a low value for the mobility. In many cases, the carriers may have to go over a high potential barrier and quantum tunneling must be used.

Since the transport is activated by electrical field in a hopping-like model, we use a Poole-Frenkel-like field dependent mobility based on Ref.[98]:

$$\mu = \mu_0 exp(\sqrt{F/F_{pf}}) \tag{14.71}$$

where F_{pf} is an activation field.

The main mechanism of recombination is that electron and holes are attracted to each other via Coulombic forces while the carrier mobility determines the rate at each the recombination occurs. The model is based on the model of Langevin Recombination of carriers (Ref. [99]).

Carriers are statistically independent so the e-h recombination is a random process and kinetically bimolecular. The model considers total current attracted by a charge into a certain volume by Coulombic forces. The bimolecular recombination coefficient depends on the mobility as follows:

$$B = \frac{q}{\varepsilon}(\mu_n + \mu_p) \tag{14.72}$$

For more details, we refer to Ref. [98].

14.8 Organic Semiconductor Emission Spectrum Model

14.8.1 Introduction

Organic semiconductor emits light via Frenkel exciton recombination and the mechanism for absorbing and emitting light is different than conventional semiconductor theories based on free-carrier/many-body interband transition. Optical spectrum theory for organic material is generally rather complicated due to the complexity in molecular structure and the advance physical models involved. However from the view point of engineering design, one does not have to understand all the details related to Frenkel exciton to be able to alter and optimize the spectrum. Simplified exciton models with few adjustable parameters correlating to molecular-crystal structure will enable design engineers to achieve such objectives.

We find that the simplified exciton model based on the Holstein model (see Refs. [100–105]) is suitable for such purposes. The Holstein model assumes a one-dimensional molecular chain with electronic and vibronic interactions between a limited number

of nearest neighbors. The advantage of this model is that one may easily alter the optical spectrum by tuning the few physical parameters. For example if we find increasing the interaction parameter between neighbors alters the optical spectrum in a desirable manner, we may be motivated to alter the real organic material to increase the number of neighboring molecules.

14.8.2 The Model

Following Refs. [100–105], we have established optical spectrum model based on exciton-phonon interaction within an organic crystal. A Holstein Hamiltonian is established to include intra-molecular and inter-molecular electronic and vibronic interactions. A phonon cloud of several unit cells is used to represent exciton-phonon interaction. All excited states of the molecular crystal are solved and optical transition dipole moments computed between all states. Please refer to the above references for details.

Although mechanisms involved in the Hamiltonian are complex, input parameters are few and fitting to experimental spectrum is easy. Typical input/adjustable parameters: exciton bandgap, molecular vibronic quanta, intra-molecular exciton-phonon interaction constant, and inter-molecular hopping parameter. Doped organic semiconductor may easily be modeled as combination of emission from the host and the dopant separately.

14.8.3 Tuning the spectrum

The implementation of the spectrum model is numerically efficient: several minutes per spectrum. We have integrated the spectrum model into APSYS-OLED option so that optical extraction may be calculated based on the spectrum. Bias and current injection dependent spectrum may be simulated to fine tune the color of the OLED.

We use a simple example to provide some insight and a guideline on how to alter the shape of the spectrum. Consider the absorption and EL spectra of Alq3 in Figures 14.13 and 14.14 which were obtained by fitting the EL spectrum to experiments. The smooth and broadened shape corresponds is the results of broadening of tens of spectrum lines as a solution of the Hamiltonian equation. Since the EL spectrum is related to the absorption spectrum, we limit our discussion here to the absorption spectrum which is more simple and it directly correlates to the basic solutions of the Hamiltonian.

To view the actual spectrum lines before broadening, we reduce the Gaussian broadening width (using parameter ox_gaussian_sd1) and obtain the spectrum in Figure 14.15. We see several major lines due to the intra-molecular vibrational excitations. These lines correspond to the major absorption spectrum lines when the unit cell

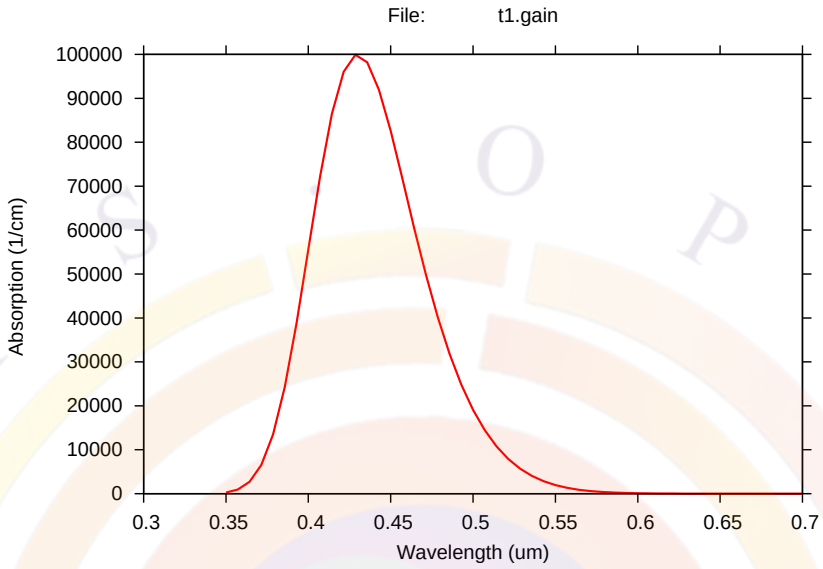


Figure 14.13: Absorption spectrum of Alq3, obtained by fitting exciton parameters with EL spectrum.

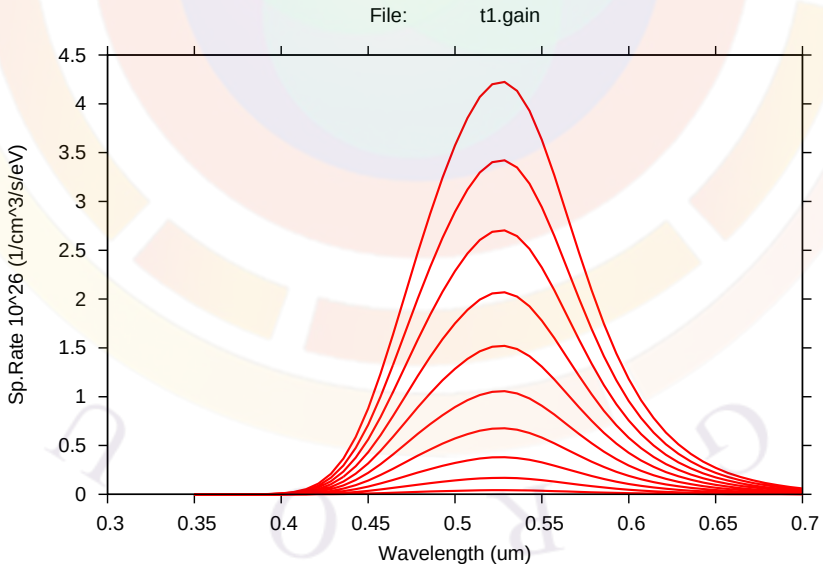


Figure 14.14: EL spectrum of Alq3, obtained by fitting exciton parameters with experimental EL spectrum. Different curves correspond to different carrier injection levels, ranging from 5.E23 to 5.E24 $1/m^3$.

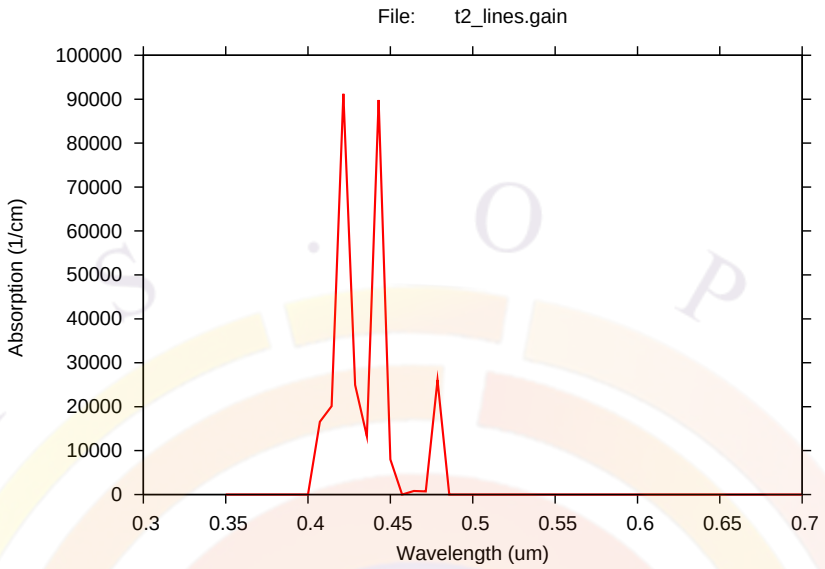


Figure 14.15: Absorption spectrum with reduced Gaussian broadening.

is isolated. We see many lower intensity lines (there are tens of those) arising from electron-electron and electron-phonon interactions within the same unit cell and between the cells.

One way to alter the spectrum is the change the electron-phonon interaction constant (parameter ox_xp_coupling). If we increase the electron-phonon interaction, the spectrum tends to peak in longer wave lengths (see Figure 14.16). The interpretation of such a change in shape is that the vibrational energy is much smaller than the electronic energy and a stronger coupling with phonons pulls down the overall energy of the excitations. To achieve such type of tuning, we may be motivated to alter the Franck-Condon energy of the organic material.

Another way to alter the spectrum is to change the inter-molecule interaction. This can be done by either changing electronic hopping energy (parameter ox_hopping_energy) or by varying the number of interacting nearest neighbors (parameter ox_phonon_cloud). We increase the hopping energy to find that the spectrum tends to peak at higher energies (Figure 14.17).

This can be understood as follows: in the limit of zero inter-molecule interaction, the systems prefers to stay at its lowest vibrational state. Higher inter-molecule interaction simply causes a mixture of all the vibrational states and thus tend to equalize the density of states. For such type of tuning, we may be motivated to alter the real organic material to increase the number of neighboring molecules.

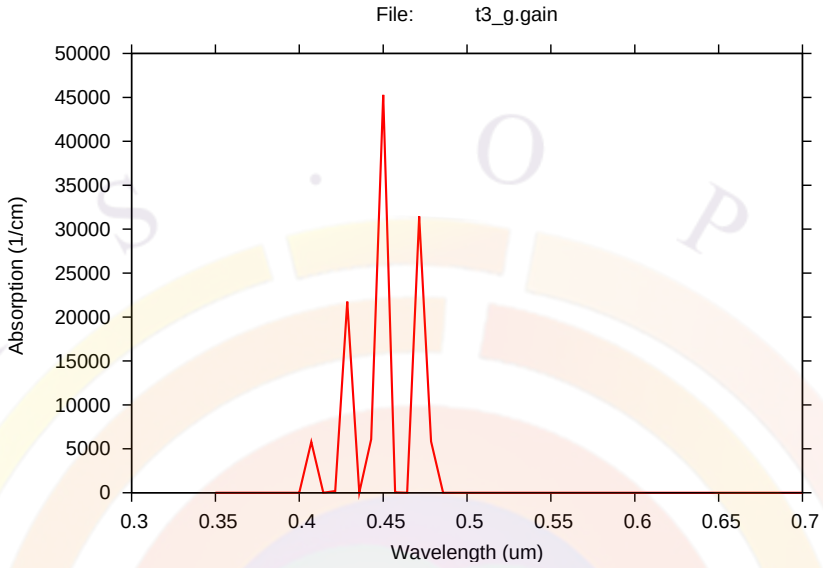


Figure 14.16: Absorption spectrum with increased electron-phonon interaction, showing shift of absorption peak to longer wavelengths.

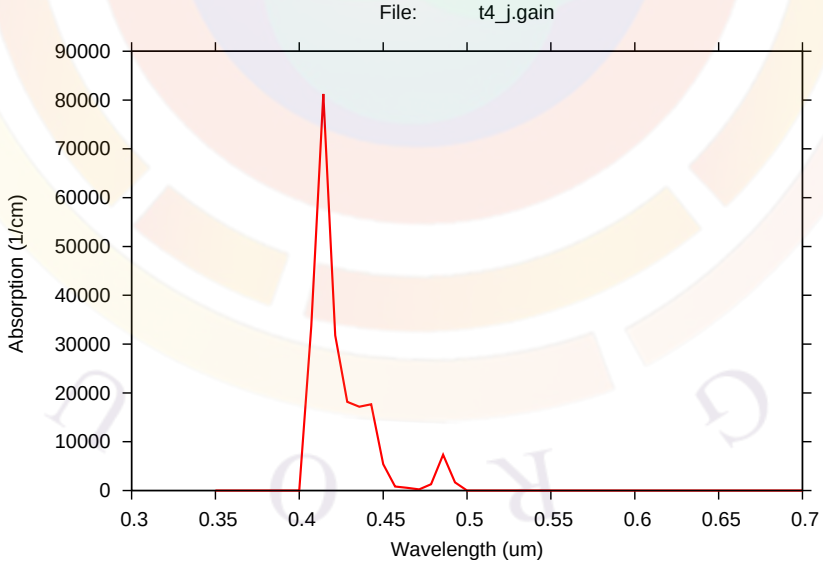


Figure 14.17: Absorption spectrum with increased inter-molecule interaction, showing shift of absorption peak to shorter wavelengths.

14.9 Diffusion of Excitons in Organic Semiconductor

Electron-hole pairs in organic semiconductor form singlet (25%) and triplet excitons (75%). Both exciton types may diffuse over a certain distance before being converted into photons or being quenched via non-radiative means.

We solve the following exciton diffusion equation:

$$\frac{dS}{dt} = R_{rad} + D_s \nabla^2 S - S/\tau_{em} - S/\tau_q \tag{14.73}$$

where S is the singlet or triplet exciton density, R_{rad} is the radiative recombination term from the carrier drift-diffusion equation. S/τ_{em} is photon emission term related to exciton lifetime and S/τ_q is the exciton quenching term. τ_q itself may be a function of S (bi-exciton quenching) and/or the free carrier densities.

The significance of this exciton diffusion model is that OLED do not emit directly from R_{rad} term but through the S/τ_x term above. This means deviation in both magnitude and profile from the bi-molecular recombination term of R_{rad} in the drift-diffusion equation.

Using the RCLED model, the $S(x, y, z)/\tau_x$ term acts as a continuous dipole source within the Green's function theory. The dipole source may be enhanced by optical interference effects to optimize the power emission characteristics of the OLED.

14.9.1 Optical Generation of Excitons

Since light emission in organic semiconductors comes from the excitons rather than the free carriers it stands to reason that optical pumping should not directly produce free carriers either. Instead, the optical generation rate should be added to Eq. (14.73). This can be specified with the ox_pump_only parameter in the **light_power** statement.

14.10 Nomenclature

Symbol	Definition
A_x	Coefficient for band-gap narrowing effects.
a_0, a	Lattice constants of GaAs (a_0) and InGaAs (a).
B	Radiative recombination coefficient.
B_1,B_2,B_z	Complex wave amplitudes of the optical wave propagating from right to left in a laser.

B_k	Constant used in mathematical fittings.
b	Shear deformation potential.
C_n, C_p	Auger recombination coefficients, for electrons (n) and holes (p).
c_{nj}, c_{pj}	Electron (n), hole (p) capture coefficients of the j th deep trap.
c	Velocity of light in vacuum.
c_{11}, c_{12}	Elastic constants.
D_c, D_v	Density of states of the conduction and valence band.
E_D, E_A	Shallow donor (D) and acceptor (A) levels.
E_i^0, E_j^0, E_{ij}^0	Energy minimums of the i th and the j th levels, and the difference between them, . The i refers to the valence band sub-band levels and j refers to the conduction band sub-band levels.
E, E_{ij}	Photon energy and energy difference between the i th and j th levels.
E_{fn}, E_{fp}	Quasi-Fermi energies of electrons (n) and holes (p).
E_g	Bandgap.
E_{g0}	Unstrained bandgap.
ΔE_{PF}	Shift in ionization energy of a dopant due to Poole-Frenkel effect.
δE_{hy}	Hydrostatic strain energy.
δE_{sh}	Energy associated with shear strain.
F	Electric field intensity.
F_l	Lorentzian shape function.
F_{0n},F_{0p}	Threshold electric field used in the electron (n) and hole (p) mobility models.
$F_{1/2}$	Fermi integral of order one-half.
f_D, f_A	Occupancy of the donor (D) and acceptor (A) levels.
f_{tj}	Occupancy of the j th deep trap level.
f_i, f_j	Fermi functions for the i th and the j th level.
f_i', f_j'	Integrated Fermi functions for the i th and the j th levels.
g	Interband local gain.
g_{ij}, g	Local gain due to transition between the i th and the j th levels, and the local gain of the material.
g_d, g_a	Degeneracies of the shallow donor (d) and acceptor (a).
H_{ij}	Hamiltonian matrix element between the i th and the j th states.
$h(x)$	step function of variable x.
h,$\hbar$	Plank's constants.
I_1, I_2	The integrals needed to evaluate the quantum well gain and spontaneous emission rate.
J_n, J_p	Electron (n) and hole (p) current flux densities.
J_{sn}, J_{sp}	Electron (n) and hole (p) current flux densities on the surface.
J_{hn}, J_{hp}	Electron (n) and hole (p) current flux densities across the heterojunction.
J_{source}	Current flux source.
k	Boltzmann constant.
L	Laser cavity length.

$L()$	Lines shape function for gain broadening.
m_0	Electron mass
m_i, m_j, m_{ij}	Relative effective masses of the i th and the j th level and the reduced effective mass between the i th and the j th level. Their relation is defined by $1/m_{ij} = 1/m_i + 1/m_j$. Indices i and j refer here to quantum well sub-bands.
m_n,m_p	Bulk effective masses for electrons (n) and holes (p).
m_e,m_h	The same as m_n and m_p, respectively.
m_{bn},m_{bp}	Bulk effective masses for electrons (n) and holes (p) on the barrier side of the heterojunction.
m_{zl}	Relative effective mass of the L-band used in the quantum level calculation.
m_{vx}, m_{vy}, m_{vz}	Effective hole masses in the x, y and z direction.
M_0, M_{ij}, M_{lh}, M_{hh}	Momentum matrix elements for bulk material, between the i th and the j th states, involving light- and heavy-hole transitions, respectively.
N	Total number of grid points in the simulation space. Also used for the total number of longitudinal layers.
N_b	Number of grid points associated with a boundary of interest.
N_D, N_A	Doping density of shallow donors (D) and shallow acceptors (A).
N_{tj}	Density of the j th deep trap.
n	Electron concentration or density.
n_b	Electron concentration or density on the barrier side of the heterojunction.
n_{b0}	Electron concentration or density on the barrier side of the heterojunction when its quasi-Fermi level coincides with that on the other side.
n_s	Electron concentration or density on the surface.
$\overline{n}$	Real part of refractive index.
n_i	Intrinsic carrier density.
n_{2D}	The surface concentration of a quantum well, equal to the bulk density times the layer thickness.
N_{rn}, N_{rp}	Reference doping density in electron (n) and hole (p) mobility equations.
P_{ij}	Probability of a transition from the i th to the j th level.
p	Hole concentration or density.
p_b	Hole concentration or density on the barrier side of the heterojunction.
p_{b0}	Hole concentration or density on the barrier side of the heterojunction when its quasi-Fermi level coincides with that on the other side.
p_s	Hole concentration or density on the surface.
Q^j_{em}	Energy density of the jth emitted longitudinal mode.

Q_{cav}^{j}	Energy density of the jth longitudinal mode within the laser cavity.
q	Electronic charge.
R_n^{tj}, R_p^{tj}	Electron (n) and hole (p) recombination rates per unit volume through the j th deep trap.
R_{sp}, R_{sp}^{b}, R_{sp}^{qw}	Spontaneous recombination rate per unit volume, also called the radiative recombination rate, and the same quantity in bulk (b) and in the quantum well (qw).
r_{sp}^{qw}	Frequency dependent spontaneous recombination rate for the quantum well.
R_s	Resistance associated with a boundary condition.
R_{st}	Stimulated recombination rate per unit volume.
R_{au}	Auger recombination rate per unit volume.
r_m,r_m^{eff}	Single facet reflectivity and effective reflectivity of a laser.
T	Absolute temperature.
T_{scat}	The scattering kernel for carrier-carrier scattering.
t	Thickness of the quantum well.
V	Electrical potential.
V_s	Electrical potential on the surface.
$V_{applied}$	Applied electrical potential.
v	Volume.
$\overline{v}_n$, $\overline{v}_p$	Average thermal velocity of electrons (n) and holes (p). The average is taken over all three dimensions.
$\overline{v}_n^{therm}$, $\overline{v}_p^{therm}$	Average thermal velocity of electrons (n) and holes (p). The average is taken only over the dimensional of interest.
$\overline{v}_{bn}^{therm}$, $\overline{v}_{bp}^{therm}$	Average thermal velocity of electrons (n) and holes (p) located on the barrier side of the heterojunction. The average is taken only over the dimensional of interest.
v_{sn},v_{sp}	Saturation velocity for electrons (n) and holes (p).
W	Carrier energy.
W_n	Electron energy.
W_n	Hole energy.
W_j^e, W_i^h	Effective width of the wave functions for electron and hole in the jth and ith subbands, respectively.
x, y	Spatial coordinates used in the model. The x-axis is parallel to the quantum well. x is also used elsewhere as the Al composition in AlGaAs or the In composition in InGaAs.
α α_{qw}	Local loss coefficient due to loss other than interband recombination, and local loss in the quantum well.
α_{int}	Internal loss defined as the weighted average of the local loss α.
α_n, α_p	Constants used in doping dependent mobility functions for electrons (n) and holes (p).
α_{fn}, α_{fp}	Coefficients of the free carrier absorption for electrons (n) and holes (p) in the quantum well.

α_0	Absorption coefficient for regions outside the quantum well.
β, β_1, β_2	Complex eigenvalues of the wave equation and its real (1) and imaginary (2) parts. They are also considered effective indices.
α_{em}	Optical loss coefficient due to emission of light in the lasing mode.
α_{emj}	Optical loss coefficient due to emission of light in the jth longitudinal mode.
β_p	Constant used in the hole mobility equation.
Δ	Spin-orbit splitting of the valence band energy.
Δ_{ij}^a	Transition energy shift term in the Asada broadening model.
δ	Constant. Value is 1 for deep donors and 0 for deep acceptors when used to represent electrical charge. Also used as a δ-function.
ϵ_0	Dielectric constant of vacuum.
ϵ_{dc}	Relative DC or low frequency dielectric constant.
ϵ, ϵ_1, ϵ_2	Complex optical dielectric constant and its real (1) and imaginary (2) parts. ϵ is also used, under a completely context, for non-linear gain coefficient.
ϵ_2^g, ϵ_2^α	Imaginary parts of the dielectric constant corresponding to gain due to interband transition and loss due to other mechanisms, respectively.
ε	Strain.
Γ	Energy level broadening parameter, or the half width of the Lorentzian broadening function.
Γ_0	The maximum energy level broadening parameter, or the half width of the Lorentzian broadening function.
Γ_a	The energy level broadening parameter, or the half width of the Lorentzian broadening function, as used in the Asada's broadening model.
$\gamma_1, \gamma_2, \gamma_3$	Luttinger numbers.
γ	Constant relating the optical wave amplitude to electric field.
γ_n,γ_p	Adjustable constants for the electron (n) and hole (p) thermionic emission model, used to account for tunneling and other effects.
γ_{hn},γ_{hp}	Adjustable constants for the electron (n) and hole (p) thermionic emission model for heterojunctions, used to account for tunneling and other effects.
χ	Affinity.
χ	Affinity of a reference material.
λ	Optical wavelength.
λ_j	Optical wavelength of the jth longitudinal mode.
μ_n, μ_p	Mobility of electrons (n) and holes (p).
μ_{0n}, μ_{0p}	Low field electron (n) and hole (p) mobilities.
μ_{1n}, μ_{1p}	Minimum electron (n) and hole (p) mobilities.
μ_{2n}, μ_{2p}	Maximum electron (n) and hole (p) mobilities.
ν	Function used in the Fermi-Direct statistics.

ρ_i, ρ_j, ρ_{ij}	Density of states for the i th and the j th levels, and the reduced density of states for transition between the j th and the i th levels.
ϕ_n,ϕ_p	Quasi-Fermi potential for electrons (n) and holes (p).
ϕ_b	Schottky barrier height.
ρ_i^0, ρ_j^0, ρ_{ij}^0	Constants for the two-dimensional density of states for the i th and the j th levels, and reduced density of states for transition between the i th and the j th levels. For example $\rho_i^0 = m_i/\pi\hbar^2 t$.
σ_{nj}, σ_{pj}	Electron (n) and hole (p) capture cross sections of the j th deep trap.
τ	Intra-band scattering lifetime.
τ_{nj},τ_{pj}	Electron (n) and hole (p) life time due to the jth trap.
τ_h, τ_v	Hole-hole scattering life time for the Asada gain broadening model. $\tau_h = 2\tau_v$.
ω	Angular frequency of the optical wave.
ω_a,ω_β	SOR parameters used for the wave equation.
ξ	Function used in the Fermi-Dirac statistics.
ζ, ζ_n, ζ_p	Variable related to the effective masses. ζ is also used for half the shear strain energy for a different context.
$< \ >$	Average with $\mid W \mid^2$ as weight.

PARS OPTICS GROUP

Chapter 15

LASTIP SPECIFIC MODELS

15.1 Stimulated Emission and Rate Equation

The stimulated emission rate, R_{st} is defined as the rate of photon emission per unit volume as a result of material gain experienced by the traveling light wave. By energy conservation, the emitted photon must cause an increase in the power (P) of the traveling wave by the same amount:

$$dP = dz(\hbar\omega) \int R_{st} dxdy \tag{15.1}$$

Since modal gain g_m relates to the power increase by the following relation:

$$g_m = \frac{1}{P}\frac{dP}{dz} \tag{15.2}$$

we obtain the following expression:

$$g_m P = (\hbar\omega) \int R_{st} dxdy \tag{15.3}$$

Using group velocity ($v_g = \left(\frac{\partial k}{\partial \omega}\right)^{-1}$) as the traveling speed of the optical power, we obtain:

$$P = \hbar\omega v_g S \tag{15.4}$$

where S is the linear photon density (i.e. number of photons in the 2D plane) such that $\int S(z)dz$ is the total number of photons in the laser cavity.

We introduce the normalized 2D optical wave function W so that:

$$\int \mid W \mid^2 dxdy = 1 \tag{15.5}$$

and

$$g_m = \int g(x,y) \mid W \mid^2 dxdy \tag{15.6}$$

Note that the wave function W has been normalized such that the above integral yields the confinement factor Γ if the local gain is uniform in the active region.

From Eqs (15.3) to (15.6), we obtain:

$$\int R_{st} dxdy = \int v_g g(x,y) S \mid W \mid^2 dxdy \tag{15.7}$$

which implies our result:

$$R_{st} = v_g g(x,y) S \mid W \mid^2 \tag{15.8}$$

or, using the local photon density $s(x,y)$:

$$R_{st} = v_g g(x,y) s(x,y) \tag{15.9}$$

In most textbooks, the common formulation is that of Equation (15.8). However, Equation (15.9) is also useful to model multimode behavior : in that case, the local photon density is calculated by summing up the contribution from each mode.

The rate equation for the linear photon density S in the laser cavity may be viewed as a statement of conservation of energy:

$$\begin{array}{rcl} \textit{stimulated emission} & - \textit{internal loss} & - \textit{emitted power} \\ +\textit{spontaneous emission} & = & \textit{power increase} \end{array} \tag{15.10}$$

The "power increase" in the above equation is the rate of linear photon density $(\partial S/\partial t)$. Equation (15.10) can be detailed as

$$\textit{stimulated emission} = \int R_{st} dxdy = v_g g_m S \tag{15.11}$$

Internal loss is defined as the modal loss experienced by the mode of the traveling wave. This loss term is usually denoted by α_i and it can often be determined experimentally. Similar to modal gain, the internal loss can be written as:

$$\alpha_{int} \approx \int \mid W \mid^2 \alpha dxdy \tag{15.12}$$

Defining an effective loss coefficient α_{em} due to emitted power loss from the device, the emitted power in Equation (15.10) can be written in the same form as Equation (15.11):

$$\textit{emitted power} = v_g \alpha_{em} S \tag{15.13}$$

In the special case of a Fabry-Perot cavity, the emitted power loss equals the facet mirror loss and we obtain the following familiar formula:

$$\textit{emitted power} = v_g S \frac{1}{2L} ln \left(\frac{1}{r_{m1} r_{m2}} \right). \tag{15.14}$$

where r_{m1} and r_{m2} are the power reflectivities of the two facets.

The power due to spontaneous emission is simply the integration of the spontaneous emission rate over the cavity:

$$spontaneous\ emission = c_m \int R_{sp} dxdy, \tag{15.15}$$

where c_m is the fraction of spontaneous emission coupled into the lasing mode.

Summation of the above terms yields the familiar rate equation:

$$v_g(g_m - \alpha_{int} - \alpha_{em})S + c_m \int R_{sp} dxdy = \partial S/\partial t \tag{15.16}$$

The derivation given above makes it clear that LASTIP is only concerned with the total number of photons in the cavity so that it can compute the stimulated recombination rate used in the basic equations. The rate equation averages out most details of longitudinal distribution of the optical fields. Therefore the rate equation is expected to be accurate if the z-dimension is uniform.

In cases where strong variation of physical variable along the z-direction, such as DFB and DBR lasers, the rate equation will be inaccurate for a complete description of the laser cavity. However, the 2D rate equation may still be used as a phenomenological model if proper values of effective mirror reflectivity can be chosen. For example, if the mirror reflectivity of the DBR mirror is known, the rate equation in this section, thus the LASTIP software, can still be used.

PARS OPTICS GROUP

Chapter 16

PICS3D-SPECIFIC MODELS I: Longitudinal Modeling

16.1 Introduction

In PICS3D, we consider a 3D model for light-emitting devices. We already discussed 3D electrical modeling in Sec. 6.3 so this chapter will touch on the optical modeling only.

16.2 Basic Equations

Let us start with an edge-emitting wave-guiding structure and define the z axis as the direction of propagation. The basic equation of concern is, of course, the Maxwell equation. However, we would like to go further and obtain a set of equations directly usable in solving the system.

Under the scalar wave assumption, the wave equation can be solved with the technique of variable separation: we assume that the solution to the wave equation can be written as the product:

$$E_\omega(x, y, z) = E_\omega(z)\phi_0(x, y) \tag{16.1}$$

where ω is the optical frequency and z is the direction of the waveguide.

For the moment, let us assume that the laser is operating in a single transverse mode. The modal distribution $\phi_0(x, y)$ can be left out of the formulation and calculated by other means such as the effective index method or the beam propagation method.

The z-dependent part of the electric field $E_\omega(z)$ satisfies the equation:

$$\left[\frac{\partial^2}{\partial z^2} + k^2(z)\right] E_\omega(z) = f_\omega(z) \tag{16.2}$$

where $f_\omega(z)$ is the Langevin noise function due to spontaneous emission. Eq. 16.2 is the basic equation which will be used for the optical wave in the PICS3D package.

We note that the noise function is extremely important for a laser device since it is a driving force for the solution. For an isolated semiconductor laser, the physical solution for Eq. 16.2 would be zero if the spontaneous noise term was absent. A simple physical explanation is that the spontaneous emission noise generates or excites the photons, which are then amplified by the presence of the optical gain. Therefore, the optical power at any bias condition is determined both by the spontaneous emission and the optical gain.

The complex propagation constant $k(z)$ contains information about the solution of the transverse and lateral dimensions. In other words, it is calculated from the effective index by using x-y cross sections at given z positions. The effective index and $k(z)$ both depend on:

- the optical wavelength/frequency
- material properties and device geometry
- bias value and photon density (due to non-linear gain suppression)

In PICS3D, Eq. 16.2 is solved to find the longitudinal modes of the device. The basic solution method will be outlined in this chapter but the numerical implementation in PICS3D uses an alternate derivation shown in Sec. 18.4.

16.3 The Green's Function Method

Using Green's function in the analysis of DFB lasers was first proposed by C.H. Henry [80] and later extensively used by Tromborg *et.al.* [106] in deriving analytical formulas for DFB lasers. PICS3D has used the Green's function method because of its accuracy in treating spontaneous emission, its conceptual simplicity and suitability for numerical implementation.

The Green's function method starts with the wave equation, (16.2). The objective is to obtain a compact expression for the solution to the noise-driven wave equation. In the Green's functions method, the solution to Eq. 16.2 can be written as:

$$E_\omega(z) = \frac{\int_0^l g(z,z')f_\omega(z')dz'}{W} \tag{16.3}$$

where $f_\omega(z)$ is the local Langevin force, $g(z,z')$ the Green's function, and W the Wronskian of the wave equation. The integration is over the diode cavity length l.

The Wronskian W is a *functional* of the *distribution* of the wavenumber $[k(\omega, z)]$ in general.

The interpretation of Eq. 16.3 is very simple. According to the basic principle of the Green's function method, any linear differential equation with a driving source term, has a solution that can be found by decomposing the source into many smaller pieces in space. The final solution is the sum (or integration) of the solutions due to the smaller source terms. In our case, the driving source is the spontaneous noise term and the electric field solution at location z is a sum (or integration) of solutions due to the spontaneous emission of a single photon at location z'. A more detailed discussion of the Green's function method the the treatment of spontaneous emission is outlined in App. D.

The expressions for both the Green's functions and the Wronskian can be written in terms of the solution for the homogeneous wave equation, *i.e.*, the solution to Eq. 16.2 without the spontaneous noise terms. The Green's function can be written as [80][106]:

$$g(z, z') = Z_R(z)Z_L(z')\Theta(z - z') + Z_R(z')Z_L(z)\Theta(z' - z) \ . \tag{16.4}$$

Here $\Theta(z)$ is the Heaviside step function.. $Z_L(z)$ is the solution to the homogeneous wave equation which satisfies the boundary condition at the left laser facet and internal interfaces (for multi-section lasers) but may not satisfy the boundary condition at the right laser facet. $Z_R(z)$ is the corresponding solution which satisfies the boundary condition at the right facet and the internal interfaces.

The Wronskian can be written as [80][106]:

$$W = Z_L(z)\frac{d}{dz}Z_R(z) - Z_R(z)\frac{d}{dz}Z_L(z) \tag{16.5}$$

Since Z_L and Z_R are solutions to the homogeneous wave equation, it follows from simple algebra that $dW/dz = 0$ in each waveguide section. This means that W is position independent. Therefore, under a particular bias condition, the Wronskian is only a function of the frequency or wavelength. This conclusion is very important and it is the basis for the remaining discussion of modeling techniques.

16.4 Longitudinal Modes and Complex Frequencies

The Green's function method gives a clear definition of a longitudinal mode: a longitudinal mode corresponds to a maximum of the emission spectrum which corresponds to a minimum value of $\mid W \mid$. Since the solution must be finite we note that for the

homogeneous wave equation, the Wronskian is exactly zero on a mode peak. In the more general case where the diving force is present, modes are instead found at local minima of the Wronskian. Since we have determined that the Wronskian is only a function of the frequency, the search for longitudinal modes is simplified to a search of the local minima of $W(\omega)$

To evaluate this, we make use of an observation by Tromborg *et.al.* [106] that, with the proper choice of normalization:

$$W(\omega) = 1 - R_g \tag{16.6}$$

where R_g is the complex round trip gain (i.e. phase and amplitude).

In a real cavity, the spontaneous emission driving force is always present. This means that the Wronskian can never be exactly zero and the round-trip gain is always less than one. This can be understood by realizing that spontaneous emission always contributes an additive term to the optical propagation of a wave and that the optical gain is not the only contribution to the laser output. Despite this, the contribution of the spontaneous emission is quite small above threshold and $W = 0$ is usually a good approximation for single-mode lasing behavior. However, multi-mode analysis requires inclusion of the spontaneous emission since in general, side modes are below threshold. We will come back to this point in Sec. 18.2.

Luckily, we can reformulate the problem by treating the additive contribution of the spontaneous emission as an equivalent gain source. Our interpretation here is based on the argument of Henry and Kazarinov [107]. Let us use the convention of $exp(j\omega t)$ and express the contribution of the spontaneous emission as a complex frequency terms. We define this contribution such that when this term is included, the round-trip gain is exactly one. That is, we assert by definition that $W = 0$ on the complex frequency plane for longitudinal mode solutions.

If we consider a simple Fabry-Perot laser of cavity length L, we get:

$$R_1 R_2 exp[(g - \alpha_i)L - 2j\omega n_{eff} L/c_0] = 1. \tag{16.7}$$

Since the frequency is now complex, the imaginary part of the frequency is equivalent to an additional gain term of:

$$\Delta g = 2\omega_{im} n_{eff}/c_0 \tag{16.8}$$

The same argument can be applied to a DFB or any other type of laser structure because we can always divide an arbitrary structure into small sections (ΔL_k) with uniform material variation. We then use transfer matrix techniques to express the round trip gain in terms of the factor $exp[(g-\alpha_i)\Delta L/2 + j\omega n_{eff}\Delta L/c_0]$. By arguing that the round trip gain must remain unity at all times, we reach the same result as in Eq. (16.8).

We quickly note that the addition of this complex frequency also has numerical benefits since it is much easier to search for the zeros of a function than to search for its local minima.

16.5 Coupled Wave Equations

As previously discussed, the wave number $k = k(\omega, z)$ may depend directly on z due to the grating structure or variation in the material composition and indirectly through the carrier density and photon density variations along the waveguide (longitudinal spatial hole burning).

In DFB and DBR lasers, corrugations are made along the waveguides which introduce coupling between the forward and backward waves (also referred to as right going and left going waves). The purpose of this is to perturb the propagation constant $k(z)$ to achieve desirable scattering effects of the propagating waves.

In a laser with a grating of period L_g, the effective refractive index can be written as:

$$n = \tilde{n} + 2(\Delta n)cos\left(\frac{2\pi}{L_g(z)}z + \Omega\right) \tag{16.9}$$

where we consider the general case that the grating period may vary as a function of position. $\tilde{n}$ denotes the slowly varying part of the complex index and $2(\Delta n)$ is the magnitude of the index variation (also a complex quantity).

We define a reference wave number β_0 such that

$$\beta_0 = \frac{\pi}{< L_g >} \tag{16.10}$$

where $<>$ is used to denote the average grating period. β_0 is a constant independent of the frequency and injection condition and is usually set to the Bragg wave number in simple grating structures at threshold condition.

In general, we can assume that the change in the grating period is smooth and write the following expansion:

$$\frac{\pi}{L_g} = \beta_0 + \Delta\beta_{ch}(z) \tag{16.11}$$

where $\Delta\beta_{ch}(z)$ is caused by some form of chirp grating (variation of grating period).

Based on Eq. (16.9), we re-write the wave number as

$$k = \beta_0 + \Delta\tilde{k} + 2\kappa cos[2(\beta_0 + \Delta\beta_{ch})z + j\Omega] \tag{16.12}$$

$$= \beta_0 + \Delta\tilde{k} + \kappa e^{2j(\beta_0+\Delta\beta_{ch})z+j\Omega} + \kappa e^{-2j(\beta_0+\Delta\beta_{ch})z-j\Omega} \tag{16.13}$$

where

$$\Delta\tilde{k} = \tilde{n}\frac{\omega}{c_0} - \beta_0 \tag{16.14}$$

and

$$\kappa = \Delta n\frac{\omega}{c_0} \tag{16.15}$$

Since only the optical frequencies close to the Bragg condition are considered, the second term in Eq. (16.13) is small compared with β_0. Similarly, we assume that the coupling coefficient κ is much smaller than β_0. In the following derivation, we will neglect higher order terms of $\Delta\tilde{k}$ and κ.

We propose a trial solution of the form:

$$E_z = R(z)e^{-j\beta_0 z} + L(z)e^{j\beta_0 z} \tag{16.16}$$

where $L(z)$ and $R(z)$ are used to denote the slow varying amplitudes of waves going left and right.

We substitute it into the wave equation to get the following algebra:

$$\begin{aligned}
&-\beta_0^2\left[R(z)e^{-j\beta_0 z} + L(z)e^{j\beta_0 z}\right] \\
&-2j\beta_0\left[\left(\frac{\partial R(z)}{\partial z}\right)e^{-j\beta_0 z} - \left(\frac{\partial L(z)}{\partial z}\right)e^{j\beta_0 z}\right] \\
&+\left(\frac{\partial^2 R(z)}{\partial z^2}\right)e^{-j\beta_0 z} + \left(\frac{\partial^2 L(z)}{\partial z^2}\right)e^{j\beta_0 z} \\
&+k^2\left(R(z)e^{-j\beta_0 z} + L(z)e^{j\beta_0 z}\right) = 0
\end{aligned} \tag{16.17}$$

Treating $\Delta\tilde{k}$ and κ as small quantities, we expand the wave number k^2 as follows.

$$\begin{aligned}
k^2 \;=\; & \beta_0^2 + 2\beta_0\left(\Delta\tilde{k} + \kappa e^{2j(\beta_0+\Delta\beta_{ch})z+j\Omega} + \kappa e^{-2j(\beta_0+\Delta\beta_{ch})z-j\Omega}\right) \\
& +\textit{higher order terms}.
\end{aligned} \tag{16.18}$$

which is then substituted into Eq. (16.17).

To simplify the notation, we introduce a slow varying function:

$$\Omega_g(z) = 2\Delta\beta_{ch}(z)z + \Omega \tag{16.19}$$

We further assume that the wave amplitude is a slow varying function of z and neglect the second derivatives involving: $\left(\frac{\partial^2 R(z)}{\partial^2 z}\right)$ and $\left(\frac{\partial^2 L(z)}{\partial^2 z}\right)$.

After the terms involving β_0^2 are canceled, we are left with the following equation:

$$\begin{aligned}
&-2j\beta_0\left[\left(\frac{\partial R(z)}{\partial z}\right)e^{-j\beta_0 z} - \left(\frac{\partial^2 R(z)}{\partial^2 z}\right)e^{j\beta_0 z}\right] \\
&+2\beta_0\left(\Delta\tilde{k} + \kappa e^{2j\beta_0 z+j\Omega_g} + \kappa e^{-2j\beta_0 z-j\Omega_g}\right)\left(R(z)e^{-j\beta_0 z} + L(z)e^{j\beta_0 z}\right) \\
=\; & 0.
\end{aligned} \tag{16.20}$$

Since $R(z)$ and $L(z)$ are independent functions, this equation makes sense only when terms with the same propagation factor sum up to zero. In other words, the terms with $e^{-j\beta_0 z}$ and $e^{j\beta_0 z}$ must sum up zero, respectively. Collecting these terms, we find:

$$-2j\beta_0\left(\frac{\partial R(z)}{\partial z}\right)+2\beta_0\Delta\tilde{k}R(z)+2\beta_0\kappa e^{-j\Omega_g}L(z) = 0 \quad (16.21)$$

$$2j\beta_0\left(\frac{\partial L(z)}{\partial z}\right)+2\beta_0\Delta\tilde{k}L(z)+2\beta_0\kappa e^{j\Omega_g}R(z) = 0 \quad (16.22)$$

which can be re-written in the familiar form:

$$\frac{\partial}{\partial z}R(z)+j(\Delta\tilde{k})R(z) = -j\kappa e^{-j\Omega_g}L(z) \quad (16.23)$$

$$\frac{\partial}{\partial z}L(z)-j(\Delta\tilde{k})L(z) = j\kappa e^{j\Omega_g}R(z) \quad (16.24)$$

or in matrix notation:

$$\frac{\partial}{\partial z}\begin{pmatrix}R(z)\\L(z)\end{pmatrix}=\begin{pmatrix}-j(\Delta\tilde{k}) & -j\kappa e^{-j\Omega_g}\\ j\kappa e^{j\Omega_g} & j(\Delta\tilde{k})\end{pmatrix}\begin{pmatrix}R(z)\\L(z)\end{pmatrix} \quad (16.25)$$

16.6 Transfer Matrix Formulas

In this section, the technique developed by McCall and Platzman [108] is used to solve the coupled wave equations.

The complex phase factor $e^{j\Omega_g}$ is separated into real and imaginary parts and the wave equation rewritten as:

$$\frac{\partial}{\partial z}R(z) = -j(\Delta\tilde{k})R(z)+[-j\kappa cos(\Omega_g)-\kappa sin(\Omega_g)]L(z) \quad (16.26)$$

$$\frac{\partial}{\partial z}L(z) = [j\kappa cos(\Omega_g)-\kappa sin(\Omega_g)]R(z)+j(\Delta\tilde{k})L(z) \quad (16.27)$$

We introduce the Pauli matrices :

$$\sigma_1=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \quad (16.28)$$

$$\sigma_2=\begin{pmatrix}0 & -j\\ j & 0\end{pmatrix} \quad (16.29)$$

$$\sigma_3=\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} \quad (16.30)$$

and express the equation in terms of a spinor:

$$\psi = \begin{pmatrix} R(z) \\ L(z) \end{pmatrix} \tag{16.31}$$

We obtain:

$$\begin{aligned} \frac{\partial}{\partial z}\psi &= [-\kappa sin(\Omega_g)\sigma_1 + \kappa cos(\Omega_g)\sigma_2 - j(\Delta\tilde{k})\sigma_3]\psi & (16.32)\\ &= H\psi & (16.33)\\ &= \boldsymbol{\mu}\cdot\boldsymbol{\sigma}\psi & (16.34) \end{aligned}$$

where boldface is used to denote a three component vector.

More specifically, in three component notation:

$$\boldsymbol{\mu} = \begin{bmatrix} -\kappa sin(\Omega_g) & \kappa cos(\Omega_g) & -j(\Delta\tilde{k}) \end{bmatrix} \tag{16.35}$$

and

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{pmatrix} \tag{16.36}$$

An arbitrary function f of the scalar $a + \boldsymbol{b}\cdot\boldsymbol{\sigma}$, linear in the Pauli matrices, can be reduced to another linear function[109]:

$$f(a + \boldsymbol{b}\cdot\boldsymbol{\sigma}) = A + \boldsymbol{B}\cdot\boldsymbol{\sigma} \tag{16.37}$$

where

$$\begin{aligned} A &= \frac{1}{2}[f(a+b) + f(a-b)]; & (16.38)\\ \boldsymbol{B} &= \frac{\boldsymbol{b}}{2b}[f(a+b) - f(a-b)]; & (16.39) \end{aligned}$$

This allows us to expand the solution to Eq. (16.34) which can be written as:

$$\psi(z) = cosh[\mu(z - z_0)] + \frac{\boldsymbol{\mu}\cdot\boldsymbol{\sigma}}{\mu} sinh[\mu(z - z_0)] \tag{16.40}$$

where μ is the complex square root of $\boldsymbol{\mu}\cdot\boldsymbol{\mu}$.

$$\mu^2 = \kappa^2 - (\Delta\tilde{k})^2 \tag{16.41}$$

We note that it makes no difference which sign the complex μ takes.

The transfer matrix that maps ψ from one location to another is given by:

$$\psi(z) = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix} \psi(z_0) \tag{16.42}$$

where

$$T_{11} = cosh[\mu(z-z_0)] - j(\Delta\tilde{k})sinh[\mu(z-z_0)]/\mu \quad (16.43)$$
$$T_{12} = -j\kappa e^{-j\Omega_g} sinh[\mu(z-z_0)]/\mu \quad (16.44)$$
$$T_{21} = j\kappa e^{j\Omega_g} sinh[\mu(z-z_0)]/\mu \quad (16.45)$$
$$T_{22} = cosh[\mu(z-z_0)] + j(\Delta\tilde{k})sinh[\mu(z-z_0)]/\mu \quad (16.46)$$

and

$$\Omega_g(z) = 2\Delta\beta_{ch}(z)z + \Omega \quad (16.47)$$

16.7 Bragg wavelength and related reference parameters

Grating and reference wavelength

In this section, we discuss several important parameters used in simulation of waveguiding structures involving DFB/DBR gratings. In a realistic device involving corrugation gratings, the Bragg wavelength in vacuum is generally not well defined: the spacing or the pitch of the grating is fixed by the fabrication procedure but the modal index of the structure varies with bias and is not known before the simulation

We have chosen to fix the parameter of expansion β_0 as a constant at all times for a particular device in the simulator. This parameter is given by the average grating period L_g as:

$$\beta_0 = \frac{\pi}{< L_g >} \quad (16.48)$$

Thus in theory, β_0 is well defined once the grating structure is known.

However, we often need to calculate parameters in the wavelength or frequency domain so we introduce a reference wavelength:

$$\lambda_{ref} = \frac{2\pi n_{ref}}{\beta_0} = 2n_{ref} < L_g > \quad (16.49)$$

where n_{ref} is a fixed reference index which is normally chosen to be close to the modal index at threshold. In current versions of PICS3D, n_{ref} is the effective index at equilibrium so that λ_{ref} is always independent of the applied bias.

This reference wavelength is different from the Bragg wavelength:

$$\lambda_{Br} = 2n_{eff} < L_g > \quad (16.50)$$

where n_{eff} is the effective index at the current bias value.

Propagation constant

The fundamental assumption in the software for index change is that the **real_index** value in the macro library corresponds to the index at thermal equilibrium. When the device is biased, the index changes due to various effects such as interband transitions. This can be controlled by the **index_model** statement although the most common effects are included by default.

As a result, the propagation constant in the coupled-mode theory is evaluated by:

$$\beta - \beta_0 = (\omega - \omega_{ref}) n_{eff}/c + \omega_{ref}(n_{eff} - n_{ref})/c \tag{16.51}$$

where ω_{ref} is the frequency at λ_{ref}. This is only used in edge-emitting devices since propagation in VCSELs is not based on coupled-mode theory.

Note that during the preview of the round-trip gain provided by **rtgain_phase**, tabulated index change values are used to evaluate the propagation constant. This may differ from results in the main simulation.

Setting the reference wavelength

When setting up a device simulation, there are two ways of defining the reference wavelength in the **longitudinal** statement:

1. Set **ref_wavel** to the value of λ_{ref}
2. Set **ref_index** to the value of n_{ref} and **ref_pitch** to the value of L_g.

The first method is the easiest and ignores the details of the fabrication process. The second method is more accurate but requires knowledge of the reference pitch of the grating. However, n_{ref} must be as close as possible to n_{eff} at threshold in order for the longitudinal modes to be in the right place.

From a device design point of view, the question is often reversed: given a Bragg wavelength, what value of L_g should be used ? To answer this question, an accurate estimate of n_{eff} is needed: it can be obtained from the simulation itself through the 2D mode solver results at various bias points. This will be an iterative process as changing the reference wavelength can alter the threshold value as well as the gain and index profiles.

16.8 Coupled Wave Equations For Second Order Grating

16.8.1 Introduction

When the grating period of the structure is 1/2 of the wavelength, we have a first order grating structure which is commonly used for DFB and DBR lasers. In a first order grating structure, the forward wave will only couple with the backward wave: there are 180 degrees between the two.

A grating structure with a period that equals the wavelength is a second order grating. The forward wave will experience a first order diffraction in 90 degrees which causes radiation loss and a second order diffraction in 180 degrees (or the backward wave). Therefore, we must modify the coupled wave equation to have a term to account for the radiation loss. In subsection 16.8.2, we derive the coupled wave equations here based on Ref. [81]. Subsection 16.8.3 is used to formulate the radiating waves. Subsection 16.8.4 derives the formulas for a commonly used grating structure and compare the results with those in Ref. [81].

16.8.2 Coupling coefficients

To be consistent with our convention for propagating waves, we assume a time dependence of $exp(j\omega t)$ so that the forward wave has a factor $exp(-jk_z z)$.

We choose a reference wave number β_0 such that the grating property can be expanded in term of it. The dielectric function at the optical frequency can be written as:

$$\varepsilon(y,z) = \varepsilon_0(y) + \Delta\varepsilon(y,z) \tag{16.52}$$

where $\Delta\varepsilon(y,z)$ is non zero only in the grating layers:

$$\begin{aligned} \Delta\varepsilon(y,z) &= \Delta\varepsilon[\zeta_{+1}(y)exp(-j\beta_0 z) + \zeta_{-1}(y)exp(+j\beta_0 z) \\ &+ \zeta_{+2}(y)exp(-j2\beta_0 z) + \zeta_{-2}(y)exp(+j2\beta_0 z)] \end{aligned} \tag{16.53}$$

Our starting point is the following wave equation in three dimensions:

$$\left[\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} + \frac{\omega^2}{c_0^2}\varepsilon(y,z)\right] E(y,z) = 0. \tag{16.54}$$

where c_0 is the speed of light.

We consider a propagating field of the following form:

$$E(y,z) = [R(z)exp(-j\beta_0 z) + L(z)exp(j\beta_0 z)]\phi(y) + \Delta E(y,z) \tag{16.55}$$

where $\Delta E(y,z)$ is the radiating wave.

The wave equation becomes:

$$\left[\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}+\frac{\omega^2}{c_0^2}\varepsilon_0(y)\right.$$
$$+\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{+1}(y)exp(-j\beta_0 z)+\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{-1}(y)exp(+j\beta_0 z)$$
$$\left.+\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{+2}(y)exp(-j2\beta_0 z)+\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{-2}(y)exp(+j2\beta_0 z)\right]E(y,z)=0 \quad (16.56)$$

Let us substitute the expression for E(y,z) and work out all the terms as follows:

$$\left[\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}+\frac{\omega^2}{c_0^2}\varepsilon_0(y)\right.$$
$$+\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{+1}(y)exp(-j\beta_0 z)+\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{-1}(y)exp(+j\beta_0 z)$$
$$\left.+\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{+2}(y)exp(-j2\beta_0 z)+\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{-2}(y)exp(+j2\beta_0 z)\right]$$
$$[R(z)exp(-j\beta_0 z)\phi(y)+L(z)exp(j\beta_0 z)\phi(y)+\Delta E(y,z)]=0. \quad (16.57)$$

If we collect terms of different exponential factors, we obtain the following equations for terms with no exponential wave factor:

$$\left[\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}+\frac{\omega^2}{c_0^2}\varepsilon_0(y)\right]\Delta E(y,z)=-\frac{\omega^2}{c_0^2}\Delta\varepsilon[\zeta_{+1}(y)L(z)+\zeta_{-1}(y)R(z)]\phi(y) \quad (16.58)$$

and for terms with $exp(-j\beta_0 z)$:

$$\left[\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}+\frac{\omega^2}{c_0^2}\varepsilon_0(y)\right]R(z)exp(-j\beta_0 z)\phi(y)$$
$$=-\frac{\omega^2}{c_0^2}\Delta\varepsilon\left[\zeta_{+1}(y)\Delta E(y,z)+\zeta_{+2}(y)L(z)\phi(y)\right]exp(-j\beta_0 z) \quad (16.59)$$

We drop the terms with $\frac{\partial^2 R}{\partial z^2}$ and $\frac{\partial^2 L}{\partial z^2}$:

$$\left[\frac{\partial^2}{\partial y^2}-2j\beta_0\frac{\partial}{\partial z}-\beta_0^2+\frac{\omega^2}{c_0^2}\varepsilon_0(y)\right]R(z)\phi(y)$$
$$=-\frac{\omega^2}{c_0^2}\Delta\varepsilon[\zeta_{+1}(y)\Delta E(y,z)+\zeta_{+2}(y)L(z)\phi(y)] \quad (16.60)$$

Similarly for terms with $exp(j\beta_0 z)$:

$$\left[\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}+\frac{\omega^2}{c_0^2}\varepsilon_0(y)\right]L(z)exp(j\beta_0 z)\phi(y)$$
$$= -\frac{\omega^2}{c_0^2}\Delta\varepsilon[\zeta_{-1}(y)\Delta E(y,z)+\zeta_{-2}(y)R(z)\phi(y)]exp(j\beta_0 z) \quad (16.61)$$

or

$$\left[\frac{\partial^2}{\partial y^2}+2j\beta_0\frac{\partial}{\partial z}-\beta_0^2+\frac{\omega^2}{c_0^2}\varepsilon_0(y)\right]L(z)\phi(y)$$
$$= -\frac{\omega^2}{c_0^2}\Delta\varepsilon[\zeta_{-1}(y)\Delta E(y,z)+\zeta_{-2}(y)R(z)\phi(y)] \quad (16.62)$$

The left hand side of equation Eq. (16.58) is very familiar to us. Assuming a fast varying factor $exp(\pm j\beta_0 z)$ for $\Delta E(y,z)$, we solve the following Green's function for it y-dependence:

$$[\frac{\partial^2}{\partial y^2}-\beta_0^2+\frac{\omega^2}{c_0^2}\varepsilon_0(y)]G(y,y_1)=\delta(y-y_1) \quad (16.63)$$

$$\Delta E(y,z)=-\frac{\omega^2}{c_0^2}\Delta\varepsilon\int G(y,y_1)[\zeta_{+1}(y_1)L(z)+\zeta_{-1}(y_1)R(z)]\phi(y_1)dy_1 \quad (16.64)$$

$$\Delta E(y,z) = -\frac{\omega^2}{c_0^2}\Delta\varepsilon\int G(y,y_1)\zeta_{+1}(y_1)\phi(y_1)]dy_1L(z)$$
$$-\frac{\omega^2}{c_0^2}\Delta\varepsilon\int G(y,y_1)\zeta_{-1}(y_1)\phi(y_1)dy_1R(z) \quad (16.65)$$

We note that $\phi(y)$ satisfies the following equation for a single transverse mode:

$$\left[\frac{\partial^2}{\partial y^2}-\beta_0^2+\frac{\omega_0^2}{c_0^2}Re[\varepsilon_0(y)]\right]\phi(y)=0 \quad (16.66)$$

This allows us to re-write Eq. (16.60) as follows:

$$\left[\frac{\omega^2}{c_0^2}\varepsilon_0(y)-\frac{\omega_0^2}{c_0^2}\varepsilon_0(y)+\frac{\omega^2}{c_0^2}Im[\varepsilon_0(y)]-2j\beta_0\frac{\partial}{\partial z}\right]R(z)\phi(y)$$
$$= -\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{+1}(y)\left[\frac{\omega^2}{c_0^2}\Delta\varepsilon\int G(y,y_1)\zeta_{+1}(y)\phi(y_1)]dy_1L(z)\right.$$
$$\left.-\frac{\omega^2}{c_0^2}\Delta\varepsilon\int G(y,y_1)\zeta_{-1}(y)\phi(y_1)dy_1R(z)\right]$$
$$-\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{+2}(y)L(z)\phi(y) \quad (16.67)$$

We multiply the above with $\phi(y)$ and integrate over y:

$$\begin{aligned}
&\left\{\beta^2-\beta_0^2+j\frac{\omega_0^2}{c_0^2}\int\phi(y)Im[\varepsilon_0(y)]\phi(y)dy-2j\beta_0\frac{\partial}{\partial z}\right\}R(z)\\
=&-\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\int\int\phi(y)\zeta_{+1}(y)G(y,y_1)\zeta_{+1}(y_1)\phi(y_1)dydy_1L(z)\\
-&\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\int\int\phi(y)\zeta_{+1}(y)G(y,y_1)\zeta_{-1}(y_1)\phi(y_1)dydy_1R(z)\\
-&\frac{\omega^2}{c_0^2}\Delta\varepsilon\int\zeta_{+2}(y)\phi(y)^2dyL(z)
\end{aligned}\tag{16.68}$$

where

$$\begin{aligned}
\beta^2-\beta_0^2&=2\beta_0\frac{\partial\beta}{\partial\omega}\Delta\omega\\
&=2\beta_0\frac{\Delta\omega}{v_g}
\end{aligned}\tag{16.69}$$

We also note that we can use modal gain and internal loss to express the following integral:

$$\int\phi(y)Im[\varepsilon_0(y)]\phi(y)dy=\frac{n'c_0}{\omega_0}(g-\alpha_i)\tag{16.70}$$

Then:

$$\begin{aligned}
&\left\{2\beta_0\frac{\Delta\omega}{v_g}+j\beta_0(g-\alpha_i)-2j\beta_0\frac{\partial}{\partial z}\right.\\
+&\left.\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\int\int\phi(y)\zeta_{+1}(y)G(y,y_1)\zeta_{-1}(y_1)\phi(y_1)dydy_1\right\}R(z)\\
=&-\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\int\int\phi(y)\zeta_{+1}(y)G(y,y_1)\zeta_{+1}(y_1)\phi(y_1)dydy_1L(z)\\
-&\frac{\omega^2}{c_0^2}\Delta\varepsilon\int\zeta_{+2}(y)\phi(y)^2dyL(z)
\end{aligned}\tag{16.71}$$

A similar derivation can be made for Eq. (16.62):

$$\begin{aligned}
&\left[\frac{\omega^2}{c_0^2}\varepsilon_0(y)-\frac{\omega_0^2}{c_0^2}\varepsilon_0(y)+2j\beta_0\frac{\partial}{\partial z}+j\frac{\omega^2}{c_0^2}Im[\varepsilon_0(y)]\right]L(z)\phi(y)\\
=&-\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{-1}(y)\Delta E(y,z)\\
-&\frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{-2}(y)R(z)\phi(y)
\end{aligned}\tag{16.72}$$

$$\begin{aligned}
&\left[\frac{\omega^2}{c_0^2}\varepsilon_0(y) - \frac{\omega_0^2}{c_0^2}\varepsilon_0(y) + 2j\beta_0\frac{\partial}{\partial z} + j\frac{\omega^2}{c_0^2}Im[\varepsilon_0(y)]\right]L(z)\phi(y) \\
= & -\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\int\zeta_{-1}(y)G(y,y_1)\zeta_{+1}(y_1)\phi(y_1)dy_1L(z) \\
- & \left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\int\zeta_{-1}(y)G(y,y_1)\zeta_{-1}(y_1)\phi(y_1)dy_1R(z) \\
- & \frac{\omega^2}{c_0^2}\Delta\varepsilon\zeta_{-2}R(z)\phi(y) \qquad (16.73)
\end{aligned}$$

We multiply $\phi(y)$ and integrate over y:

$$\begin{aligned}
&\left[\beta^2 - \beta_0^2 + 2j\beta_0\frac{\partial}{\partial z} + j\frac{\omega^2}{c_0^2}\int\phi(y)Im[\varepsilon_0(y)]\phi(y)dy\right]L(z) \\
= & -\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\int\int\phi(y)\zeta_{-1}(y)G(y,y_1)\zeta_{+1}(y_1)\phi(y_1)dydy_1L(z) \\
- & \left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\int\int\phi(y)\zeta_{-1}(y)G(y,y_1)\zeta_{-1}(y_1)\phi(y_1)dydy_1R(z) \\
- & \frac{\omega^2}{c_0^2}\Delta\varepsilon\int\zeta_{-2}(y)\phi(y)^2dyR(z) \qquad (16.74)
\end{aligned}$$

$$\begin{aligned}
&\left\{2\beta_0\frac{\Delta\omega}{v_g} + 2j\beta_0\frac{\partial}{\partial z} + j\frac{\omega}{c_0}n'(g-\alpha_i)\right. \\
+ & \left.[\frac{\omega^2}{c_0^2}\Delta\varepsilon]^2\int\int\phi(y)\zeta_{-1}(y)G(y,y_1)\zeta_{+1}(y_1)\phi(y_1)dydy_1\right\}L(z) \\
= & -\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\int\int\phi(y)\zeta_{-1}(y)G(y,y_1)\zeta_{-1}(y_1)\phi(y_1)dydy_1R(z) \\
- & \frac{\omega^2}{c_0^2}\Delta\varepsilon\int\zeta_{-2}(y)\phi(y)^2dyR(z) \qquad (16.75)
\end{aligned}$$

Our key results are of the following form:

$$\begin{aligned}
&\left\{\frac{\Delta\omega}{v_g}+j\frac{g-\alpha_i}{2}-j\frac{\partial}{\partial z}\right.\\
+\;&\left.\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\frac{1}{2\beta_0}\int\int\phi(y)\zeta_{+1}(y)G(y,y_1)\zeta_{-1}(y_1)\phi(y_1)dydy_1\right\}R(z)\\
+\;&\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\frac{1}{2\beta_0}\int\int\phi(y)\zeta_{+1}(y)G(y,y_1)\zeta_{+1}(y_1)\phi(y_1)dydy_1L(z)\\
+\;&\frac{\omega^2}{c_0^2}\Delta\varepsilon\frac{1}{2\beta_0}\int\zeta_{+2}\phi(y)^2dyL(z)\\
=\;&0
\end{aligned}\tag{16.76}$$

$$\begin{aligned}
&\left\{\frac{\Delta\omega}{v_g}+j\frac{\partial}{\partial z}+j\frac{g-\alpha_i}{2}\right.\\
+\;&\left.\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\frac{1}{2\beta_0}\int\int\phi(y)\zeta_{-1}(y)G(y,y_1)\zeta_{+1}(y_1)\phi(y_1)dydy_1\right\}L(z)\\
+\;&\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\frac{1}{2\beta_0}\int\int\phi(y)\zeta_{-1}(y)G(y,y_1)\zeta_{-1}(y_1)\phi(y_1)dydy_1R(z)\\
+\;&\frac{\omega^2}{c_0^2}\Delta\varepsilon\frac{1}{2\beta_0}\int\zeta_{-2}(y)\phi(y)^2dyR(z)\\
=\;&0
\end{aligned}\tag{16.77}$$

These are rewritten as

$$\left\{\frac{\partial}{\partial z}+j\frac{\Delta\omega}{v_g}-\frac{g-\alpha_i}{2}+h_{1a}\right\}R(z)+h_{1b}L(z)+jh_{2b}L(z)=0\tag{16.78}$$

$$\left\{\frac{\partial}{\partial z}-j\frac{\Delta\omega}{v_g}+\frac{g-\alpha_i}{2}-h_{1a}\right\}L(z)-h_{1c}R(z)-jh_{2a}R(z)=0\tag{16.79}$$

$$h_{1a}=-j\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\frac{1}{2\beta_0}\int\int\phi(y)\zeta_{+1}(y)G(y,y_1)\zeta_{-1}(y_1)\phi(y_1)dydy_1\tag{16.80}$$

$$h_{1b}=-j\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\frac{1}{2\beta_0}\int\int\phi(y)\zeta_{+1}(y)G(y,y_1)\zeta_{+1}(y_1)\phi(y_1)dydy_1\tag{16.81}$$

$$h_{1c}=-j\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2\frac{1}{2\beta_0}\int\int\phi(y)\zeta_{-1}(y)G(y,y_1)\zeta_{-1}(y_1)\phi(y_1)dydy_1\tag{16.82}$$

$$h_{2a} = \frac{\omega^2}{c_0^2}\Delta\varepsilon\frac{1}{2\beta_0}\int \zeta_{-2}(y)\phi(y)^2 dy \tag{16.83}$$

$$h_{2b} = \frac{\omega^2}{c_0^2}\Delta\varepsilon\frac{1}{2\beta_0}\int \zeta_{+2}(y)\phi(y)^2 dy \tag{16.84}$$

The Green's function can be written in terms of the solutions of the defining equation with the right hand side equal to zero. These solutions are written as $\phi(y)_+$ and $\phi(y)_-$, which are the radiating waves to the positive and negative direction of y, respectively:

$$G(y, y_1) = \frac{1}{W}\begin{cases}\phi(y)_+\phi(y_1)_- & y > y_1\\ \phi(y)_-\phi(y_1)_+ & y < y_1\end{cases} \tag{16.85}$$

where the Wronskian is given by:

$$W = \frac{d\phi(y)_+}{dy}\phi(y)_- - \frac{d\phi(y)_-}{dy}\phi(y)_+ \tag{16.86}$$

For simplicity, we assume the radiating waves are plane waves in y direction with dependence $exp(\pm jk_y y)$. Then,

$$W = -2jk_y \tag{16.87}$$

$$G(y, y_1) = \frac{1}{W}\begin{cases}exp[-jk_y(y - y_1)] & y > y_1\\ exp[-jk_y(y_1 - y)] & y < y_1\end{cases} \tag{16.88}$$

or

$$G(y, y_1) = -\frac{1}{2jk_y}\begin{cases}cos[-k_y(y - y_1)] + jsin[-k_y(y - y_1)] & y > y_1\\ cos[-k_y(y_1 - y)] + jsin[-k_y(y_1 - y)] & y < y_1\end{cases} \tag{16.89}$$

For simplicity, we only consider the case of symmetric grating with $\zeta_{+m} = \zeta_{-m}$. Then $h_{1a} = h_{1b} = h_{1c} = h_1$ and $h_{2a} = h_{2b} = h_2$.

We consider the following integral:

$$\begin{aligned}
& h_1 4\beta_0 k_y \left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^{-2} \\
= & \int_{y>y_1}\int \phi(y)\zeta_1(y)\{cos[-k_y(y-y_1)]+jsin[-k_y(y-y_1)]\}\zeta_1(y_1)\phi(y_1)dydy_1 \\
+ & \int_{y<y_1}\int \phi(y)\zeta_1(y)\{cos[-k_y(y_1-y)]+jsin[-k_y(y_1-y)]\}\zeta_1(y_1)\phi(y_1)dydy_1 \\
= & \int_{y>y_1}\int \phi(y)\zeta_1(y)\{cos[-k_y(y-y_1)]+jsin[-k_y(y-y_1)]\}\zeta_1(y_1)\phi(y_1)dydy_1 \\
+ & \int_{y<y_1}\int \phi(y)\zeta_1(y)\{cos[-k_y(y_1-y)]+jsin[-k_y(y-y_1)]\}\zeta_1(y_1)\phi(y_1)dydy_1 \\
+ & \int_{y<y_1}\int \phi(y)\zeta_1(y)\{-jsin[-k_y(y-y_1)]+jsin[-k_y(y_1-y)]\}\zeta_1(y_1)\phi(y_1)dydy_1 \\
= & \int\int \phi(y)\zeta_1(y)exp[-jk_y(y-y_1)]\zeta_1(y_1)\phi(y_1)dydy_1 \\
- & 2j\int_{y_1>y}\int \phi(y)\zeta_1(y)sin[k_y(y_1-y)]\zeta_1(y_1)\phi(y_1)dydy_1 \\
= & \left|\int \phi(y)\zeta_1(y)exp[-jk_y y]\right|^2 dy \\
- & 2j\int_{y_1>y}\int \phi(y)\zeta_1(y)sin[k_y(y_1-y)]\zeta_1(y_1)\phi(y_1)dydy_1 \qquad (16.90)
\end{aligned}$$

In many cases, the grating layer thickness is much smaller than the wavelength. As a result, the $\zeta(y)$ function is much like a delta-function in y. Therefore, the 2nd term in the above equation is negligible and we obtain the following:

$$h_1 = \left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2 \frac{1}{4\beta_0 k_y} \mid \int \phi(y)\zeta_1(y)exp(-jk_y y)dy \mid^2 \qquad (16.91)$$

$$h_2 = \frac{\omega^2}{c_0^2}\Delta\varepsilon\frac{1}{2\beta_0}\int \zeta_2(y)\phi(y)^2 dy \qquad (16.92)$$

The above equations can be further simplified assuming $\beta_0 = k_y = \frac{n\omega}{c} = \frac{2\pi n}{\lambda}$ $\Delta\varepsilon = 2n\Delta n$.

The factor for h_1 becomes:

$$\left(\frac{\omega^2}{c_0^2}\Delta\varepsilon\right)^2 \frac{1}{4\beta_0 k_y} = \frac{\omega^2}{c_0^2}(\Delta\varepsilon)^2\frac{1}{4n^2} = \left(\frac{2\pi}{\lambda}\right)^2(\Delta n)^2 \qquad (16.93)$$

The factor for h_2 becomes

$$\frac{\omega^2}{c_0^2}\Delta\varepsilon\frac{1}{2\beta_0} = \frac{\omega}{c_0}2n\Delta n\frac{1}{2n} = \frac{2\pi}{\lambda}\Delta n \qquad (16.94)$$

Our final results are as follows:

$$h_1 = \left(\frac{2\pi}{\lambda}\right)^2 (\Delta n)^2 \mid \int \phi(y)\zeta_1(y) exp(-jk_y y) dy \mid^2 \tag{16.95}$$

$$h_2 = \frac{2\pi}{\lambda} \Delta n \int \zeta_2(y)\phi(y)^2 dy \tag{16.96}$$

For a symmetric grating structure, the coupled wave equations are as follows:

$$\left\{\frac{\partial}{\partial z} + j\frac{\Delta\omega}{v_g} - \frac{g-\alpha_i}{2} + h_1\right\} R(z) + h_1 L(z) + jh_2 L(z) = 0 \tag{16.97}$$

$$\left\{\frac{\partial}{\partial z} - j\frac{\Delta\omega}{v_g} + \frac{g-\alpha_i}{2} - h_1\right\} L(z) - h_1 R(z) - jh_2 R(z) = 0 \tag{16.98}$$

For a real index 2nd order grating, h_1 is real and positive. The effect of the first order diffraction (at 90 degrees) is to introduce an additional loss term of $2h_1$ to the internal loss and an imaginary coupling term of $-h_1$ to the conventional coupling coefficient h_2 or (κ). Therefore the effect is not only to make the laser lossy but also to introduce a lossy coupling term which may affect the longitudinal modes significantly.

For a complex index 2nd order grating, the imaginary part of h_1 adds a term $Im(h_1)$ to the frequency deviation $\frac{\Delta\omega}{v_g}$. It also adds a term $Im(h_1)$ to the real part of κ. Therefore a loss or gain coupled 2nd order grating can also enhance the real kappa and shift the emission frequency.

16.8.3 Radiating Waves

In many applications, it is desirable to calculate the radiating wave. We recall from previous subsection that the radiating field can be written as follows,:

$$\Delta E(y,z) = -\frac{\omega^2}{c_0^2}\Delta\varepsilon \int G(y,y_1)[\zeta_{+1}(y_1)L(z) + \zeta_{-1}(y_1)R(z)]\phi(y_1)dy_1 \tag{16.99}$$

where

$$G(y,y_1) = -\frac{1}{2jk_y}\begin{cases} exp[-jk_y(y-y_1)] & y > y_1 \\ exp[-jk_y(y_1-y)] & y < y_1 \end{cases} \tag{16.100}$$

We obtain:

$$\begin{aligned}
\Delta E(y,z) &= -\frac{\omega^2}{c_0^2}\Delta\varepsilon \int G(y,y_1)\zeta_{+1}(y_1)[L(z)+R(z)]\phi(y_1)dy_1 \\
&= -\frac{\omega^2}{c_0^2}\Delta\varepsilon \int_{y>y_1} G(y,y_1)\zeta_{+1}(y_1)[L(z)+R(z)]\phi(y_1)dy_1 \\
&- \frac{\omega^2}{c_0^2}\Delta\varepsilon \int_{y<y_1} G(y,y_1)\zeta_{+1}(y_1)[L(z)+R(z)]\phi(y_1)dy_1 \\
&= \frac{\omega^2}{c_0^2}\Delta\varepsilon\frac{1}{2jk_y} \int_{y>y_1} exp[-jk_y(y-y_1)]\zeta_{+1}(y_1)[L(z)+R(z)]\phi(y_1)dy_1 \\
&+ \frac{\omega^2}{c_0^2}\Delta\varepsilon\frac{1}{2jk_y} \int_{y<y_1} exp[-jk_y(y_1-y)]\zeta_{+1}(y_1)[L(z)+R(z)]\phi(y_1)dy_1
\end{aligned} \tag{16.101}$$

We assume that the 90 degree diffraction downwards gets absorbed and we only consider the upward waves:

$$\Delta E(y,z) = \frac{\omega^2}{c_0^2}\Delta\varepsilon\frac{1}{2jk_y} exp[-jk_y(y-y_0)]\zeta_{+1}(y_0)[L(z)+R(z)]\phi(y_0)d_g \tag{16.102}$$

where we assume a thin grating layer of d_g thickness located at y_0.

If the grating is on the surface (case A) of the device, $k_y = \frac{\omega}{c_0}$ and we have:

$$\Delta E(y,z) = \frac{\pi d_g}{j\lambda}\Delta\varepsilon\zeta_{+1}(y_0)\phi(y_0)[L(z)+R(z)]exp[-jk_y(y-y_0)] \tag{16.103}$$

where Γ_g is the confinement factor for the grating layer. If the grating is buried within the device (case B), $k_y = \frac{n\omega}{c_0}$ and we have:

$$\Delta E(y,z) = \frac{\pi d_g}{jn\lambda}\Delta\varepsilon\zeta_{+1}(y_0)\phi(y_0)[L(z)+R(z)]exp[-jk_y(y-y_0)] \tag{16.104}$$

To evaluate the emitted power of the radiating field, we consider the expression for the edge emitting power first:

$$E(y,z) = [R(z)exp(-j\beta_0 z) + L(z)exp(j\beta_0 z)]\phi(y) \tag{16.105}$$

The bulk photon density is written as

$$s(y,z) = \epsilon_0 n^2|E|^2 = \epsilon_0 n^2|R(z)|^2 + |L(z)|^2]|\phi(y)|^2 \tag{16.106}$$

where we have neglected the interference term. The linear photon density is as follows:

$$s_l(z) = \epsilon_0 n^2 \int |E|^2 dxdy = \epsilon_0 n^2|R(z)|^2 + |L(z)|^2]| \tag{16.107}$$

We assume that

$$\int \phi(y)dydx = 1 \tag{16.108}$$

In our simulator, many of power quantities are expressed in terms of the total photon number of the mode, S, which can be directly calculated from the Green's function.

$$S = \int s_l(z)dz = \epsilon_0 n^2 \int |R(z)|^2 + |L(z)|^2]|dz \tag{16.109}$$

from which we can calculate the amplitude of R and L given the photon number in the cavity.

The emitted power from the surface for the case A is as follows:

$$\begin{aligned} P_{top} &= \hbar\omega c_0\epsilon_0 \int |\Delta E(y,z)|^2 dxdz \\ &= \hbar\omega c_0\epsilon_0 \frac{\pi^2 d_g^2}{\lambda^2}(\Delta\varepsilon)^2\zeta_1^2\phi(y_0)^2 d_{wid}\int [L(z)+R(z)]^2 dz \\ &= \hbar\omega c_0\epsilon_0 \frac{\pi^2 d_g}{\lambda^2}(\Delta\varepsilon)^2\zeta_1^2\Gamma_g \int |L(z)+R(z)|^2 dz \\ &= \hbar\omega c_0\epsilon_0 \frac{\pi^2 d_g}{\lambda^2}(\Delta\varepsilon)^2\zeta_1^2\Gamma_g S\left\{\frac{\int |L(z)+R(z)|^2 dz}{\epsilon_0 n^2 \int |R(z)|^2+|L(z)|^2]|dz}\right\} \end{aligned} \tag{16.110}$$

or

$$P_{top} = \hbar\omega c_0 \frac{\pi^2 d_g}{n^2\lambda^2}(\Delta\varepsilon\zeta_1)^2\Gamma_g S\left\{\frac{\int |L(z)+R(z)|^2 dz}{\int |R(z)|^2+|L(z)|^2]|dz}\right\} \tag{16.111}$$

The upwards power within the device for the case B is as follows:

$$\begin{aligned} P_{up} &= \hbar\omega v_g\epsilon_0 n^2 \int |\Delta E(y,z)|^2 dxdz \\ &= \hbar\omega v_g\epsilon_0 n^2 \frac{\pi^2 d_g^2}{n^2\lambda^2}(\Delta\varepsilon)^2\zeta_1^2\phi(y_0)^2 d_{wid}\int [L(z)+R(z)]^2 dz \\ &= \hbar\omega v_g\epsilon_0 \frac{\pi^2 d_g}{\lambda^2}(\Delta\varepsilon)^2\zeta_1^2\Gamma_g \int |L(z)+R(z)|^2 dz \\ &= \hbar\omega v_g\epsilon_0 \frac{\pi^2 d_g}{\lambda^2}(\Delta\varepsilon)^2\zeta_1^2\Gamma_g S\left\{\frac{\int |L(z)+R(z)|^2 dz}{\epsilon_0 n^2 \int |R(z)|^2+|L(z)|^2]|dz}\right\} \\ &= \hbar\omega v_g \frac{\pi^2 d_g}{n^2\lambda^2}(\Delta\varepsilon\zeta_1)^2\Gamma_g S\left\{\frac{\int |L(z)+R(z)|^2 dz}{\int |R(z)|^2+|L(z)|^2]|dz}\right\} \end{aligned} \tag{16.112}$$

The surface emitting power (case B) is as follows:

$$P_{top} = (1-r_{top})\hbar\omega v_g \frac{\pi^2 d_g}{n^2\lambda^2}(\Delta\varepsilon\zeta_1)^2\Gamma_g S\left\{\frac{\int |L(z)+R(z)|^2 dz}{\int |R(z)|^2+|L(z)|^2]|dz}\right\} \tag{16.113}$$

where r_{top} is the power reflectivity of the top facet.

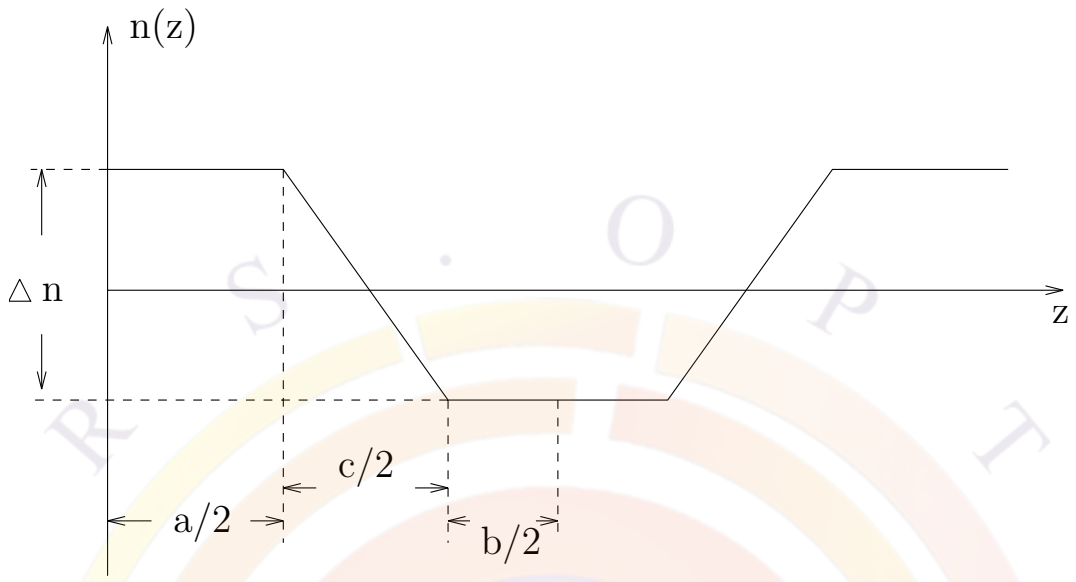


Figure 16.1: Schematics of a common grating structure.

We define a unitless coefficient (surface power emisison coefficient) to measure the surface emission intensity:

$$C_{surf} = \frac{\pi^2 d_g}{n^2 \lambda}(\Delta\varepsilon\zeta_1)^2\Gamma_g \tag{16.114}$$

so that the emisison power for case A is given by

$$P_{top} = \frac{\hbar\omega c_0}{\lambda} C_{surf} S \left\{ \frac{\int |L(z)+R(z)|^2 dz}{\int |R(z)|^2 + |L(z)|^2]|dz} \right\} \tag{16.115}$$

and for case B:

$$P_{top} = (1-r_{top})\frac{\hbar\omega v_g}{\lambda} C_{surf} S \left\{ \frac{\int |L(z)+R(z)|^2 dz}{\int |R(z)|^2 + |L(z)|^2]|dz} \right\} \tag{16.116}$$

16.8.4 A common grating structure

We are going to solve for a commonly used grating structure with the following index distribution (see Fig. 16.1).

$$\begin{aligned}
\Delta n(z) &= \Delta n/2; \qquad 0 < z < a/2 \\
\Delta n(z) &= \Delta n/2 - \frac{2\Delta n}{c}(z - a/2); \qquad a/2 < z < a/2 + c/2 \\
\Delta n(z) &= -\Delta n/2; \qquad a/2 + c/2 < z < a/2 + c/2 + b/2
\end{aligned} \tag{16.117}$$

$$\Delta n(z) = \Delta n[2\zeta_1 cos(\beta_0 z) + 2\zeta_2 cos(2\beta_0 z)] \tag{16.118}$$

where $\beta_0 = \frac{2\pi}{a+c+b}$

Integrate the above equation by $\int_0^{a/2+c/2+b/2} cos(\beta_0 z)dz$ to obtain:

$$\begin{aligned}
&\int_0^{a/2+c/2+b/2} \Delta n(z) cos(\beta_0 z)dz \\
= &\int_0^{a/2} \Delta n/2 cos(\beta_0 z)dz \\
+ &\int_{a/2}^{a/2+c/2} \left[\Delta n/2 - \frac{2\Delta n}{c}(z - a/2)\right] cos(\beta_0 z)dz \\
+ &\int_{a/2+c/2}^{a/2+c/2+b/2} (-\Delta n/2) cos(\beta_0 z)dz \\
= &\Delta n/2 \frac{1}{\beta_0} sin(\beta_0 z)\Big|_0^{a/2} \\
+ &\left[\Delta n/2 + \frac{\Delta n}{c}a\right] \frac{1}{\beta_0} sin(\beta_0 z)\Big|_{a/2}^{a/2+c/2} \\
- &\frac{2\Delta n}{c}\frac{1}{\beta_0^2}[(\beta_0 z)sin(\beta_0 z) + cos(\beta_0 z)]\Big|_{a/2}^{a/2+c/2} \\
- &\Delta n/2 \frac{1}{\beta_0} sin(\beta_0 z)\Big|_{a/2+c/2}^{a/2+c/2+b/2}
\end{aligned} \tag{16.119}$$

Our result is as follows:

$$\begin{aligned}
&\int_0^{a/2+c/2+b/2} \Delta n(z) cos(\beta_0 z)dz \\
= &\Delta n/2 \frac{1}{\beta_0} sin(\beta_0 a/2) \\
+ &\left[\Delta n/2 + \frac{\Delta n}{c}a\right] \frac{1}{\beta_0}[sin(\beta_0(a/2 + c/2)) - sin(\beta_0 a/2)] \\
- &\frac{2\Delta n}{c}\frac{1}{\beta_0^2}[\beta_0(a/2 + c/2)sin(\beta_0(a/2 + c/2)) - (\beta_0 a/2)sin(\beta_0 a/2) \\
+ &cos(\beta_0(a/2 + c/2)) - cos(\beta_0 a/2)] \\
- &\Delta n/2 \frac{1}{\beta_0}[sin(\beta_0(a/2 + c/2 + b/2)) - sin(\beta_0(a/2 + c/2))]
\end{aligned} \tag{16.120}$$

We derive the expression for ζ_1 here.

$$\int \Delta n(z) cos(\beta_0 z) dz = \Delta n 2\zeta_1 \int cos^2(\beta_0 z) dz \tag{16.121}$$

$$\int_0^{a/2+c/2+b/2} cos^2(\beta_0 z) dz = (a/2 + c/2 + b/2)/2 \tag{16.122}$$

$$\int \Delta n(z) cos(\beta_0 z) dz = \Delta n \zeta_1 (a/2 + c/2 + b/2) \tag{16.123}$$

$$\zeta_1 = \frac{1}{\Delta n(a/2 + c/2 + b/2)} \int \Delta n(z) cos(\beta_0 z) dz \tag{16.124}$$

Similarly,

$$\begin{aligned}
& \int_0^{a/2+c/2+b/2} \Delta n(z) cos(2\beta_0 z) dz \\
= \; & \Delta n/2 \frac{1}{2\beta_0} sin(2\beta_0 a/2) \\
+ \; & \left[\Delta n/2 + \frac{\Delta n}{c} a \right] \frac{1}{2\beta_0} [sin(2\beta_0(a/2 + c/2)) - sin(2\beta_0 a/2)] \\
- \; & \frac{2\Delta n}{c} \frac{1}{2\beta_0^2} [2\beta_0(a/2 + c/2) sin(2\beta_0(a/2 + c/2)) - (2\beta_0 a/2) sin(2\beta_0 a/2) \\
+ \; & cos(2\beta_0(a/2 + c/2)) - cos(2\beta_0 a/2)] \\
- \; & \Delta n/2 \frac{1}{2\beta_0} [sin(2\beta_0(a/2 + c/2 + b/2)) - sin(2\beta_0(a/2 + c/2))]
\end{aligned} \tag{16.125}$$

$$\zeta_2 = \frac{1}{\Delta n(a/2 + c/2 + b/2)} \int \Delta n(z) cos(2\beta_0 z) dz \tag{16.126}$$

In the special case of $c = 0$:

$$\begin{aligned}
& \int_0^{a/2+b/2} \Delta n(z) cos(\beta_0 z) dz \\
= \; & \frac{\Delta n}{\beta_0} sin(\pi a/(a + b))
\end{aligned} \tag{16.127}$$

$$\zeta_1 = \frac{1}{(a/2 + c/2 + b/2)} \frac{a + b + c}{2\pi} sin(\pi a/(a + b)) \tag{16.128}$$

$$\zeta_1 = \frac{1}{\pi} sin(\pi a/(a + b)) \tag{16.129}$$

$$\zeta_2 = \frac{1}{\Delta n(a/2+c/2+b/2)} \int \Delta n(z) cos(2\beta_0 z) dz \tag{16.130}$$

$$\begin{aligned} & \int_0^{a/2+c/2+b/2} \Delta n(z) cos(2\beta_0 z) dz \\ = \; & \Delta n \frac{1}{2\beta_0} sin(2\beta_0 a/2) \end{aligned} \tag{16.131}$$

$$\zeta_2 = \frac{1}{2\pi} sin(2\pi a/(a+b)) \tag{16.132}$$

In the case of thin grating layer of d_g with a confining factor for the grating layer of Γ_g, $exp(jk_y y) = 1$ and we have the following results:

$$\begin{aligned} h_1 &= \left(\frac{2\pi}{\lambda}\right)^2 (\Delta n)^2 \zeta_1^2 \Gamma_g d_g \\ &= \left(\frac{2\pi}{\lambda}\right)^2 (\Delta n)^2 \frac{1}{\pi^2} sin^2(\pi a/(a+b)) \Gamma_g d_g \\ &= \left(\frac{2}{\lambda}\right)^2 (\Delta n)^2 sin^2(\pi a/(a+b)) \Gamma_g d_g \end{aligned} \tag{16.133}$$

$$h_2 = \frac{2\pi}{\lambda} \Delta n \int \zeta_2(y) \phi(y)^2 dy = \frac{1}{\lambda} \Delta n sin(2\pi a/(a+b)) \Gamma_g \tag{16.134}$$

$$h_1/h_2 = \lambda \left(\frac{2}{\lambda}\right)^2 \Delta n sin^2(\pi a/(a+b)) d_g / sin(2\pi a/(a+b)) \tag{16.135}$$

$$\begin{aligned} h_1/h_2 = \; & \frac{2}{\lambda} \Delta n sin^2(\pi a/(a+b)) d_g / [sin(\pi a/(a+b)) cos(\pi a/(a+b))] \\ = \; & \frac{2\Delta n d_g}{\lambda} tan(\pi a/(a+b)) \end{aligned} \tag{16.136}$$

The ratio above is identical to that derived in Ref. [81].

16.8.5 Case of thicker grating layer

In case of thicker grating layer (comparable or greater than wavelength) we need to evaluate the coupling coefficient and surfurce emitting power more accurately. We

need to modify Eqs. 16.114 and 16.133. In both of these equations, the factor $\Gamma_g d_g$ should be replaced by the following integral:

$$\Gamma_g d_g = |\int \phi(x,y) exp[-jk_y(y-y_c)]dxdy|^2/d_{wid} \tag{16.137}$$

where d_{wid} is the width of the grating layer. The integral is performed over the x-y plane on the grating region. The y_c is the y-position within the grating chosen such that the 2nd term in Eq. 16.90 is zero. It is usually reasonable to choose y_c as the center of the grating. The above integral can be performed numerically.

16.9 3D Optical Amplifier Model

PICS3D is ideal for the study of semiconductor optical amplifiers. The simulation approach is similar to that for lasers. The lack of optical feedback results in a simplification of our numerical models. We can skip the longitudinal model search and only deal with the spatial hole burning effects.

To use PICS3D for optical amplifier, we assume that the input light is from the right facet with a single wavelength. We ramp up the bias current as in a laser simulation. As the optical gain increases we expect the output power from the left facet to increase. The statement to activate the amplifier model is **3d_amplifier_model**.

16.10 Beam Propagation Method

All the models described so far in this chapter are based on the separation of variables performed in Sec. 16.2. That is, the lateral mode profile $\phi_0(x,y)$ changes very little in the z direction.

In cases where this is assumption is not respected (e.g. tapered lasers), the beam propagation method (BPM) may be used to obtain the forward and backward field profiles and evaluate the Wronskian and round-trip gain.

As of version 2009, PICS3D uses a new FEM-BPM method: it will be detailed in subsequent revisions of the manual.

16.11 Device structures

It is now appropriate to discuss some of the laser structures modeled by PICS3D and how they correspond to the above equations:

- Fabry-Perot lasers are the limiting case where κ=0.

- Conventional **DFB** lasers are assumed to have AR coating on both facets and a real κ with no z-dependence.
- **Phase Shifted DFB** lasers are assumed to have real κ and zero $\Omega(z)$ but with a phase shifted section (or a section without gratings) somewhere along the waveguide.
- **Chirped Grating DFB** lasers can be represented by a linear change in z of $\Delta\beta_{ch}$. This can be realized with changing the grating period linearly.
- **Longitudinal Graded Index** lasers can be represented by a linear change in $\Delta\tilde{k}$.
- **DBR** lasers can be considered as a special case of DFB lasers with no grating in the active section. The passive mirror section has a regular grating with a low material gain. In such a case, more than one material segment may need to be defined.
- **Gain/Loss Coupled DFB** lasers can be considered as having complex κ.
- **Tapered structures** with longitudinal variation of the cross section and optical confinement of the laser are modeled with the beam propagation method (BPM).

PICS3D provides every possibility of defining various variables within the framework of the coupled wave equation. It is interesting to note that a complex κ (*i.e.*, gain coupled structure) has a similar (but not identical) effect on κ as changing the phase $\Omega_g(z)$ (chirp grating). This is consistent with the well known fact that gain coupling, chirp grating and phase shifting all give similar results in the emission characteristics of the laser and help move the main emission peak into the spectrum bandgap.

16.12 Slope Efficiency Issues: 3D vs. 2D Simulation

16.12.1 Introduction

For laser diodes with strong longitudinal variation such as DBF and DBR lasers, 3D simulation is necessary. For Fabry-Perot (FP) lasers, it would be interesting to compare 2D and 3D model of power emission characteristics. To be more specific, we are interested in comparing the slope efficiency of 2D (as in LASTIP simulation) and 3D (as in PICS3D). The purpose of this section is to provide insight and guidance to users who use both LASTIP and PICS3D would like to know exactly what kind of difference to expect when comparing the results of 2D and 3D simulations.

In the following discussion, we assume the laser has been well driven into lasing with stimulated emission dominating.

Let us recall power emission models in LASTIP (2D model) for a Fabry-Perot laser. We use the mirror loss α_m:

$$\alpha_m = 1/(2L)ln[1/(R_1R_2)] \tag{16.138}$$

where L is the cavity length; R_1 and R_2 are the facet power reflectivities. Following standard practice, our 2D simulator LASTIP computes the power emitted (P_e) from the laser diode as the photon number in the cavity (S) times the mirror loss:

$$P_e/(\hbar\omega) = v_g\alpha_m S \tag{16.139}$$

where v_g is the group velocity of light within the laser.

In a 2D simulation, we usually assume the modal optical gain is uniform in the z-dimension. Thus the injected current/power converts into stimulated recombination:

$$P_{inj}(z)/(\hbar\omega) = v_g g_m(z) P_{tot}(z) \tag{16.140}$$

The lasing condition requires that:

$$g_{m0} = \alpha_i + \alpha_m \tag{16.141}$$

where α_i is the internal material loss.

The efficiency can thus be written as,

$$\eta_{2D} = P_e/P_{inj} = 1/(1 + \alpha_i/\alpha_m) = 1/(1 + F_r) \tag{16.142}$$

where, for the convenience of discussions later, we defined a fraction F_r:

$$F_{r2D} = \alpha_i/\alpha_m = \frac{2L\alpha_i}{ln[1/(R_1R_2)]} \tag{16.143}$$

16.12.2 3D simulation: general consideration

Now let use examine the situation in 3D simulation with consideration of spatial hole burning. Here we define spatial hole burning (SHB) as gain (and index variation) in the z-direction caused by inhomogeneous stimulated recombination In the case of FP laser with uniform injection (as is the case for default setting in PICS3D) along the cavity of an edge laser, any variation in optical gain must be accompanied by variation in wave intensity when the laser is well above threshold [see Eq. (16.140)].

In the following derivation, we neglect the factor of $\frac{1}{\hbar\omega}$ so that the power should be regarded as having units of $\hbar\omega$.

We separate the linear power density in an arbitrary laser into right-going and left-going waves:

$$P_{tot}(z) = P_r(z) + P_l(z) \tag{16.144}$$

By definition of gain and loss, we have the following relation if we neglect any coupling between the left and right-going waves.

$$\begin{aligned} dP_r/dz &= (g_m(z) - \alpha_i)P_r. \\ dP_l/dz &= -(g_m(z) - \alpha_i)P_l. \end{aligned} \tag{16.145}$$

The injected power through stimulated recombination for the right going power is as follows,

$$\begin{aligned} \int v_g P_r(z) g_m(z) dz &= \int v_g \alpha_i P_r(z) dz + \int v_g P_r(z)(g_m(z) - \alpha_i) dz \\ &= v_g \int \alpha_i P_r(z) dz + v_g \int d[P_r(z)] \\ &= v_g \int \alpha_i P_r(z) dz + v_g[P_r(z2) - P_r(z1)] \end{aligned} \tag{16.146}$$

Please note that we have merely used the definition in Eq. (16.145) and no assumptions of uniformity in any quantities are made.

Similarly, integrated injected power for the left going wave is as follows:

$$v_g \int P_l \alpha_i dz + v_g(P_l(z1) - P_l(z2)) \tag{16.147}$$

By simple algebraic manipulations, we can rewrite the injection power as follows:

$$P_{inj} = v_g \int P_{tot}(z)\alpha_i dz + v_g P_l(z_1)(1 - R_1) + v_g P_r(z_2)(1 - R_2) \tag{16.148}$$

whose last two terms we identify as the power emission from both laser facets:

$$\begin{aligned} P_e &= v_g P_l(z_1)(1 - R_1) + v_g P_r(z_2)(1 - R_2) \\ &= v_g P_{tot}(z_1)(1 - R_1)/(1 + R_1) + v_g P_{tot}(z_2)(1 - R_2)/(1 + R_2) \end{aligned} \tag{16.149}$$

Therefore we obtain the following expression of efficiency for a general laser diode:

$$\eta_{3D} = 1/(1 + F_{r3D}) \tag{16.150}$$

where, using definition of F_r,

$$F_{r3D} = \frac{\int \alpha_i P_{tot}(z) dz}{P_{tot}(z_1)(1 - R_1)/(1 + R_1) + P_{tot}(z_2)(1 - R_2)/(1 + R_2)} \tag{16.151}$$

Please note that the above derivation neglects any grating/coupling effects. However, it can be shown that if the coupling coefficient is real, the formula in Eq. (16.151)applies to any type of lasers including FP, DFB, DBR and VCSEL with variations in all quantities.

This expression leads to some useful conclusions. Suppose we fix the photon density at the facets $P_{tot}(z_1)$ and $P_{tot}(z_2)$ and allow the intensity to vary inside the cavity. If the intensity inside is low (such as in FP laser), the integral in the numerator will be small and thus lead to a higher efficiency.

On the other hand, if the power is high in the mid-section and small near the facets, such as in 1/4 wavelength shifted DFB laser and VCSEL , the laser will be a less efficient laser.

16.12.3 Fabry-Perot laser with uniform gain

We shall show that for FP laser with uniform gain, Eq. (16.151) reduces to our standard 2D expression in Eq. (16.143).

For simplicity, we assume that $R_1 = R_2$. If the optical gain is uniform in the longitudinal direction, we can write the wave intensity as follows.

$$P_{tot}(z) = P_0 \{exp[gz] + exp[g(L-z)]\} \tag{16.152}$$

where g is the net gain: $g = g_m - \alpha_i$. The integral in the numerator of Eq. (16.151) becomes:

$$\int \alpha_i P_{tot}(z)dz = 2\alpha_i P_0[exp(gL) - 1]/g \tag{16.153}$$

Using lasing condition $gL = ln(1/R)$, we obtain

$$\begin{aligned} \int \alpha_i P_{tot}(z)dz &= \frac{2\alpha_i P_0[(1/R) - 1]}{(1/L)ln(1/R)} \\ &= 2\alpha_i P_0[(1/R) - 1]/\alpha_m \end{aligned} \tag{16.154}$$

We notice that in the denominator of Eq. (16.151),

$$P_{tot}(z_1) = P_0[1 + exp(gL)] = P_0[1 + (1/R)] \tag{16.155}$$

Therefore, we obtain the following results for FP laser under uniform gain:

$$\begin{aligned} F_{r3D} &= \frac{\int \alpha_i P_{tot}(z)dz}{2P_{tot}(z_1)(1-R)/(1+R)} \\ &= \frac{2\alpha_i P_0[(1/R) - 1]/\alpha_m}{2P_0[1 + (1/R)](1-R)/(1+R)} \\ &= \alpha_i/\alpha_m \end{aligned} \tag{16.156}$$

which is exactly the model we use in 2D model.

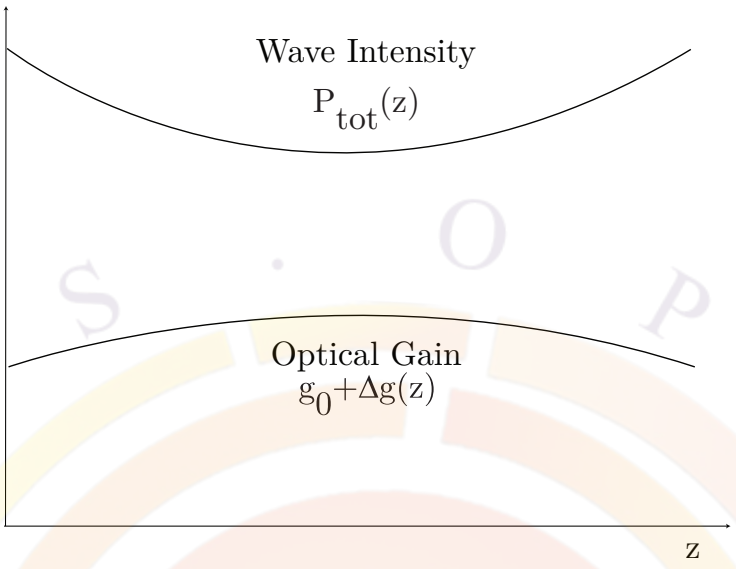


Figure 16.2: Schematics of variation of power and gain within a Fabry-Perot laser.

16.12.4 Fabry-Perot laser with non-uniform gain

In this subsection, we allow the optical gain in FP laser to vary. In general, there is no analytical expression if optical gain is allowed to vary. But we can assume a reasonable form for the gain variation in order to study its effect on laser emission characteristics. We assume the following parabolic variation of the net optical gain (see Fig. 16.2):

$$g(z) = g_0 + \Delta g - \Delta g \left(\frac{2}{L}\right)\left(z - \frac{L}{2}\right)^2 \tag{16.157}$$

The optical power of the right-going wave can be expressed as follows from the definition of the net optical gain:

$$ln\left(\frac{P_r(z)}{P_r(0)}\right) = (g_0 + \Delta g)z - \frac{\Delta g}{3}\left(\frac{2}{L}\right)^2\left[\left(z - \frac{L}{2}\right)^3 + \left(\frac{L}{2}\right)^3\right] \tag{16.158}$$

By symmetry consideration, we can write down similar expression for the optical power of the left-going wave.

The lasing condition becomes:

$$\int_0^L g(z)dz = ln(1/R) \tag{16.159}$$

or

$$g_0 + 2\Delta g/3 = (1/L)ln(1/R) \tag{16.160}$$

which is to be compared with the case of uniform gain: $g_0 = (1/L)ln(1/R)$.

Since the stimulated recombination is dominant above threshold, we require that the product of gain and total optical power is uniform because of uniform injection assumption. This is in general not possible for parabolic gain variation. However, we believe it will be reasonable to require that the product of gain-power be equal at facet and at the middle point of the cavity as follows:

$$g(0)[P_r(0) + P_l(0)] = g(L/2)[P_r(L/2) + P_l(L/2)] \tag{16.161}$$

Noting that

$$P_r(0) = P_0[1 + exp[(g_0 L + 2\Delta g L/3)] \tag{16.162}$$

and

$$P_r(L/2) = P_l(L/2) = P_0 exp[(g_0 + 2\Delta g/3)(L/2)] \tag{16.163}$$

Eq. (16.161) becomes

$$g_0[1 + exp(g_0 L + 2\Delta g L/3)] = (g_0 + \Delta g)2exp[(g_0 + 2\Delta g/3)(L/2)] \tag{16.164}$$

Using the lasing condition in Eq. (16.160), the above becomes:

$$g_0(1 + 1/R) = 2(g_0 + \Delta g)\sqrt{1/R} \tag{16.165}$$

Combing the lasing condition in Eq. (16.160) with with Eq. (16.165), we obtain the following solution of spatial hole burning under the assumption of parabolic gain variation:

$$\Delta g = \left(\frac{1}{L} ln \frac{1}{R}\right) \times \left[\frac{2}{3} + \frac{2\sqrt{1/R}}{1 + (1/R) - 2\sqrt{1/R}}\right]^{-1} \tag{16.166}$$

$$g_0 = \frac{1}{L} ln \frac{1}{R} - \frac{2}{3}\Delta g \tag{16.167}$$

Having successfully solved the power distribution within the FP laser with spatial hole burning, we may now proceed to compute the quantum efficiency using Eq (16.151).

$$\begin{aligned} F_{r3D} &= \frac{\alpha_i \int P_{tot}(z)dz}{2P_{tot}(z_1)(1-R)/(1+R)} \\ &= \frac{\alpha_i R}{1-R}\int_0^L exp\left\{(g_0 + \Delta g)z - \frac{\Delta g}{3}\left(\frac{2}{L}\right)^2\left[\left(z - \frac{L}{2}\right)^3 + \left(\frac{L}{2}\right)^3\right]\right\} dz \end{aligned} \tag{16.168}$$

We find it convenient to write the above integral in the following form:

$$F_{r3D} = \frac{\alpha_i R}{1-R}\int_0^L exp[(g_0 + 2\Delta g/3)z + h(z)]dz \tag{16.169}$$

$$h(z) = \frac{\Delta g z}{3} - \frac{\Delta g}{3}\left(\frac{2}{L}\right)^2\left[\left(z - \frac{L}{2}\right)^3 + \left(\frac{L}{2}\right)^3\right] \tag{16.170}$$

Since $h(z)$ is a small quantity (linear in Δg) equal to zero at facet and at the midpoint of the cavity, we may use the following Taylor expansion:

$$\begin{aligned} F_{r3D} &= \frac{\alpha_i R}{1-R}\int_0^L exp[(g_0+2\Delta g/3)z]dz \\ &\quad + \frac{\alpha_i R}{1-R}\int_0^L exp[(g_0+2\Delta g/3)z]h(z)dz \qquad (16.171) \\ &= term1 + term2 \qquad (16.172) \end{aligned}$$

We find the first term above reduces to that of the usual mirror loss:

$$\begin{aligned} term1 &= \frac{\alpha_i R}{1-R}\int_0^L exp[(g_0+2\Delta g/3)z]dz \qquad (16.173) \\ &= \frac{\alpha_i R}{(1-R)(g_0+2\Delta g/3)}\{exp[(g_0+2\Delta g/3)L]-1\} \qquad (16.174) \end{aligned}$$

Using the lasing condition again,

$$\begin{aligned} term1 &= \frac{\alpha_i R}{(1-R)(1/L)ln(1/R)}[(1/R)-1] \\ &= \frac{\alpha_i}{(1/L)ln(1/R)} \qquad (16.175) \end{aligned}$$

The second term $term2$ may be regarded as a correction term to the standard mirror loss model for the computation of quantum efficiency. It may be evaluated analytically. Since the final expression is rather lengthy, we shall only plot the numerical results of the correction term for a typical case of $R = 0.32$ and $\alpha_i = 1000$ 1/m in Fig. 16.3. Please note that the correction term, $term2$, is proportional to the internal loss. The correction term becomes larger as the cavity length increases, as we may expect from increased SHB. However, the magnitude of the correction term to the conventional mirror loss in FP laser is typically around 0.1%.

16.12.5 Conclusions

In conclusion, we have derived a convenient formula for computation of slope efficiency of lasing characteristics from a given power distribution. It is convenient to be used to analyze the emission efficiency of an arbitrary laser under condition of spatial hole burning (SHB). For a Fabry-Perot laser, use of mirror loss (1/L)ln(1/R) is equivalent to a model of power distribution with uniform gain. The effect of SHB on the efficiency of the standard mirror is in general rather small (about 0.1%) for a Fabry-Perot laser. Thus, as far as FP lasers are concerned, 2D simulation (as in LASTIP) and 3D simulation (as in PICS3D) should yield the same result for practical purposes.

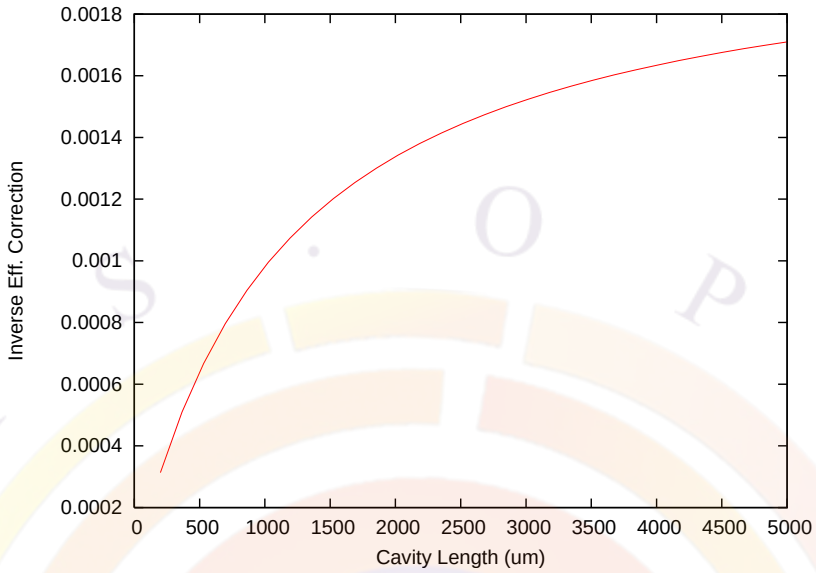


Figure 16.3: Correction to the inverse efficiency of a Fabry Perot laser due to spatial hole burning. The facet power reflectivity is taken to be 0.32. The internal loss is set to be 1000 1/m

In the case of a phase shifted DFB the SHB influence can be very significant. In that case both the gain distribution and index distribution will greatly affect the mirror loss. Thus it is important to perform longitudinal modelling.

The author of this section (Simon Li) wishes to acknowledge helpful discussions on the topic of spatial hole burning with Richard Schatz and Sebastian Mogg, Royal Institute of Technology, Stockholm."

Chapter 17

PICS3D-SPECIFIC MODELS II: VCSEL Theory

17.1 Introduction

In the previous chapter, we introduced the longitudinal model used by PICS3D for edge-emitting devices. For VCSELs, much of the theory remains the same but there are a few important changes that need to be discussed.

The first major change is that the equations must be solved in the cylindrical coordinate system shown in Fig. 17.1. For a more detailed explanations on how the drift-diffusion and Poisson equations are altered by the change in coordinate system, the user is referred to Sec. 6.2.

17.2 Lateral Modes: Fiber-Like Index Model

The exact solution of the optical mode in VCSEL involves the vector wave solution of a microcavity with complex boundaries. To see the complexity involved in a vector wave solution of even a simple cylindrical boundary, the reader is referred to Ref. [110]. Consequently, we have decided to start our VCSEL model from a simple scalar wave solution and build up our capabilities in later versions.

We consider the solution of a scalar wave propagating in the axial direction of a VCSEL structure (see the first and the second structure in Fig. 17.2. We assume that the wave distribution in the VCSEL is such that the top cylinder determines the optical confinement in the lateral direction and the lower cylinder only contributes an effective index change in the lateral cladding (outer core). Essentially, we approximate the optical confinement problem to that of an optical fiber.

We shall call this approximation a fiber-like index model in vertical structures [see

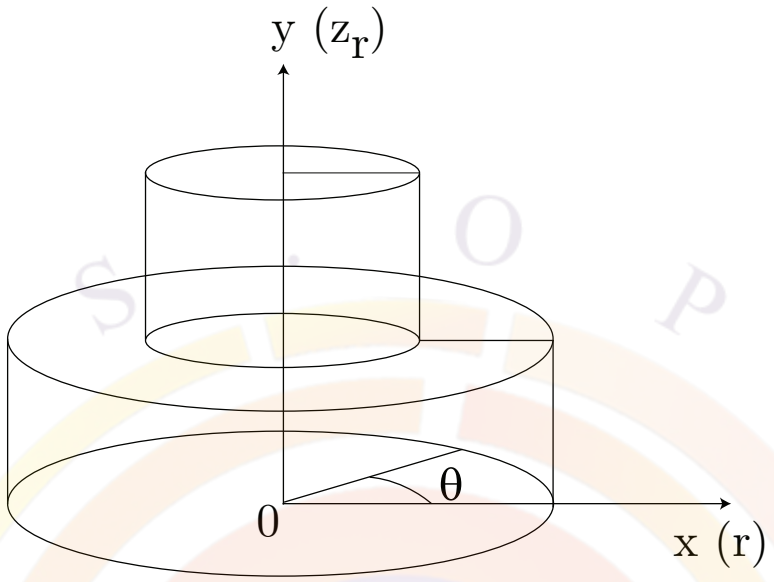


Figure 17.1: Cylindrical coordinate system used to solve VCSEL equations

the third structure in Figure 17.2]. Under this approximation, the problem is reduced to a simple one-dimensional equation with cylindrical boundaries and a closed-form solution [111].

The basic scalar wave equation is given by:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial W}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 W}{\partial \theta^2}+\frac{\partial^2 W}{\partial z_r^2}+k^2 W=0. \tag{17.1}$$

where z_r is in the axial direction and coincides with the y-axis.

It has the following simple solution with cylindrical symmetry:

$$W=W_r(r)exp(-jk_z z_r) \tag{17.2}$$

where $W_r(r)$ is solved from the Bessel equation of order m:

$$\frac{d^2 W_r}{dr^2}+\frac{1}{r}\frac{dW_r}{dr}+\left(k^2-k_z^2-\frac{m^2}{r^2}\right)W_r=0. \tag{17.3}$$

Using the well established Bessel and Hankel functions, the scalar wave equation can be solved efficiently in PICS3D. The fiber-like solution is the default lateral optical model used in PICS3D.

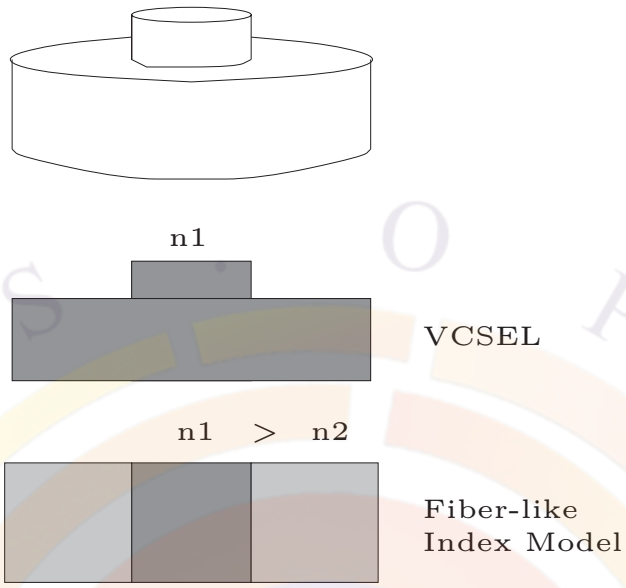


Figure 17.2: Schematic of how the fiber-like effective index approximation is used for a cylindrical VCSEL.

17.3 Effective Index Method for VCSELs

17.3.1 Introduction

In the previous section we describe a fiber-like solution for the lateral optical mode of a VCSEL; such a model has the advantage of being easy to interpret and handle. However, when the VCSEL structure becomes more complex, especially when there is oxide confinement layer, a more sophisticated lateral model becomes necessary.

The VCSEL Effective Index Method (VCSEL-EIM) approximation was first proposed by Hadley[112] and has been successfully used to calculate the optical modes of more complicated VCSEL structures. In PICS3D, we use this VCSEL EIM calculation method for the calculation of the lateral optical modes in VCSELs and it especially recommended for structures with an oxide confinement aperture.

17.3.2 Models

The scalar wave equation in VCSELs is described by the following wave equation:

$$\nabla^2 F(r,z,\phi,t) - \frac{\epsilon(r,z)}{c^2}\frac{\partial^2 F(r,z,\phi,t)}{\partial t^2} = 0, \tag{17.4}$$

where $\epsilon(r,z)$ is the relative permittivity, c is the speed of light in vacuum. The solution of above electric field can be separated into longitudinal ($E_i(z)$) and lateral

$(E(r,\phi,t))$ components in VCSELs:

$$F(r,z,\phi,t) \approx E_i(z)E(r,\phi,t)e^{-i\omega t}, \tag{17.5}$$

where i denotes a region in the transverse direction in which the effective index is calculated, and ω_0 is the angular oscillation frequency of the light. Substituting equation (17.5) into (17.4) and writing $\nabla^2 = \partial^2/\partial z^2 + \nabla_\perp^2$, we have:

$$\frac{\partial^2 E_i(z)}{\partial z^2}E(r,\phi,t) + E_i(z)\nabla_\perp^2 E(r,\phi,t) + \epsilon_i(r,z)k_0^2 E_i(z)E(r,\phi,t) + 2i\epsilon_i(r,z)k_0 E_i(z)\frac{\partial E(r,\phi,t)}{c\partial t} - \epsilon_i(r,z)E_i(z)\frac{\partial^2 E(r,\phi,t)}{c^2\partial^2 t} = 0 \tag{17.6}$$

where $k_0 = \omega_0/c$. By neglecting the second derivative of $E(r,\phi,t)$ with respect to time (slowly-varying envelope approximation) and writing the relative permittivity $\epsilon(r,z)$ as the sum of a structural component $\epsilon_i(z)$ and a nonstructural (depending on the carrier concentration, temperature etc.) component $\epsilon_r(r)$, we can get the following two equations for $E_i(z)$ (i.e. the optical standing wave pattern) and $E(r,\phi,t)$ (i.e. the lateral mode) under the effective index approximation:

$$\frac{\partial^2 E_i(z)}{\partial z^2} + k_0^2(1-\xi_i)\epsilon_i(z)E_i(z) = 0, \tag{17.7}$$

$$\frac{\partial E(r,\phi,t)}{\partial t} = \frac{ic_0}{2k_0\langle\epsilon_i\rangle}[\nabla_\perp^2 + k_0^2\Delta\epsilon_{eff}(r)]E(r,\phi,t), \tag{17.8}$$

where $\langle\epsilon_i\rangle = \int E_i^*(z)\epsilon_i(z)E_i(z)dz / \int E_i^*(z)E_i(z)dz$ and $\Delta\epsilon_{eff}(r) = \xi_i\langle\epsilon_i\rangle + \epsilon_r(r)$. c_0 is the speed of light. The real part of the eigenvalue ξ_i is related to the effective index and the imaginary part is related to the cavity loss (or gain). Using the conventional LP_{mn} notation, the transverse component of the electrical field can be written in the following way:

$$E_{mn}(r,\phi,t) \propto \Psi_{mn}(r,\phi,t)e^{-i\Delta\omega_{mn}t}, \tag{17.9}$$

where Ψ_{mn} is the slowly varying modal field distribution function, and the real part of $\Delta\omega_{mn}$ is the deviation of the LP_{mn} mode from the angular oscillation frequency ω_0. For a system with a cylindrical symmetry with respect to the electrical field, the modal field distribution function Ψ_{mn} must have the following form

$$\Psi_{mn}(r,\phi,t) \equiv \Psi_{mn}(r,t)e^{im\phi}. \tag{17.10}$$

Combining expressions of (17.8)-(17.10) and neglecting the time derivative of the slowly varying modal field distribution function $\Psi_{mn}(r,\phi,t)$, we get the following eigenvalue equation in cylindrical coordinates:

$$\frac{-c_0}{2k_0\langle\epsilon_i\rangle}\left[\frac{1}{r}\frac{\partial}{\partial r}r\frac{\partial}{\partial r} - \frac{m^2}{r^2} + k_0^2\Delta\epsilon_{eff}\right]\Psi_{mn} = \Delta\omega_{mn}\Psi_{mn}. \tag{17.11}$$

In PICS3D, the equations (17.6) and (17.11) are solved numerically to obtain lateral optical modes in VCSELs.

17.4 Longitudinal Modes in VCSEL

In both the lateral models described earlier in this chapter, we have a separation of variables. This means that like in our edge-emitting model from Chap. 16, we can solve the longitudinal field profile separately. However, there are some differences that justify the use of a different model to do so.

Recall that for a first-order grating, the period is approximately:

$$L_g = \frac{\lambda}{2n} \approx \frac{1.3}{2 \times 3.2} \approx 0.2\mu m$$

This means that a typical DFB/DBR laser cavity will have several thousand grating periods. The combined effect of this grating on the propagating wave is described by the coupled mode theory of Sec. 16.5. In this model, it is assumed that the grating is a weak perturbation with a sinusoidal shape.

However in a VCSEL (Fig. 17.3), the gratings consist of multiple layers of different materials with a large difference in refractive indices: using a sinusoidal function can be rather inaccurate. We also note that the number of gratings used in VCSEL is relatively small (≈ 100). Therefore it is more convenient to apply the 2×2 matrix method commonly used in multiple layer optics.

In the transfer matrix method, (TMM, see also Fig. 17.4), simple matrix elements represent propagation within a layer and the interface between layers: these are stacked together to build an equivalent transmittance/reflectance transfer matrix for a stack of layers. This is used to evaluate the round-trip gain and locate the position of the longitudinal modes.

The basic matrix elements are

$$\begin{bmatrix} exp[-j\beta_1(z-z_k)] & 0 \\ 0 & exp[j\beta_1(z-z_k)] \end{bmatrix} \tag{17.12}$$

for propagation inside a layer of index n_1 and

$$\begin{bmatrix} 1/t_1 & r_1/t_1 \\ r_1/t_1 & 1/t_1 \end{bmatrix} \tag{17.13}$$

for the interface between layers. The reflection and transmission coefficients are given by:

$$r_1 = \frac{n_1 - n_2}{n_1 + n_2} \tag{17.14}$$

$$t_1 = \sqrt{\frac{n_1}{n_2}} \frac{2n_2}{n_1 + n_2} \tag{17.15}$$

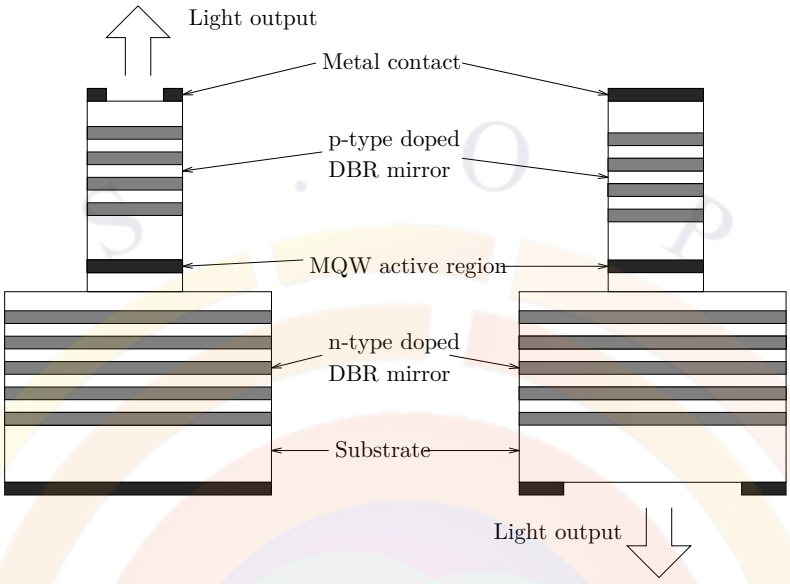


Figure 17.3: Schematics of surface-emitting lasers of front and back emission, respectively.

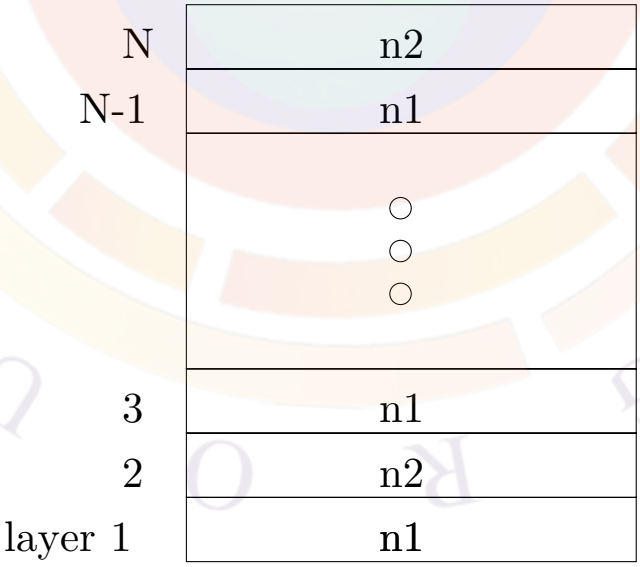


Figure 17.4: Schematic of a simple multi-layer optical model for DFB/DBR gratings.

For a periodic stack of layers, these matrices are combined using the approach suggested by Makino in [113]. We also note that this model couples well with our earlier lateral models: the lateral effective index is used in the propagation and interface matrix.

PARS OPTICS GROUP

Chapter 18

PICS3D-SPECIFIC MODELS III: Spectrum and Modulation Response

18.1 Update notes

PICS3D has been undergoing several important changes in the model in the last few years as we transition from an older quasi-3D model controlled by the **scan3d** statement to a fully coupled 3D model based on the round-trip gain (RTG).

As a result, there have been several important model changes: most of the information from the previous two chapters remains unchanged from previous versions but this chapter is still undergoing major revisions. Some sections may be incomplete or have been temporarily removed while features from the previous model are re-implemented under the new system.

Here is a brief summary of changes.

- As of version 2008, PICS3D can no longer process the **scan3d** statement: all input files must be converted to use the RTG model.
- As of version 2009, PICS3D uses a new more stable rate equation model for the longitudinal modes. AC model now consistent with that of LASTIP.
- As of version 2012, the older (pre-2008) AC model derived by Tromborg [106, 114, 115] is re-introduced as an "analytical" solution to supplement the existing fully coupled analysis. While we believe the AM/FM response from this model is inferior to our new derivation, the old model allows for noise terms (RIN, FN) and second harmonic distortion (SHD) analysis which are not supported in the full 3D model.

18.2 Multi-mode complex pole expansion

As we discussed in Sec. 16.4, a complex frequency term is used to represent the contribution of the spontaneous emission to the laser spectrum. This means that all longitudinal modes are zeroes of the Wronskian when evaluated on the complex plane.

Using this approach, we concentrate on getting the stationary multi-mode solution of a laser. We neglect any carrier dynamics and note that in Eq. (16.3), the numerator is only weakly dependent on ω.

Since now $1/W$ is non-singular for *real* optical frequencies, it can be expanded over the poles ω_i of $1/W$ on the *complex* frequency plane:

$$\frac{1}{W} \doteq \sum_i \frac{1}{\frac{dW}{d\omega}(\omega - \omega_i)} \tag{18.1}$$

The derivative $dW/d\omega$ in (18.1) is evaluated for ω equals ω_i, which is one of the zeros of the Wronskian on the complex plane.

In App. E, we show that the optical mode spectrum of a laser can be written as:

$$I_\omega = \frac{(2\pi)^{-1} R_{sp,i}}{[(\omega - \omega_{re,i})^2 + \omega_{im,i}^2]} \tag{18.2}$$

where the parameter $R_{sp,i}$ defines the spontaneous emission rate into mode i. It is given by:

$$R_{sp,i} = c \frac{\left[\int_0^l n(z) n_g |Z_0(z)|^2 dz\right] \left[\int_0^l |Z_0(z')|^2 n(z') g(z') n_{sp} dz'\right]}{\left|\int_0^l n(z) n_g Z_0^2(z) dz\right|^2} \tag{18.3}$$

Integrating over z, we get the total number of photons in the cavity $I_{p,i}$ in the i^{th} mode:

$$I_{p,i} = \frac{R_{sp,i}}{2\omega_{im,i}} \tag{18.4}$$

which agrees with results from other works in the literature (see, e.g., Ref. [106]).

In the above formulas, $Z_0(z) = Z_{0,i}(z)$ is the solution to the homogeneous scalar wave equation at $\omega = \omega_i$, v_g the group velocity, $g(z)$ the modal gain, and n_{sp} the inversion parameter. The expression (18.3) agrees completely with the results in Ref. [106].

The above derivation shows that the longitudinal modes can be represented by a set of poles on the complex frequency plane. The simulator's multi-mode calculation then becomes a task of searching and following such a set of complex poles.

18.3 Mode Spectrum

Once the steady state solution for the complex frequencies is found, we can compute the emission spectrum of the laser. We start by noting that the modal spectrum of Eq. (18.2) is actually the spectrum for the total photon number in the whole cavity.

For the emission spectrum from each facet, this must be scaled to represent the number of photons actually transmitted by that facet. To get the local photon flux inside the facet, we normalize the forward and backward waves $L(z)$ and $R(z)$ so that for each longitudinal mode, we have the normalization $\int_0^L S_i(z)dz = 1$ where $S_i(z) = |L(z,i)|^2 + |R(z,i)|^2$.

We note that $L(z)$ and $R(z)$ are known quantities used to evaluate the round-trip gain and the Wronskian during the search for longitudinal modes. The discretization is chosen so that the device is divided into sections where the optical propagation coefficient is uniform and the fields are sampled on either side of these sections.

For any given facet, we use only the forward or backward flux and correct for the the facet transmission coefficient. The photon density per unit of angular frequency is then given by:

$$S_{\omega,left} = |L(0,i)|^2 (1 - R_{left}) \frac{(2\pi)^{-1} R_{sp,i}}{[(\omega - \omega_{re,i})^2 + \omega_{im,i}^2]} \tag{18.5}$$

for the left facet and:

$$S_{\omega,right} = |R(L,i)|^2 (1 - R_{right}) \frac{(2\pi)^{-1} R_{sp,i}}{[(\omega - \omega_{re,i})^2 + \omega_{im,i}^2]} \tag{18.6}$$

for the right facet.

We note that this result is the "pure" spectrum of the laser, before any convolution with the finite resolution of a spectrum analyzer.

18.4 Time-Dependent Solution of Longitudinal Modes

We now consider a time-dependent model where the mode amplitude varies as a result of an external time-dependent bias. We assume that the external bias source has a frequency much smaller than the optical frequency ω. This assumption is reasonable since the typical optical frequency is 3×10^{14} Hz, while a typical external modulation biasing source is on the order of 10 GHz (1×10^{10} Hz).

For any theory of large signal, our requirement is that it must be consistent with our model in CW. That is, at $\Delta t \to \infty$, the equation must reduce to our CW theory.

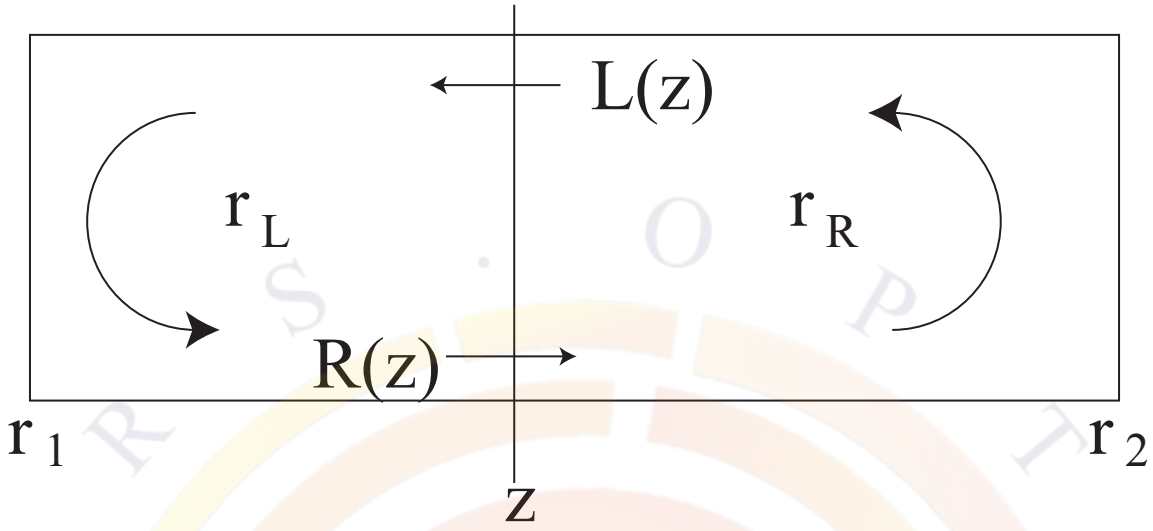


Figure 18.1: Schematic of the evaluation of round-trip gain in a laser.

Therefore, it is useful for us to expand on our Green's function results from Chap. 16 to build our time-dependent model.

18.4.1 Alternative CW Solution of Longitudinal Modes

Let us consider a laser device as shown in Fig. 18.1. We choose a reference point where both the left- and right-going waves ($L(z)$,$R(z)$) are non-zero. By examining the equivalent reflectivities to the left (r_L) and right (r_R), we have the following equations:

$$L(z) = r_R R(z) + F_L \tag{18.7}$$
$$R(z) = r_L L(z) + F_R \tag{18.8}$$

where F_L and F_R are noise sources terms.

We note that we do not define r_L and r_R at this point. They are merely effective reflection coefficients and this derivation hold true no matter which method is being used to calculate the mode profiles. We then combine these two equation and note that by definition, the round-trip gain R_g is equal to the product of r_L and r_R:

$$(1 - R_g)R(z) = r_L F_L + F_R = F_\omega \tag{18.9}$$

As previously shown, the Wronskian is position-independent so the same holds true for R_g. We thus re-use our previous result and note that under CW conditions, the laser oscillates at the complex solution ω_s:

$$1 - R_g(\omega_s) = 0 \tag{18.10}$$

18.4.2 Longitudinal Mode Transient Model

As we mentioned earlier, the time scale involved can be separated into two different types. The optical frequency causes the modes to vary with $exp(j\omega t)$ while the slow external modulation causes a much slower change $exp(j\Omega t)$ in the wave amplitude. We assume that above time scales are so different that they can be treated as decoupled.

If we apply an external modulation of $exp(j\Omega t)$, the round trip gain as calculated by the optical frequency only may not be unity since the photon density must be allowed to change with time. However the derivation in the previous section is still valid if we assume the reflectivity is not the reflectivity at optical frequency but the instantaneous reflectivity at the transient state. That is we can still use $R_g(\omega + \Omega)$ which includes the transient effect.

This approximation is consistent with our slow modulation hypothesis. At these time scales, there is enough time for photons to do several round-trips in the cavity and establish a solution that satisfies both mirror conditions simultaneously just like in our earlier derivation for the Wronskian. However, these photons may not necessarily be in equilibrium with the slower carriers.

In this case, we shall go back to our basic equation and regard the frequency as $\omega+\Omega$ where $\omega >> \Omega$. Our basic equation becomes:

$$(1 - R_g(\omega + \Omega))R(\omega + \Omega) = F_{\omega+\Omega} \tag{18.11}$$

Since $R_g(\omega) \approx 1$, we make the following simplification:

$$R_g(\omega + \Omega) = R_g(\omega)\left[1 + \frac{\partial ln(R_g)}{\partial \omega}\Omega\right] \tag{18.12}$$

$$\approx R_g(\omega) + \frac{\partial ln(R_g)}{\partial \omega}\Omega \tag{18.13}$$

We can derive $\frac{\partial ln(R_g)}{\partial \omega}$ by noting that for a FP laser:

$$R_g = r_1 r_2 exp(-j2Lk_r + g_m L) \tag{18.14}$$

Applying this, we get:

$$\frac{\partial ln(R_g)}{\partial \omega} = -j\left(\frac{2L}{v_g} + jL\frac{\partial g_m}{\partial \omega}\right) \tag{18.15}$$

where the group velocity is be written as $v_g = \left(\frac{\partial k_r}{\partial \omega}\right)^{-1}$.

We assert that since Eq. (18.15) is derived from the round-trip gain, it is applicable to any kind of laser, not just Fabry-Perot cavities. However, the real and imaginary parts on the right-hand side are effective coefficients only. For example, in a DFB laser L would be the effective cavity length of a particular longitudinal mode rather the actual length of the laser.

By substitution into our basic equation, we get:

$$\left[1 - R_g(\omega) + j\left(\frac{2L}{v_g} + jL\frac{\partial g_m}{\partial \omega}\right)\Omega\right] R(\omega + \Omega) = F_{\omega+\Omega} \tag{18.16}$$

At this point, we assume that the laser is oscillating at the peak gain wavelength so that $\frac{\partial g_m}{\partial \omega} \approx 0$. Taking the inverse Fourier transform to get the slow time dependence and noting $j\Omega = \frac{\partial}{\partial t}$, we obtain:

$$\left[1 - R_g(\omega, t) + \frac{2L}{v_g}\frac{1}{R(\omega, t)}\frac{\partial R(\omega, t)}{\partial t}\right] R(\omega, t) = F_{\omega,t} \tag{18.17}$$

We can use the photon number $S(t)$ to express $R = \sqrt{S(t)}exp(j\phi(t))$. From this, we get:

$$\frac{2L}{v_g}\frac{1}{R(t)}\frac{\partial R(t)}{\partial t} = \frac{L}{v_g}\frac{1}{S(t)}\frac{\partial S(t)}{\partial t} + j\frac{2L}{v_g}\frac{\partial \phi(t)}{\partial t} \tag{18.18}$$

Our key result is as follows:

$$\left[1 - R_g(t) + \frac{L}{v_g}\frac{1}{S(t)}\frac{\partial S(t)}{\partial t} + j\frac{2L}{v_g}\frac{\partial \phi(t)}{\partial t}\right] R(t) = F_t \tag{18.19}$$

In our simulation, we are not directly concerned with the phase variation but we are more interested in the emission frequency. The phase factor can be written as $exp[j\omega t + \phi(t)]$. We consider a short time interval $\Delta t = t - t_0$.

The phase at time t_0 is given by $j\omega_0 t_0 + \phi(t_0)$. Assuming the fast oscillation frequency does not change substantially, the new phase is:

$$j\omega_0 t + \phi(t_0) + \frac{\partial \phi(t)}{\partial t}\Delta t \tag{18.20}$$

Therefore the new frequency should be:

$$\omega(t) = \omega_0 + \frac{\partial \phi(t)}{\partial t} \tag{18.21}$$

Our basic equation in terms of $S(t)$ and $\omega(t)$ can be written in a form easy for numerical implementation:

$$\left[1 - R_g(t) + \frac{L}{v_g}\frac{1}{S(t)}\frac{\partial S(t)}{\partial t} + j\frac{2L}{v_g}(\omega(t) - \omega(t_0))\right] R(t) = F_t \tag{18.22}$$

The above equation leads to the new oscillation condition:

$$1 - R_g(t) + \frac{L}{v_g}\frac{1}{S(t)}\frac{\partial S(t)}{\partial t} + j\frac{2L}{v_g}(\omega(t) - \omega(t_0)) = 0 \tag{18.23}$$

We have implemented Eq. (18.23) in PICS3D: each longitudinal mode is governed by such an equation. These equations are coupled together in the main Newton solver alongside all the other equations (e.g. current continuity) described in Chap. 5. This makes PICS3D a full 3D model and is at the heart of what we call the "coupled RTG method".

18.4.3 Comparison with Conventional Photon Rate Equation

We are going to make sure the above is consistent with the commonly used rate equation for a FP laser. In the common textbook model, we are only interested in the propagation of photons, not fields, and changes in the emission frequency are neglected.

If we remove the terms related to these effects from Eq. (18.23) and rearrange it, we get:

$$|R_g(t)| = 1 + \frac{L}{v_g}\frac{1}{S(t)}\frac{\partial S(t)}{\partial t} \tag{18.24}$$

If we take the logarithm on each side, we get:

$$ln\,|R_g(t)| = ln(r_1 r_2) + g_m * L \tag{18.25}$$

for the left side and

$$ln\left(1 + \frac{L}{v_g}\frac{1}{S(t)}\frac{\partial S(t)}{\partial t}\right) \approx \frac{L}{v_g}\frac{1}{S(t)}\frac{\partial S(t)}{\partial t} \tag{18.26}$$

for the right side, assuming $\frac{1}{S(t)}\frac{\partial S(t)}{\partial t} \ll \frac{L}{v_g}$ for the expansion.

So for slow and/or small-signal modulation, we get:

$$ln(r_1 r_2) + g_m * L = \frac{L}{v_g}\frac{1}{S(t)}\frac{\partial S(t)}{\partial t} \tag{18.27}$$

which is equivalent to our desired expression:

$$g_m - \frac{1}{L} ln\left(\frac{1}{r_1 r_2}\right) = \frac{1}{v_g S(t)}\frac{\partial S(t)}{\partial t} \tag{18.28}$$

Note that $r1$,$r2$ are field reflection coefficients so a factor of 2 due to $r = \sqrt{R}$ may be present in some textbooks.

18.5 AC modeling in PICS3D

As of version 2009, the new rate equation model in Eq. (18.23) is used for all aspects of PICS3D. For AC modeling, it becomes one more equation to be solved and the techniques of Chap. 7 are used. This means that many effects such as carrier transit times are now included in the modulation response.

However we note that Eq. (18.23) is complex so there are actually two terms. The real part in $\frac{\partial S}{\partial t}$ gives the amplitude response and, as shown in the previous section, should reduce to the AM response from LASTIP in the case of a Fabry-Perot laser. The second term in $\omega(t) - \omega(t_0)$ produces the frequency response (FM).

18.6 Analytical model

As of version 2012, the older (pre-2008) AC model derived by Tromborg [106, 114, 115] is re-introduced as an "analytical" solution to supplement the existing fully coupled analysis. This solution is based on a transient perturbation of the steady state solution but explicitly considers derivatives of the Wronskian rather than lumping all effects into the round-trip gain as we have previous done.

As a result, Tromborg's AC model relies on longitudinal integrals of these perturbation terms rather than coupling directly to the sparse AC solver used for the drift diffusion equations. We thus consider this an "analytical" 1D model since it mainly considers the longitudinal spatial hole burning effects. We believe that our fully coupled approach is superior since carrier transport times can have a strong effect on the modulation speed of a laser and this requires treatment of transverse effects.

However, the Tromborg model does define noise and second-harmonic generation terms which are not currently available in our full 3D model so we re-introduce it here.

18.6.1 Modulation response

Without going into the detail of Tromborg's approach, we define the following general formula for the weighted longitudinal perturbation of some quantity $X(z)$:

$$C_X(z) = -j\frac{\delta W}{\delta k(z)}\frac{\partial k}{\partial X}(z)\Big/\frac{\partial W}{\partial \omega} \tag{18.29}$$

where $k(z)$ is the wave propagation constant and W is the Wronskian defined in earlier sections.

The transient modulation response of the optical power and phase is defined as a function of the change in the carrier (N) and photon (S) longitudinal profiles. Using the shorthand notation of the dot product for $\int dz$, we write:

$$\frac{1}{2}P_0\frac{d\Delta P}{dt} = C_{Nr}\cdot\Delta N + C_{Sr}\cdot\Delta S \tag{18.30}$$

$$\frac{d\Delta\phi}{dt} = C_{Ni}\cdot\Delta N + C_{Si}\cdot\Delta S \tag{18.31}$$

where r and i indicate the real and imaginary parts of the perturbation functions.

We also assume that the current is acting as the modulation source which imposes the following current continuity equation at all z-points:

$$\frac{d\Delta N}{dt} = \Delta J - \Delta N/\tau_R(N) - v_g g\Delta S \tag{18.32}$$

where $\tau_R(N)$ is the carrier lifetime representing all recombination mechanisms except the stimulated emission.

Assuming that the modulation does not strongly distort the shape of the photon density profile, we can relate the variation of the power P (a scalar) to that of $S(z)$:

$$\Delta S(z)/S_0(z) = \Delta P/P_0 \tag{18.33}$$

Further assuming an input modulation $\Delta J = \Delta J_0(z)exp(j\Omega t)$, we can eliminate the carrier density fluctuation and obtain a useful formulation for the AM response:

$$\Delta P/P_0 = \frac{2C_{Nr}\cdot\Delta J\,/(j\Omega + 1/\tau_R)}{j\Omega + 2C_{Nr}\cdot v_g g S_0\,/(j\Omega+1/\tau_R) - 2C_{Sr}\cdot S_0} \tag{18.34}$$

The FM response can be derived in the same way from the imaginary terms of the perturbation functions. Re-using our results from the AM response, we get:

$$\Delta\omega = \frac{d\Delta\phi(t)}{dt} = \frac{C_{Ni}\cdot\Delta J}{j\Omega+1/\tau_R} + \left[C_{Si}\cdot S_0 - \frac{C_{Ni}\cdot v_g g S_0}{j\Omega+1/\tau_R}\right]\left(\frac{\Delta P}{P_0}\right) \tag{18.35}$$

18.6.2 Second Harmonic Distortion

The rate equations for the photon number $I_p(t)$ and the carrier density $N(z,t)$ can be written using the same notation as above:

$$\frac{dI_p}{dt} = 2I_p(t)\left\{C_{Nr}\cdot\Delta N + C_{Sr}\cdot\Delta S\right\} \tag{18.36}$$

$$\frac{dN}{dt} = J - R(N) - G(N,S)S \tag{18.37}$$

We assume that the injection current density is modulated at an angular frequency ω:

$$J(z,t) = J_s(z) + \Delta J(z)e^{j\Omega t} \tag{18.38}$$

To evaluate the second harmonic distortion (SHD), the carrier density and the photon number are expanded to second order:

$$N(z,t) = N_s(z) + \Delta N_1(z)e^{j\Omega t} + \Delta N_2(z)e^{j2\Omega t} \tag{18.39}$$

$$I_p(t) = I_{ps} + \Delta I_{p1}e^{j\Omega t} + \Delta I_{p2}e^{j2\Omega t} \tag{18.40}$$

In addition, to simplify the calculation, we make the assumption that

$$S(z,t) = \frac{S_s(z)}{I_{ps}}I_p(t) \tag{18.41}$$

Equations (18.36)–(18.41) are solved for ΔI_{p1} and ΔI_{p2}, following a straight forward small signal analysis. We then define the SHD as:

$$SHD = \frac{\Delta I_{p2}}{\Delta I_{p1}} \tag{18.42}$$

18.6.3 Noise Terms

The noise analysis in Tromborg's model follows the same approach as the modulation functions defined above. However, instead of a current source modulating the laser, additional noise terms are added to equations 18.30 and 18.31. The noise spectrum is then derived from the Fourier transform and correlation of these Langevin noise functions. The derivation of these terms is beyond the scope of this section and the user is referred to [106] for further details.

Part IV

SELECTED TUTORIAL EXAMPLES

PARS OPTICS GROUP

Chapter 19

APSYS EXAMPLES

19.1 Introduction

This chapter will describe a few selected examples of APSYS simulations. The relevant input files can be found in your local installation directory (default is c:\crosslig\apsys_examples): the section name corresponds to the examples's location on your hard drive. These examples have been selected to teach the basic operation of the software, explain key concepts essential to device simulation, showcase some popular applications/devices and preemptively answer common user questions.

Note that the examples from this chapter are only a small subset of the included tutorial files. There are a number of other examples that are supported through accompanying README files and comments in the simulation files. Crosslight support staff will also be happy to explain these examples in more detail if you need further assistance.

It is relevant to note that all examples are subject to change as the software is updated. Between each release, there are often bug fixes, material macro changes, new models and other improvements which can affect the results. In most cases, these should result in negligible changes in the results which do not alter the tutorial value of these examples. If any major inconsistencies are found, please report them to Crosslight support staff. Most examples listed here were last updated as part of the 2009 manual release. Whenever possible, we will note examples which have been updated since then.

Throughout this chapter, we will concentrate on the specific simulation statements and physical models that are needed for a particular device. Particular attention will also be given to model parameters that may affect the convergence or reliability of the simulation.

It is also assumed that the reader is familiar with the basic structure of the input files so some of the basic setup steps for will be omitted for the sake of brevity. If you have

not already done so, we strongly urge you to read through Chapter 3. Of particular interest will be the sections dealing with the use of the command-line setup tools as well as their integration into SimuCenter. If you are working using GUI tools such as LayerBuilder, additional online help is available through SimuCenter.

In some cases, the setup tools will only provide the basic framework of an input file and some direct editing may be required to complete the setup. If this is the case, new users are encouraged to use the built-in Wizard option in SimuAPSYS: this will present all the available statements and parameters. New statements may be created by right-clicking on an empty line and selecting the Wizard option: existing statements can be modified in the same way.

Note that users of other Crosslight tools besides APSYS may also benefit from this chapter. In many cases, the physical models described here are relevant to other devices that are not within the scope this software. While the input files themselves cannot always be reused, most of the syntax will carry over to other Crosslight tools.

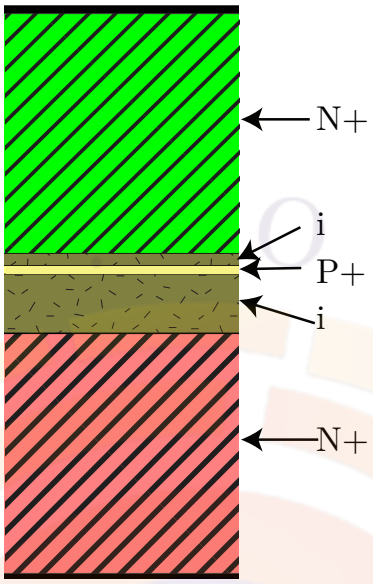


Figure 19.1: Schematic of camel diode

19.2 A_tutorial

This is example is a camel diode, which is related to the planar doped barrier diode. It is a *n-p-n* majority carrier diode which finds applications in high-speed switches, mixers, fast photodiodes, BARRITT devices and thermionic emission transistors. This particular device is adapted from[116].

Layer Structure

The major feature of this device is a thin highly-doped p-layer inside the intrinsic region as shown in Fig. 19.1. The corresponding layer file is shown below:

```
begin_layer
$
column column_num=1 w=1.  mesh_num=2  r=1.
$
bottom_contact column_num=1 from=0 to=1. contact_num=1
$
layer_mater macro_name=si column_num=1
layer d=1.  n=15  r=-1.1 n_doping1=1.5e24

layer_mater macro_name=si column_num=1
layer d=0.2480  n=30  r=-1.1 n_doping1=1e2
```

```
layer_mater macro_name=si column_num=1
layer d=0.0039  n=15  r=1. p_doping1=1.5e24 &&
  gaussian_tail=0.0001

layer_mater macro_name=si column_num=1
layer d=0.0480  n=30  r=-1.1 n_doping1=1.e2
layer_mater macro_name=si column_num=1
layer d=1.  n=15  r=-1.1 n_doping1=4.e24
$
top_contact column_num=1 from=0 to=1. contact_num=2
$
end_layer
```

One major difference from the device in the reference above is the lower doping concentration of the p+ layer. Since this is a current-blocking structure, we have chosen to lower the barrier height to make convergence easier. As explained in Sec. 4.10.1, there are many possible solutions of the current continuity equations in structures where the current is being blocked. In these conditions, the simulation is not stable since it may oscillate between different solutions that lie within the specified numerical precision.

One possible solution to this instability is to use relaxed convergence criterion such as a larger tolerance value (**var_tol** parameter) in the **newton_par** statement. As long as we find that the equation residue is low and that the currents on the electrodes sum up to zero (i.e. Kirchoff's law), we can consider the solution as valid. To maintain convergence with stricter tolerance values, more advanced techniques such as those of Chap. 4 may be required.

Simulation Setup

To run the simulation, use the following .sol file:

```
$file:camel.sol
$***********
begin
load_mesh mesh_inf=camel.msh
include file=camel.doping
include file=camel.mater
output sol_outf=camel.out

more_output ac_data=yes

newton_par damping_step=5. var_tol=1.e-9 res_tol=1.e-9 &&
```

```
  max_iter=100 opt_iter=15 stop_iter=50 print_flag=3

$ equilibrium solution is counted as scanline=1
equilibrium

newton_par damping_step=1 &&
  max_iter=30 opt_iter=15 stop_iter=15 print_flag=3

$ The first scan statement is counted as scanline=2
scan var=voltage_1 value_to=1.6

$ The first scan statement is counted as scanline=3
scan var=voltage_1 value_to=-0.5 max_step=0.05 print_step=0.2

end
```

Since both contacts are n-doped, it is difficult to determine whether a particular bias will put the device in a forward or reverse state. To capture both characteristics, we apply bias in one direction and then reverse it with the next scan statement. The shape of the I-V curve as well as the internal potential will help us identify the reverse bias region.

Note that we will also be interested in doing some AC post-processing analysis for this device so we use the **more_output** statement to generate the necessary auxiliary data files (.ac extension). Also note the numbering of the scan lines in the comments: this will be used to group together some of the data sets for later analysis.

Post-Processing

After the simulation, the results are post-processed in the .plt file as shown below:

```
$file:camel.plt
$ **************
begin_pstprc
plot_data plot_device=postscript

$ ---- plot structure data at first data set (equilibrium)

get_data  main_input=camel.sol sol_inf=camel.out &&
  xy_data=(1 1)

plot_1d variable=band from=(0.5 0.0) to=(0.5, 2.3)
plot_1d variable=donor_conc from=(0.5 0.0) to=(0.5, 2.3)
```

```
plot_1d variable=acceptor_conc from=(0.5 0.0) to=(0.5, 2.3)

$ ---- plot structure data at last data set (max. voltage bias)
get_data  main_input=camel.sol sol_inf=camel.out &&
  xy_data=(2 2)

plot_1d variable=band from=(0.5 0.0) to=(0.5, 2.3)
plot_1d variable=potential from=(0.5 0.0) to=(0.5, 2.3)
plot_1d variable=field_mag from=(0.5 0.0) to=(0.5, 2.3)

plot_1d variable=elec_curr_y from=(0.5 0.0) to=(0.5, 2.3)
plot_1d variable=hole_curr_y from=(0.5 0.0) to=(0.5, 2.3)
plot_1d variable=elec_conc from=(0.5 0.0) to=(0.5, 2.3)
plot_1d variable=hole_conc from=(0.5 0.0) to=(0.5, 2.3)

$ Plot bias dep. quantities using scanline specification.
$ Please make sure scan_data range is large enough to cover
$ the intended scanline.
get_data  main_input=camel.sol sol_inf=camel.out &&
  scan_data=(3 13)
plot_scan scan_var=voltage_1 variable=current_1 scanline=3

$ ====> AC analysis as a post-processing step

$ AC analysis at max. voltage
get_data  main_input=camel.sol sol_inf=camel.out &&
  xy_data=(2 2)

$ AC analysis vs. freq.
ac_voltage log_freq1=6. log_freq2=10. contact_num=1
plot_ac_curr variable=capacitance_1
plot_ac_curr variable=conductance_1

$ Physical distribution of AC current @ 1 MHz
ac_voltage current_distr=yes log_freq1=6. log_freq2=6.
plot_1d_ac_curr variable=elec_curr_y imag_part=no from=(0.5 0.) to=(0.5  2.3)
plot_1d_ac_curr variable=hole_curr_y imag_part=no from=(0.5 0.) to=(0.5  2.3)

$ We plot capacitance-voltage and conductance-voltage here at 1 MHz
$ To generate reasonable AC vs. bias plots, please save sufficient
$ number of data sets.  AC analysis plots cannot include equilibrium data
get_data  main_input=camel.sol sol_inf=camel.out &&
  scan_data=(3 13)
```

```
ac_voltage log_freq1=6. log_freq2=6. contact_num=1 &&
  freq_point=2 versus_bias=yes scanline=3

set_xydata_for_scan scan_var=voltage_1
plot_ac_curr variable=capacitance_1
plot_ac_curr variable=conductance_1

$ For purpose of demonstrating the plotting,
$ let us plot y parameters assuming input=output=eletrode No. 1
ac_parameters versus_bias=yes log_freq1=6.  log_freq2=6. &&
  input_contact=1 output_contact=1 freq_point=2 scanline=3

plot_ac_parameters parameter_type=y smith_chart=no

end_pstprc
```

The I-V curve of the device is shown in Fig. 19.2. Note that we use scan line #3 to capture the part of the IV curve that has both negative and positive bias (i.e. the third scan statement). From the shape of the I-V curve, we see that the positive bias on electrode #1 yields reverse bias of the diode. This is confirmed by the internal potential profile shown in Fig. 19.3: the potential of the p+ region is below that of both n+ regions so the two junctions are reverse-biased.

At the maximum bias point, we also find there is a strong local field (Fig 19.4): this indicates that even though we have neglected it for the sake of simplicity, impact ionization may be significant in this region.

As mentioned before, we also want to do small-signal AC analysis on this device. This is done via the **ac_voltage** statement to apply 1 V of AC voltage on a particular electrode: all other electrodes are connected to the AC ground even though they may have a different DC bias.

Depending on the quantity we are interested in, there are different kinds of AC analysis:

- AC quantity vs. frequency at a particular bias point
- AC quantity vs. bias at a particular frequency
- Physical distribution of AC current at a particular bias and frequency

All of the above analyses are done in the above .plt file. For example, the capacitance vs. bias plot is shown in Fig. 19.5. Since these different kinds of AC analysis are mutually exclusive, the **ac_voltage** statement is issued multiple times to generate new AC data before each plot.

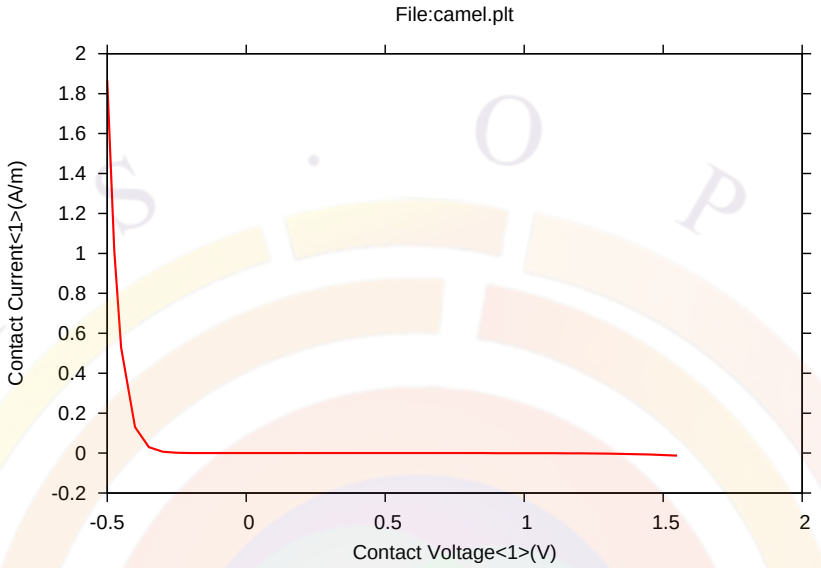


Figure 19.2: I-V curve of low-barrier camel diode

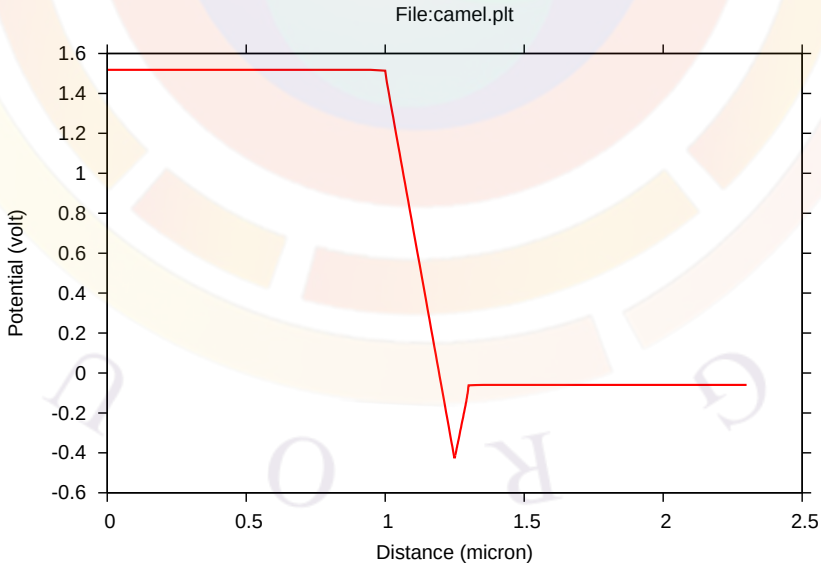


Figure 19.3: Internal potential profile of camel diode under reverse bias

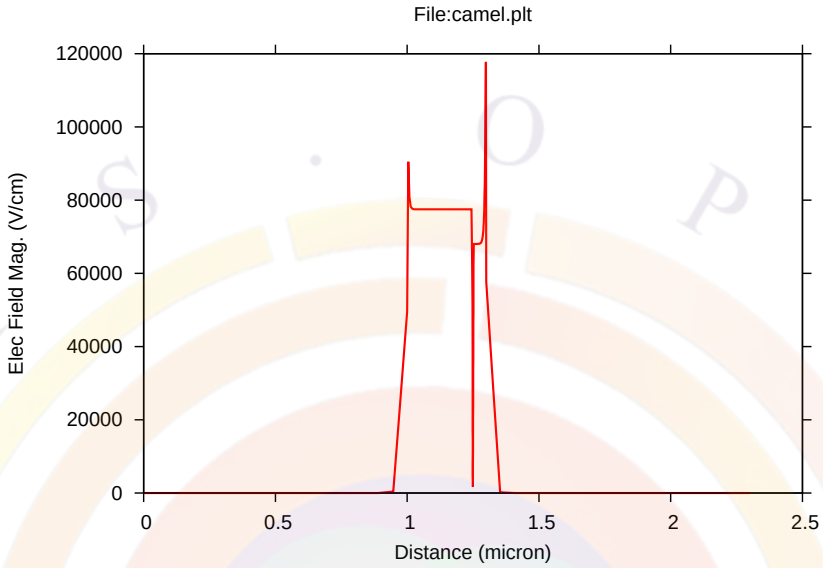


Figure 19.4: Local field magnitude of camel diode under reverse bias

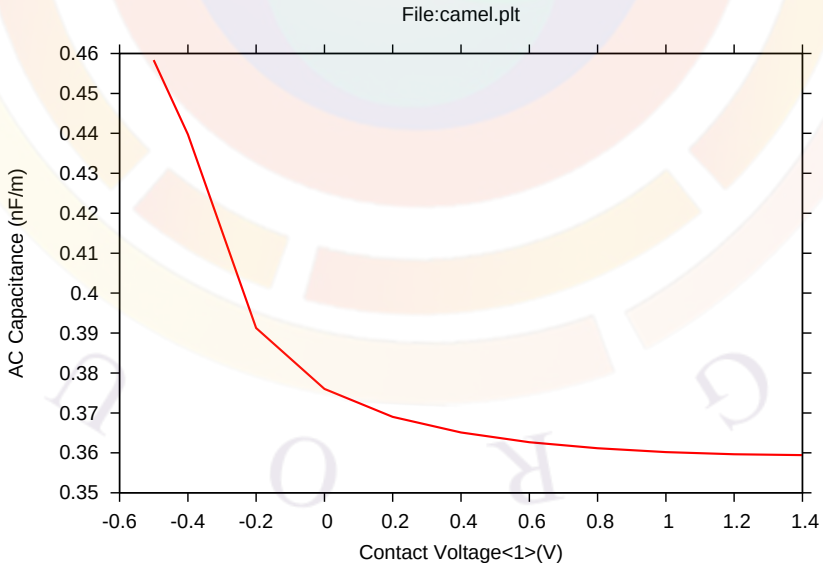


Figure 19.5: Capacitance of camel diode vs. bias

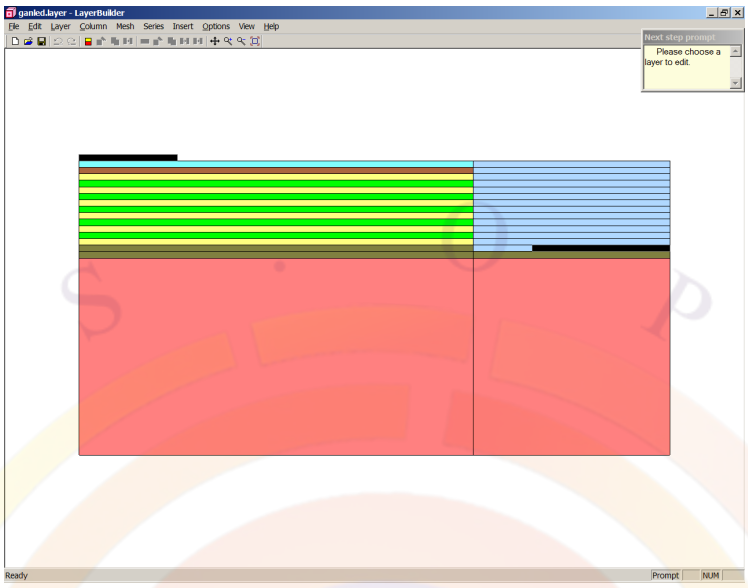


Figure 19.6: LED structure as shown in LayerBuilder

19.3 LED_GaN_MQW\2d\InGaN

This example[1] is a light emitting diode (LED) based on multiple $In_xGa_{1-x}N$ quantum wells. Modeling of nitride devices is an active area of research and so requires special care for accurate modeling: as a result, this example will be more complicated than usual.

Since these devices are grown on an insulating sapphire substrate, both contacts are on the top of the device (Fig. 19.6). The transparent indium tin oxide (ITO) contacts are represented here as ideal ohmic contacts for the purposes of the electrical simulation. However, these contacts are not completely transparent and have optical properties that need to be considered: this will be done as part a ray tracing simulation in the post-processing stage.

Layer Structure

The layer file used to define this device is shown below:

```
$ ----
begin_layer
$
$ this has been moved to ganled.sol
$set_polarization ref_column=1 screening=0.5
```

[1] Last updated July 2011

```
independent_mqw
column column_num=1 w=200  mesh_num=10  r=0.85
column column_num=2 w=100  mesh_num=8  r=1.15
top_contact column_num=1 from=0.0 to=50 contact_num=2
top_contact column_num=2 from=30 to=100 contact_num=1
$

layer_mater macro_name=sapphire  column_num=1
layer_mater macro_name=sapphire  column_num=2
layer d=100.  n=8   r=0.7

layer_mater macro_name=algan var1=0 column_num=1 var_symbol1=x
layer_mater macro_name=algan var1=0 column_num=2 var_symbol1=x
layer d=2.5  n=8   r=0.8 n_doping1=5e24 n_doping2=5e24

layer_mater macro_name=algan var1=0 column_num=1 var_symbol1=x
layer_mater macro_name=void  column_num=2
layer d=0.5  n=5   r=0.8 n_doping1=5e24

$ MQW region
include file=ganled.bar
include file=ganled.qw
include file=ganled.bar
include file=ganled.qw
include file=ganled.bar
include file=ganled.qw
include file=ganled.bar
include file=ganled.qw
include file=ganled.bar
include file=ganled.qw
include file=ganled.bar

$ for superlattice, we use effective medium theory and anisotropic
$ mobility and thermal conductivity, 24 SL
layer_mater macro_name=algan var1=0.07 column_num=1 var_symbol1=x
layer_mater macro_name=void column_num=2
layer d=0.18 n=6  r=1.0 p_doping1=3e23

$p+
layer_mater macro_name=algan var1=0 column_num=1 var_symbol1=x
layer_mater macro_name=void column_num=2
layer d=0.015 n=4  r=1.0 p_doping1=1.2e24
```

```
$
end_layer
```

Note the use of the **include** statement used to define the barrier and well regions: this is a common technique to more easily define periodic structures. The main advantage is that any change in well or barrier composition only needs to be done once instead of for each well. When using direct input of the layer file, it also makes the repeated structure of the MQW more obvious. The barrier and well input files are shown below:

```
$
layer_mater macro_name=ingan var1=0.0 column_num=1 var_symbol1=x &&
  n_doping=3.e23
layer_mater macro_name=void column_num=2 n_doping=3.e23
layer d=0.015 n=12  r=-1.4

$
layer_mater macro_name=ingan var1=0.11 &&
column_num=1 active_macro=InGaN/InGaN &&
avar1=0.11 avar2=0. &&
avar_symbol1=xw avar_symbol2=xb var_symbol1=x

layer_mater macro_name=void column_num=2 active_macro=void

layer d=0.0022 n=8  r=-1.3
```

Users working with LayerBuilder can also easily build a MQW region by copying & pasting layers in the GUI.

We note that we have used the algan macro for many layers but that the composition is set to $x = 0$. In truth, those layers are simply GaN and we could have used the gan macro. However, using the ternary/quaternary macro is often a good idea in this situation: since the same interpolation formulas are used everywhere, it ensures that there is no conflict in the material parameters from different macros.

Aside from the MQW region, there is one other layer of particular importance: the AlGaN layer with $x = 0.07$ near the top of the device. This layer is special in that we use it to represent a superlattice region: the composition is averaged and represents an effective medium. Superlattices are often used to block leakage current and have different properties than a material having the same composition as the effective medium: this will be handled in the .sol file later on.

There are a few other points of importance in this layer file which are related to the interface charges between layers (common in c-plane nitride materials). These

settings are not available in the LayerBuilder GUI and must be added manually afterwards.

The first point is the **set_polarization** statement: this instructs the layer.exe program to automatically calculate the interface charge between layers and add them to the .mater file. However, this model is considered deprecated as of the 2011 version of APSYS and has been commented out. A more accurate model based on material macros will be discussed below. The older model is still available and many other wurtzite examples have not been ported to the new syntax yet. We strongly recommend the use of the new model in any new simulation project. However, care should be taken not to use the new and old models at once as this will double-count the interface charges.

Because this simulation will have interface charges, the bands will be tilted by the local field these charges generate. In order to move beyond the flat-band approximation in the QW mdoel, a self-consistent simulation will need to be used. Since the field value will not necessarily be uniform, we also use the **independent_mqw** statement. This forces the software to assign different material numbers to each QW and explicitly solve the Schrödinger equation for each one; by the default, the software would normally try to save time by re-using the QW solution of wells with identical compositions. See subsection 8.2.4 for details.

Simulation Setup

To run the simulation, we use the following .sol file:

```
$file:ganled.sol
$ *******
begin
load_mesh mesh_inf=ganled.msh
output sol_outf=ganled.out
more_output qw_states=yes
$ *******
include file=ganled.doping
include file=ganled.mater

polarization_charge_model screening=0.5 vector=(0 1 0)
set_active_reg tau_scat=0.4e-13
modify_qw tail_energy=0.03

$self-consistent is necessary for polarization
self_consistent  wave_range=0.005
```

```
$ Quantum transport model helps deal with thin,deep wells
q_transport

$ .mater shows superlattice is mater=9
mobility_xy dir=y factor_elec=0.2 mater=9
thermal_kappa_xy dir=y factor=0.1 mater=9
$ n-layer mobility may be enhanced due to SL design
max_electron_mob value=1 mater=2
min_electron_mob value=1 mater=2

$ Isothermal temperature at equilibrium
temperature temp=300

$ Turn on self-heating model
heat_flow damping_step=1

$ thermal_cond at contact here will determine self-heating
contact num=2 type=ohmic thermal_type=3 &&
  thermal_cond=200. extern_temp=300

$ ----- initialize optical constatns------
set_wavelength wavelength=0.40 backg_loss=2000

$ Set LED model to "simple": we will calculate actual
$ extraction efficiency with raytracing later
led_simple wavelength=0.40 spectrum_num=50

$ Export raytracing data: convert 2D electrical simulation into
$ 3D raytracing boxes
export_raytrace ray3d_convert=yes

$
$ start solving
$
newton_par damping_step=5. max_iter=100 print_flag=3
equilibrium

newton_par damping_step=1. print_flag=3
scan var=voltage_1 value_to=-10 init_step=0.1 max_step=0.5 &&
  auto_finish=current_1 auto_until=1.0 auto_condition=above

scan var=current_1 value_to=600.  print_step=150 &&
  init_step=1 min_step=1e-3 max_step=30
```

```
end
```

As mentioned earlier, the 2011 version of APSYS introduces a new model for the interface charges present in most wurtzite-based devices. This model is in two parts: updated material macros which define the **polarization_charge** formulas and the **polarization_charge_model** which activates it. When active, this model computes the fixed interface charge in a particular region by examining the polarization vector on each side of the interface. Graded regions (not used here) automatically produce a 3D distribution of fixed charges with this new model.

The first statement is included in the new material macros and defines the spontaneous and strain-induced polarization. By default, all our macros assume the existence of a GaN buffer which determines the strain in the device; the macro should be adjusted if another type of buffer layer (such as AlN) is used. A sample declaration can be seen below and uses a well-established formula from theory[117]:

```
polarization_charge variation=function
function(temper)
substrate_latt=3.189;
alattice_GaN=3.189;
P_sp=  -0.034;
a_latt=alattice_GaN;
strain=(substrate_latt - a_latt)/a_latt;
P_pz2 = -0.918*strain +9.541*strain**2;
P_pz_all =  P_pz2;
P_sp + P_pz_all
end_function
```

The second statement modifies this polarization vector and adds a screening coefficient to account for deviations from theory due to compensation by defects and other effects. It also defines the direction of the polarization vector: here (010) indicates that the c-plane growth axis is in the +y direction, like most conventional devices. This may be altered when studying non-polar or semi-polar devices.

To turn on the self-consistent QW calculations, we use the **self_consistent**. This model allows the local field generated by the interface charges to affect the QW states and gain/spontaneous emission calculations. When this model is turned on, .qws output files are created which contain details of the Schrödinger solver calculations. As we will see later, we can use this to plot the QW wave functions.

The gain calculation are also modified by the **set_active_reg** and **modify_qw** statements. The first is used to modify the broadening width and the second introduces a tail into the bandgap region to create inhomogeneous broadening.

In many MQW nitride devices, we have found that the standard drift-diffusion model greatly overestimates the turn-on voltage. It is also believed that carrier leakage is an important effect which contributes to efficiency droop [2]. Both of these effects may be explained by the fact that these devices have very deep and thin wells where the classical drift-diffusion model fails and non-local transport mechanisms play an important role. We have implemented this in the **q_transport** statement: please consult the reference section for details. This subject is highly experimental and careful calibration of parameters may be required.

As indicated above, we have a superlattice layers has different properties than the effective medium. We introduce anisotropic behavior with the **mobility_xy** and **thermal_kappa_xy** statements. These will scale the mobility and thermal conductivity of the effective medium in the direction of the superlattice periodicity.

Note that for electrical modeling of superlattices, there is also a miniband tunneling model available (see reference section for **tunneling**). This model is not used here for simplicity and because it does not include any of the thermal effects associated with a superlattice.

For thermal modeling, we activate self-heating with the **heat_flow** statement. The thermal boundary conditions are important here and determine how much the active region temperature increases. The **contact** statement must be re-issued in .sol to override a default thermal boundary setting. In truth, thermal modeling is not strictly required for LED devices such as this; the current density is not as high as in laser devices the thermal increase is not sufficient to strongly affect material parameters. We include this effect here purely for its tutorial value. For a more detailed discussion on thermal modeling, consult Sec. 20.3.

For all LED modeling, a key concern is how much light is actually output from the device (i.e. the extraction efficiency). This requires an optical model which means that the optical constants for the various materials must be initialized in the code. For historical reasons, this used to be done with the **init_wave** statement which is also used to define waveguides.

This is still the method which should be used when using the LED model of Sec. 14.2 and the **led_control** statement. However, we have decided to use the simpler model of **led_simple** in this simulation so the optical constants are initialized with **set_wavelength** instead.

In the **led_simple** model, the solver makes no attempt to calculate the extraction efficiency and a value must be provided. The LED optical output will be assumed to be the total spontaneous emission source power multiplied by this coefficient. However, we still want to calculate the extraction efficiency rather than rely on an estimated value.

[2] This is still the subject of active research. Other explanations based on Auger recombination mechanisms have also been suggested

To do this, we will use the ray tracing approach of Sec. 14.5. This is a post-processing step but it requires that some necessary data be output during the main solution. This is done with the **export_raytrace** statement. There is a conversion to be done here since the electrical simulation is in 2D whereas the ray tracing model operates on 3D objects.

The rest of the .sol file should be simple to understand but we mention in passing the auto_finish parameter in the voltage scan. This ensures that we transition from voltage to current control once there is at least a little bit of current flowing. More details on the reasons why this is necessary can be found in Chapter 4.

Post-Processing

In this simulation, there is a lot of post-processing to be done so we split things into separate, more manageable files. Note that we use the same root name for all the post-processing files: the extensions also all start with .plt as a matter of convention.

The first file is ganled.plt and is used to plot the results from the electrical/thermal simulation as shown below:

```
$file:ganled.plt
$ ***************
begin_pstprc
plot_data plot_device=postscript

get_data  main_input=ganled.sol sol_inf=ganled.out &&
  scan_data=(1 6)

plot_scan scan_var=voltage_1 variable=current_1 scale_horizontal=-1
plot_scan scan_var=current_1 variable=led_power
plot_scan scan_var=current_1 variable=led_effi

get_data  main_input=ganled.sol sol_inf=ganled.out &&
  xy_data=(4 6)
led_spectrum
gain_spectrum variable=gain

get_data  main_input=ganled.sol sol_inf=ganled.out &&
  xy_data=(6 6)

plot_1d variable=band from=(100. 100.0) to=(100. 104.)
plot_1d variable=band from=(100. 102.98) to=(100. 103.12)
plot_1d variable=band from=(100. 102.98) to=(100. 103.12) &&
```

```
  qw_wave=1 qw_wave_ht=0.3
plot_1d variable=elec_conc from=(100. 102.98) to=(100. 103.12)
$-----------------------
plot_2d variable=lattice_temp grid_sizes=40 40
plot_2d variable=potential grid_sizes=40 40

plot_2d variable=total_curr grid_sizes=20 20  &&
  point_ll=(0. 100.) point_ur=(300 103.296) &&
  xrange=(0. 300) yrange=(100 103.296)
plot_2d variable=hole_curr grid_sizes=20 20  &&
  point_ll=(0. 100.) point_ur=(300 103.296) &&
  xrange=(0. 300) yrange=(100 103.296)
plot_2d variable=elec_curr grid_sizes=20 20 &&
  point_ll=(0. 100.) point_ur=(300 103.296) &&
  xrange=(0. 300) yrange=(100 103.296)
plot_2d variable=lattice_temp grid_sizes=20 20 &&
  point_ll=(0. 100.) point_ur=(300 103.296) &&
  xrange=(0. 300) yrange=(100 103.296)
plot_2d variable=potential grid_sizes=20 20 &&
  point_ll=(0. 100.) point_ur=(300 103.296) &&
  xrange=(0. 300) yrange=(100 103.296)

end_pstprc
```

The I-V and curve is shown in Fig. 19.7 while the light output is shown in Fig. 19.8. Note that this "broad-area power" is the output of the LED model chosen in .sol. In the simplified model we have chosen, the extraction efficiency is assumed to be 10%. We will see below that the ray tracing model predicts a very different value.

The internal quantum efficiency is also an important quantity in LED modeling and many efforts have been made to study the origin of the reduced efficiency at high current density (i.e. "IQE droop"). In Fig. 19.9, we see the calculated IQE curve which shows this effect.

As mentioned earlier, self-consistent simulations allow us to plot the wave functions of the quantum wells. This is shown in Fig. 19.10. In the original .plt file, we use the **qw_wave_ht parameter** to scale the wave functions vertically (for aesthetic reasons).

As of the 2011 version of APSYS, the ray tracing model has been moved to a separate program called Optowizard where we also split things into separate files. The raytrace.sol has statements that configure and launch the external ray tracing program. This can be a time-consuming step so we separate it from the raytrace.plt file which plots the results of the last ray tracing simulation:

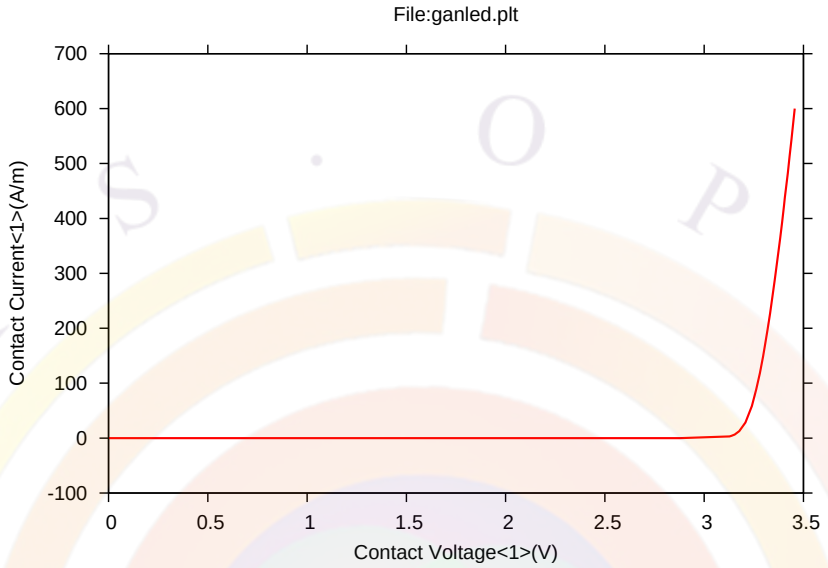


Figure 19.7: I-V curve of InGaN LED

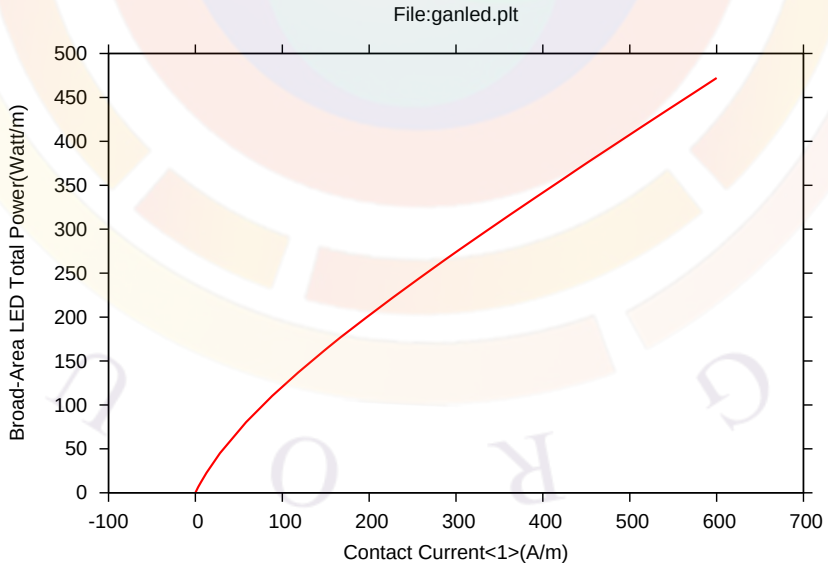


Figure 19.8: Broad-area power vs. current for InGaN LED

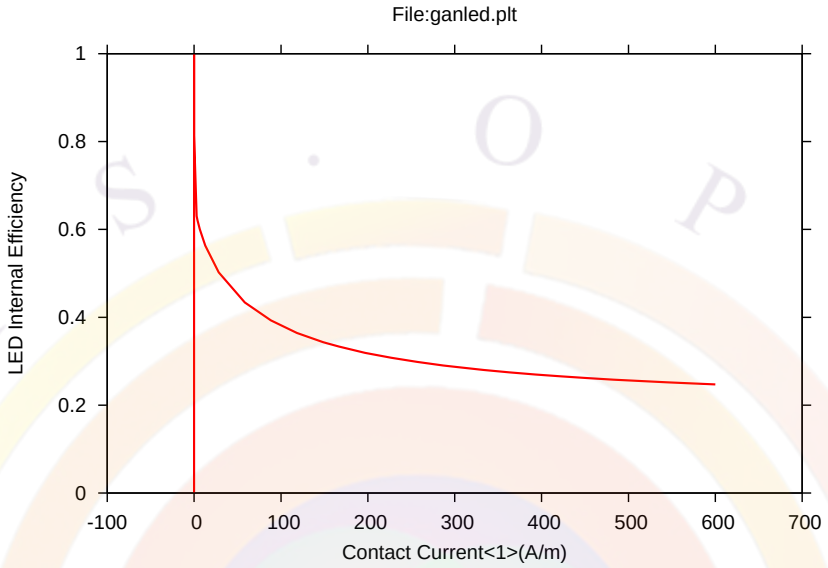


Figure 19.9: IQE curve of InGaN LED

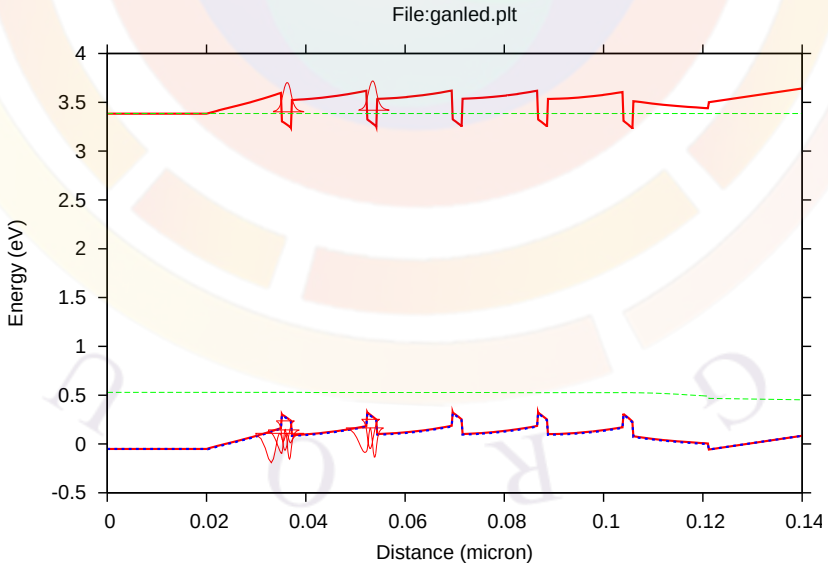


Figure 19.10: Band diagram and wave functions of InGaN LED

```
begin_optowizard
setup_raytrace filebase=ganled


rt3d_contact_reflector transp_contact=yes &&
      contact1_compindex=(2. 0.5) contact1_thick=0.05 &&
      contact2_compindex=(2. 0.5) contact2_thick=0.05

$ No dome
do_raytrace_3d precision=0.01 datafile_range=(2,6) initial_rays=8000

$ With dome external package
$do_raytrace_3d precision=0.01 datafile_range=(2,6) initial_rays=8000  &&
$  external_package=dome working_direction=+y &&
$  dome_cyl_height=110 dome_base_thickness=10 &&
$  dome_cyl_radius=500 dome_sph_radius=500 &&
$  package_refr_index=(1.7,1.0e-6)


end_optowizard


begin_optowizard
plot_data plot_device=postscript
get_raytrace_data filebase=ganled xy_data=(6 6) scan_data=(2 6)

$ Plot previously generated RT results

3drayplot_bias variable = total_source
3drayplot_bias variable = transmitted
3drayplot_bias variable = transmitted relative=yes
3drayplot_phi theta=90.0  smooth=yes

end_optowizard
```

We note the use of the **rt3d_contact_reflector** statement in the setup of the ray tracing simulation: this sets up the optical properties of the contacts. In this case, we have semi-transparent ITO contacts with a thickness of 0.05 μm and a complex refractive index of $2 + 0.5i$.

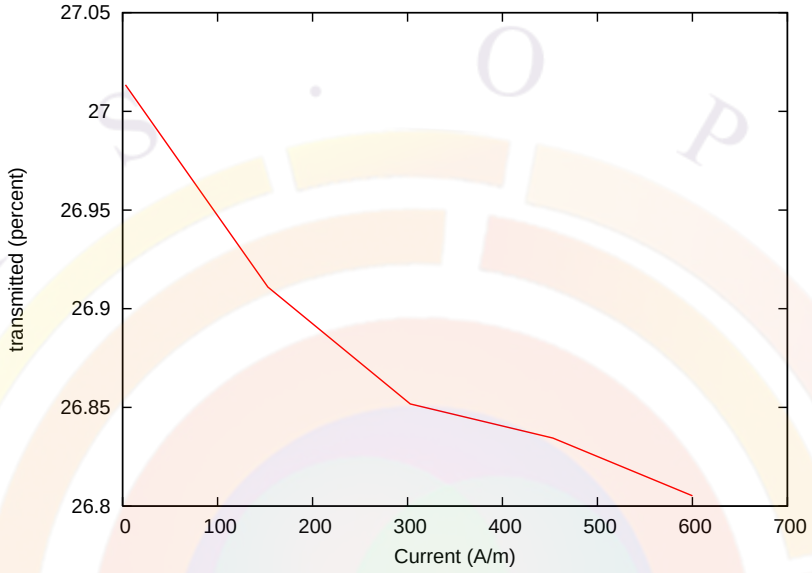
Users of previous versions of APSYS may remember that the contact optical properties were defined in the **export_raytrace** statement. Moving this to the post-processing avoids having to re-run the electrical simulation to study how the optical properties of the contacts affect the light extraction.

We also note that there are two set of ray tracing commands; one of them is commented out so we do not do two ray tracing simulations in sequence. The first operates on the LED itself while the other encapsulate the LED in a hemispherical dome with a mirrored flat bottom. We see in Fig. 19.11 and Fig. 19.12 how the dome affects the extraction efficiency and emission pattern of the LED.

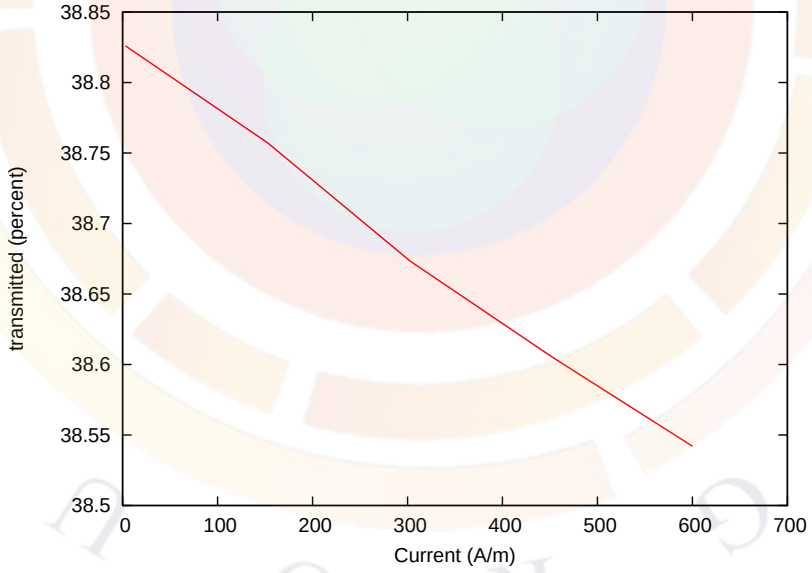
The ray tracing model shows in Fig. 19.11 that our earlier estimate of 10% in **led_simple** was inaccurate: even without encapsulation, we expect extraction efficiency above 25%. With encapsulation, this can rise to above 35%.

The output log of the ray tracing program also provides clues to help in further optimization of the design. For example, the power loss due to various reasons is accounted for separately and printed. In this case, we see there is a lot of power loss in the contacts:

```
..(ETC...)...
 Total source power (W/m)
   472.380007985757
 TRANSMITTED POWER in absolute and relative to emitted (%) units
        126.663735282548        26.8139491810093
 SEMICONDUCTOR ABSORBED POWER in absolute and relative to emitted (%) units
        88.3668556033336        18.7067306214191
 CONTACTS ABSORBED POWER in absolute and relative to emitted (%) units
        257.349417099875        54.4793201975716
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

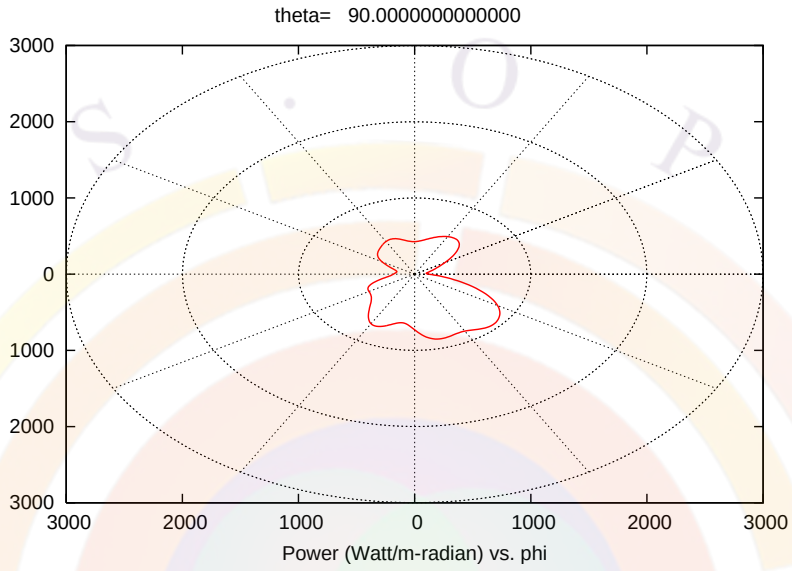


(a)

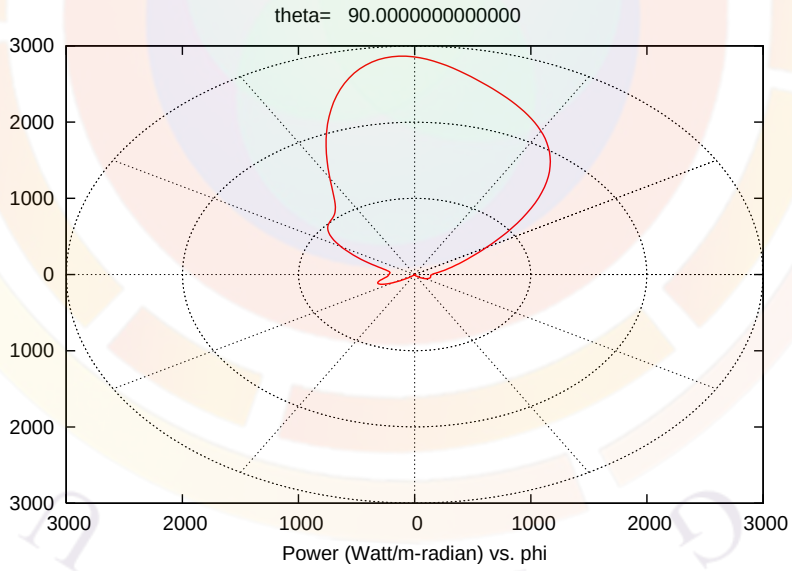


(b)

Figure 19.11: (a) Extraction efficiency of InGaN LED. (b) With dome encapsulation.



(a)



(b)

Figure 19.12: (a) Angular emission pattern of InGaN LED. (b) With dome encapsulation.

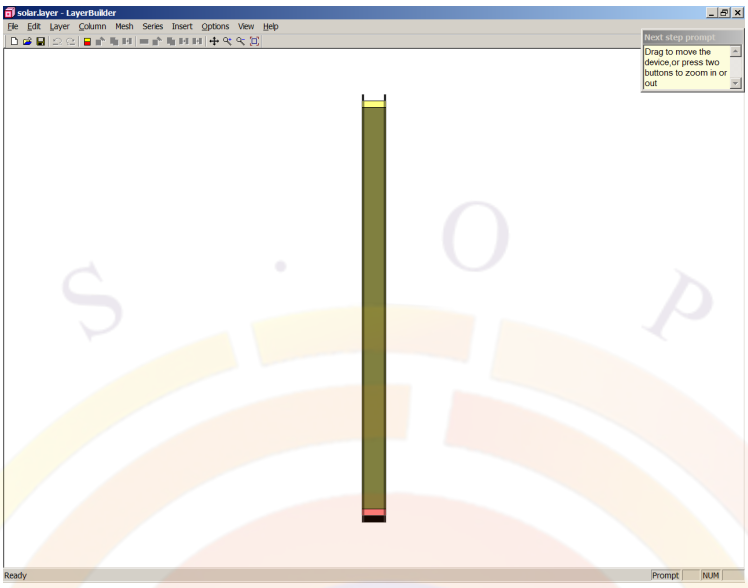


Figure 19.13: Solar cell structure as shown in LayerBuilder GUI

19.4 solar_cell\Si_simple

This example is a simple bulk crystalline silicon solar cell with contact lines on the top side. Since a practical device must consider the effects of shading by the contacts, we make this a 2D device (see Fig 19.13). Note that because of symmetry, we could conceivably cut this device in half to save on mesh.

Layer Structure

The device is built p-side up with a 100 μm lightly doped intrinsic region: this is where we want most of the absorption to take place so the drift current can dominate the transport. The width of the device is 6 μm with a 5 μm space between the contacts:

```
$file:solar.layer
begin_layer
column column_num=1 w=0.5 mesh_num=4  r=1.
column column_num=2 w=5 mesh_num=12  r=-1.2
column column_num=3 w=0.5 mesh_num=4  r=1.
bottom_contact column_num=1 from=0 to=0.5 &&
  contact_num=1 contact_type=ohmic
bottom_contact column_num=2 from=0 to=5 &&
  contact_num=1 contact_type=ohmic
bottom_contact column_num=3 from=0 to=0.5 &&
```

```
  contact_num=1 contact_type=ohmic
$
layer_mater  macro_name=si column_num=1
layer_mater  macro_name=si column_num=2
layer_mater  macro_name=si column_num=3
layer d=  0.5 n= 6 r=  0.8000E+00 n_doping1=0.2000E+26 &&
  n_doping2=  0.2000E+26 n_doping3=  0.2000E+26
$
layer_mater  macro_name=si  column_num=1
layer_mater  macro_name=si  column_num=2
layer_mater  macro_name=si  column_num=3
layer d=100. n= 100 r=  0.8000E+00 n_doping1=  0.2000E+22 &&
  n_doping2=  0.2000E+22 n_doping3=  0.2000E+22
$
layer_mater  macro_name=si  column_num=1
layer_mater  macro_name=si  column_num=2
layer_mater  macro_name=si  column_num=3
layer d=0.5 n=  6 r=  0.1200E+01 p_doping1=  0.5000E+26 &&
  p_doping2=  0.5000E+26 p_doping3=  0.5000E+26
$
top_contact column_num=1 from=0 to=0.5 &&
  contact_num=2 contact_type=ohmic
top_contact column_num=3 from=0 to=0.5 &&
  contact_num=2 contact_type=ohmic
end_layer
```

We note in the above layer file that most of the mesh points are allocated near the top of the device: this is necessary in a solar cell. Since the light decays exponentially, most of the recombinations occur near the surface so this region must be well-sampled. If it is not, then the linear interpolation can overestimate the optical generation rate and, in some cases, produce results with over 100% efficiency.

Simulation Setup

To run the simulation, we use the following .sol file:

```
$file:solar.sol
$***********
begin
load_mesh mesh_inf=solar.msh
include file=solar.mater
include file=solar.doping
```

```
output sol_outf=solar.out
more_output light_reflection=yes

$ Attach a 50 ohm resistor on the top contact.
$ Assume cell depth of 10 cm = 1e5 um
$external_cir resistance=50 z_dim=1e5 contact=2

minispice circuit_file=res.cir z_dim=1e5 &&
contact2_to_spice_node=0 spice_node_scan=1

newton_par damping_step=5. var_tol=1.e-9 res_tol=1.e-9 &&
  max_iter=100 opt_iter=15 stop_iter=50 print_flag=3
$
$ Solve for equilibrium condition.
$
equilibrium

newton_par damping_step=1. var_tol=1.e-4 res_tol=1.e-5 &&
  max_iter=30 opt_iter=15 stop_iter=15 print_flag=3 &&
  change_variable=yes
$
$ Turn up the light
$
scan var=light value_to=1. print_step=1. &&
  init_step=0.01 min_step=1.e-5 max_step=0.1
$
$ Apply a forward bias to offset the photo current
$
scan var=voltage_1 value_to=-0.7 &&
  init_step=0.01 min_step=1.e-5 max_step=0.02

$
$ photo-sensitive device:
$
light_power spectrum_file=solar.am15 light_dir=top profile=(0.5 5.5 0.1 0.1) &&
  angle=0.0
optic_coating  spectrum_file=solar.sio2 thickness=0.05
index_spectrum spectrum_file=solar.silicon mater=1

end
```

The most important feature of this .sol file is the way in which the incident light is

defined. This is done with the **light_power** statement as discussed in Section 12.7. Here, we choose to use the AM15 solar spectrum as the optical pumping source: it is provided in the solar.am15 file in the simulation directory.

The light input is confined to the center of the device to simulate the shadowing effect of the contacts. However, should the user actually want to model the optical decay in semi-transparent contacts, actual metal layers need to be used: contact regions in a layer file are merely equipotential boundary regions with no optical properties of their own.

In addition to the light input, the boundary conditions are important. In this example, we use a 50 nm layer of SiO_2 on the top surface as a simple antireflection layer. The bottom surface is left unspecified which means that it is assumed to be transparent: a boundary condition can be applied with the **back_reflection** statement.

Another point to consider in solar cell simulations is that we must deal with a very broad range of wavelengths. Therefore, a single value of the complex refractive index (n, k) is inappropriate and we use the **index_spectrum** statement to provide experimental n, k values the software can use.

A common concern on solar cell performance is how the device will behave when external circuit elements are taken into account. We show on to model this by providing an external resistance with the **external_cir** statement.

Now that we have explained the basic setup of the simulation, let us move on to the actual simulation. As explained in Section 12.7, the initial equilibrium calculations describe a state of thermal equilibrium with no external bias. Since optical pumping is a form of bias, it needs be turned on via a scan statement: when the "light" variable reaches 1.0, then the full value of the light defined in **light_power** is applied to the device. Larger values can be used to model the effects of concentrators and multi-sun illumination.

After the first scan statement is over, the current has reached its maximum value. In order to get the open-circuit voltage (V_{oc}), we must apply a forward bias large enough to cancel the photocurrent.

Post-Processing

After the simulation, we can plot the results with the following .plt file. Note that we use .plt here in order to use some post-processing parameter extraction commands that are not yet available in CrosslightView.

```
$file:solar.plt
$ **************
begin_pstprc
```

```
plot_data plot_device=postscript
$ before light is on
get_data  main_input=solar.sol sol_inf=solar.out &&
  xy_data=(1 1)

plot_multilayer_optics variable=absorption
plot_multilayer_optics variable=reflection

plot_1d variable=band from=(0.5 0.0) to=(0.5, 101.0)

plot_1d variable=incident_power from=(0.5 0.0) to=(0.5, 101.0)
plot_2d variable=incident_power
plot_3d variable=incident_power

$ after light is on
get_data  main_input=solar.sol sol_inf=solar.out &&
  xy_data=(2 2)
plot_1d variable=band from=(0.5 0.0) to=(0.5, 101.0)
plot_2d variable=total_curr
plot_2d variable=incident_power
plot_3d variable=incident_power

$ I-V curve
get_data  main_input=solar.sol sol_inf=solar.out &&
  scan_data=(2 3)

plot_scan scan_var=voltage_1 variable=current_2 &&
  scale_vertical=1 scale_horizontal=-1

$ Scale from A/m to A/m^2 = 1/5e-6 = 2e5
$ Use parameter extraction on this curve to get Isc, Voc and max. power
para_extract file=Isc.txt fit_hori_from=0.02  &&
    fit_hori_to=0.2 type=fit_line
plot_scan scan_var=voltage_1 variable=current_2 &&
  scale_vertical=2e5 scale_horizontal=-1 &&
  user_ylabel=A/m^2 yrange=[0 300]

para_extract file=Voc.txt hori_intercept=0
plot_scan scan_var=voltage_1 variable=current_2 &&
  scale_vertical=2e5 scale_horizontal=-1 &&
  user_ylabel=A/m^2 yrange=[0 300]

para_extract file=power.txt xy_hori_from=0  xy_vert_from=0 &&
```

```
  type=xy_product_search
plot_scan scan_var=voltage_1 variable=current_2 &&
  scale_vertical=2e5 scale_horizontal=-1 &&
  user_ylabel=A/m^2 yrange=[0 300]

$ Use parameter extraction to get max. effiency
$ Power result above is in W/m^2 so this time, we normalize
$ by the total power intensity of the AM15 solar spectrum: 963.56 W/m^2
para_extract file=eff.txt xy_hori_from=0  xy_vert_from=0 &&
  xy_product_scale=1.037818e-3 type=xy_product_search
plot_scan scan_var=voltage_1 variable=current_2 &&
  scale_vertical=2e5 scale_horizontal=-1 &&
  user_ylabel=A/m^2 yrange=[0 300]

end_pstprc
```

It is also useful to use the .plt to scale the axes into more convenient plotting units. Recall that the current continuity equations are formulated in terms of the local current density (A/m^2). However, in a 2D simulation, the solver reports the linear current density in A/m. This is done by integrating the local current density at various mesh points over the width of the contact region. Similarly, dividing this value by the contact width defines an average current density.

However, current density is a spatially-dependent quantity so the average density varies depending on where it is measured in the device: for example, the value defined above would be the average density at that particular contact. In a laser or LED, the user might care about the density in the active region and would use a mesa width or ridge width.

For a solar cell, we choose to divide by the size of illuminated area (5 μm) in order to better compare with experiments. When running their own comparisons, users are strongly encouraged to verify how the current density value has been defined experimentally.

Using this scale factor allows us to plot the I-V curve of Fig. 19.14:

Note this figure appears several times in the output of the post-processing step: this is because the **para_extract** statement operates on the next plot in the .plt file. Since we wish to extract several different parameters, we need to plot the I-V curve several times.

From this extraction, we note that $I_{sc} = 260.68 A/m^2$ and $V_{oc} = 0.68115V$. The maximum power product of the I-V curve is $P = 142.69 W/m^2$. The maximum efficiency (14.81%) can be computed from this value by dividing by the total power density of the input AM15 spectrum. The fill factor (80.36%) can also be computed by dividing the maximum power by the product of I_{sc} and V_{oc}.

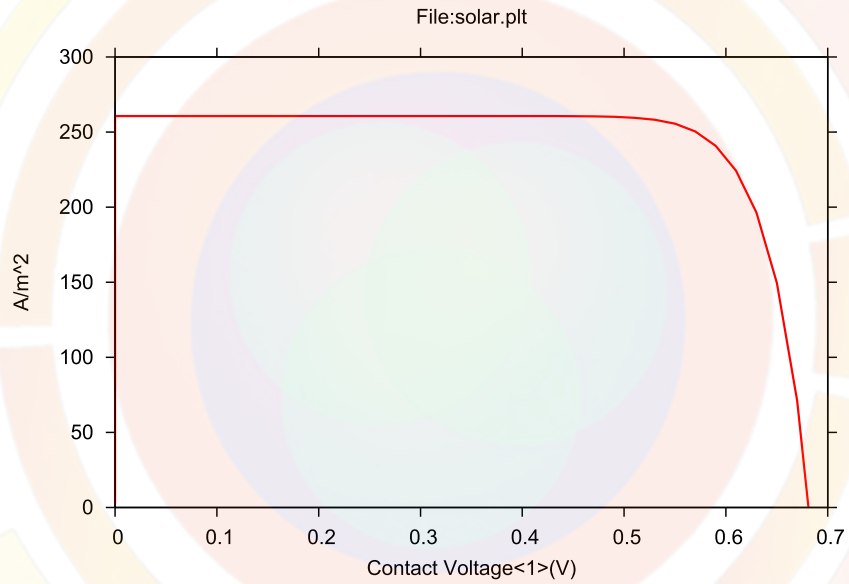


Figure 19.14: Solar cell I-V curve with current density units

PARS OPTICS GROUP

Chapter 20

LASTIP EXAMPLES

20.1 Introduction

This chapter will describe a few selected examples of LASTIP simulations[1]. The relevant input files can be found in your local installation directory (default is c:\crosslig\lastip_examples): the section name corresponds to the examples's location on your hard drive. These examples have been selected to teach the basic operation of the software, explain key concepts essential to device simulation, showcase some popular applications/devices and preemptively answer common user questions.

Note that the examples from this chapter are only a small subset of the included tutorial files. There are a number of other examples that are supported through accompanying README files and comments in the simulation files. Crosslight support staff will also be happy to explain these examples in more detail if you need further assistance.

It is relevant to note that all examples are subject to change as the software is updated. Between each release, there are often bug fixes, material macro changes, new models and other improvements which can affect the results. In most cases, these should result in negligible changes in the results which do not alter the tutorial value of these examples. If any major inconsistencies are found, please report them to Crosslight support staff. Most examples listed here were last updated as part of the 2009 manual release. Whenever possible, we will note examples which have been updated since then.

Throughout this chapter, we will concentrate on the specific simulation statements and physical models that are needed for a particular device. Particular attention will also be given to model parameters that may affect the convergence or reliability of the simulation.

[1] As of v.2016, the functionality of LASTIP has been merged with PICS3D. All files processed using LASTIP in this chapter should now use PICS3D instead.

It is also assumed that the reader is familiar with the basic structure of the input files so some of the basic setup steps for will be omitted for the sake of brevity. If you have not already done so, we strongly urge you to read through Chapter 3. Of particular interest will be the sections dealing with the use of the command-line setup tools as well as their integration into SimuCenter. If you are working using GUI tools such as LayerBuilder, additional online help is available through SimuCenter.

In some cases, the setup tools will only provide the basic framework of an input file and some direct editing may be required to complete the setup. If this is the case, new users are encouraged to use the built-in Wizard option in SimuLASTIP: this will present all the available statements and parameters. New statements may be created by right-clicking on an empty line and selecting the Wizard option: existing statements can be modified in the same way.

Note that users of other Crosslight tools besides LASTIP may also benefit from this chapter. In many cases, the physical models described here are relevant to other devices that are not within the scope this software. While the input files themselves cannot always be reused, most of the syntax will carry over to other Crosslight tools.

20.2 A_tutorial\1D_laser

This example is a simple GaAs/AlGaAs laser with a single quantum well and a GRINSCH structure. It also shows how to define layers with composition gradings.

Layer Structure

The layers are defined as follows:

1.35 μm	$Al_{1-x}Ga_xAs$	x=0.71	n=1e24
0.1462 μm	$Al_{1-x}Ga_xAs$	x=0.71 $\rightarrow$ x=0.33	n=1e23
0.0076 μm	GaAs		undoped
0.1462 μm	$Al_{1-x}Ga_xAs$	x=0.33 $\rightarrow$ x=0.71	p=1e23
1.35 μm	$Al_{1-x}Ga_xAs$	x=0.71	p=1e24

We use a 1D model for simplicity so the width of the device is not relevant. However, some value must be provided for program to function so we simply use 1 μm. Experienced users of the software may notice that earlier versions of this example did not have doping in the graded layers. However we have found that by adding this, it smooths out the carrier injection and eliminates a kink in the I-V curve. As a result, the threshold current is lowered and the resonance frequency is increased.

The actual layer file corresponding to this structure can be built using the LayerBuilder program or the command-line setuplayer program as discussed in Chapter 3. In LayerBuilder, the result looks like Fig. 20.1.

With some modifications, the final layer file is as follows:

```
$file:gaas10.layer
begin_layer
column column_num=1 w=1. mesh_num=2  r=1.
bottom_contact column_num=1 from=0 to=1 contact_num=1 &&
  contact_type=ohmic
$
layer_mater column_num=1 macro_name=algaas &&
  var_symbol1=x var1=0.71
layer d=1.35 n=20 r=0.9 n_doping1=1e24
$
layer_mater column_num=1 macro_name=algaas &&
  var_symbol1=x grade_var=1 &&
  grade_from=0.71 grade_to=0.33
layer d=0.1462 n=10 r=0.9 n_doping1=1e23
$
```

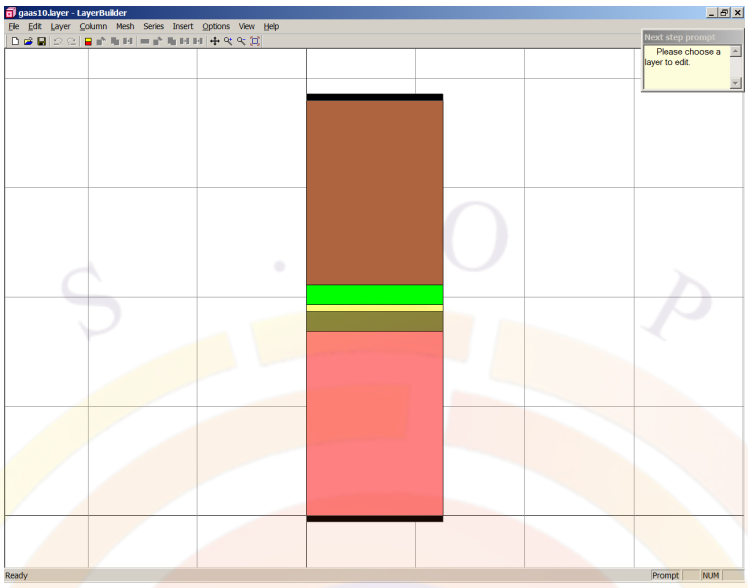


Figure 20.1: Device structure

```
layer_mater column_num=1 macro_name=gaas &&
  active_macro=AlGaAs/AlGaAs &&
  avar_symbol1=xw avar1=0.0 &&
  avar_symbol2=xb avar2=0.33
layer d=0.0076 n=5 r=1
$
layer_mater column_num=1 macro_name=algaas &&
  var_symbol1=x grade_var=1 &&
  grade_from=0.33 grade_to=0.71
layer d=0.1462 n=10 r=+1.1 p_doping1=1e23
$
layer_mater column_num=1 macro_name=algaas &&
  var_symbol1=x var1=0.71
layer d=1.35 n= 20 r=+1.1 p_doping1=1e24
$
top_contact column_num=1 from=0 to=1 contact_num=2 &&
  contact_type=ohmic
end_layer
```

For the mesh point allocation, we must correctly sample the optical mode shape and the barrier potentials in the device. By controlling the mesh point ratios with the **r** parameter in **layer**, we can meet these two goals simultaneously. The **layer** statement also controls the linear grading of the doping that accompanies the composition grading in this example.

Simulation Setup

A basic .sol file to run the simulation can be generated with the command-line setuplastip program as discussed in Chapter 3. With a few modifications, we get the following:

```
$file:gaas10.sol
$***********
begin
load_mesh mesh_inf=gaas10.msh
output sol_outf=gaas10.out
$ *************************
include file=gaas10.doping
include file=gaas10.mater
more_output ac_data=yes

direct_eigen
init_wave length=400 backg_loss=500. &&
  boundary_type=(2 2 1 1 )  init_wavel=0.83 mirror_ref=0.32 &&
  wavel_range=(0.80, 0.85)

newton_par damping_step=5. max_iter=100 print_flag=3

$ scan_num=0

equilibrium
$stop

newton_par damping_step=1. print_flag=3

$ scan_num=1
scan var=voltage_1 value_to=-2 &&
  init_step=1e-3 max_step=0.1 &&
  auto_finish=current_1 auto_until=1e-3 auto_within=5e-4

$ scan_num=2
scan var=current_1 value_to=20.  print_step=2. &&
  init_step=1.e-3 max_step=0.2
$
end
```

For LASTIP simulations, one of the key statements in the .sol file is **init_wave**. This statement defines the length of the laser, mirror coefficients and background

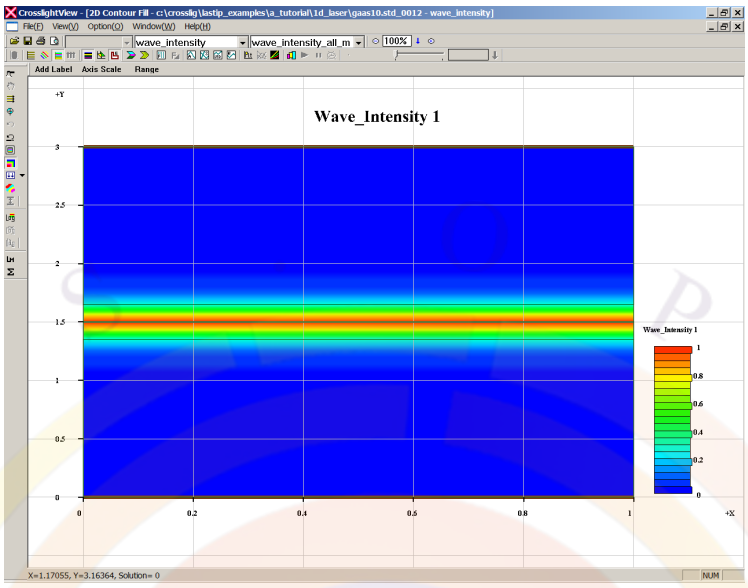


Figure 20.2: Optical mode profile

scattering losses. It also initializes the mode solver by setting the boundary conditions as well the wavelength range for the mode search. The mode solver itself is invoked with **direct_eigen**. Since we will want to do AC analysis on this device, we also add the **more_output** statement so the necessary data is saved during the simulation.

Once these quantities have been defined, we can apply bias. As explained in Section 4.1, we cannot apply a current bias right away so after getting the equilibrium state, a small voltage bias is applied in order to get a small amount of current flowing. Here, we show how to use the **auto_finish** parameter of the **scan** statement to automate the procedure but it is not required. A value of 80-90% of the built-in bias as the ending point of the voltage scan would do just as well:

$$0.8 * \frac{1.24\text{eV}}{0.83\mu m} \approx 1.2\text{V}$$

Post-Processing

To plot the simulation results, several options are available. One of them is CrosslightView, which can be accessed from SimuLastip by right-clicking any .std output file. The results are shown in Fig 20.2.

Some post-processing steps such as AC analysis still require the older .plt system which produces plots in PostScript (.ps) format. We use the following .plt file in this example:

```
$file:gaas10.plt
$ **************
begin_pstprc
plot_data plot_device=postscript

$ plot structure data at equilibrium
get_data  main_input=gaas10.sol sol_inf=gaas10.out &&
  xy_data=(1 1)

plot_1d variable=wave_intensity from=(0.5 0.0) to=(0.5, 3.0)
plot_1d variable=band from=(0.5 1.3) to=(0.5, 1.7)

$ plot structure data at max. data set (max. current injection)
get_data  main_input=gaas10.sol sol_inf=gaas10.out &&
  xy_data=(12 12)

plot_1d variable=band from=(0.5 1.3) to=(0.5, 1.7)

$ plot DC/CW bias dependent quantities.
get_data  main_input=gaas10.sol sol_inf=gaas10.out &&
  scan_data=(1 12)
plot_scan scan_var=voltage_1 variable=current_1 scan_num=2

$ para_extract extracts data from the next plotting statetment
$ (L-I curve in this case). We fit the data above 1 mW to
$ a straight line in order to find the threshold current.
$ We also find the drive current at 8 mW output.
para_extract type=fit_line fit_vert_from=1 &&
  hori_intercept=8.

$ laser_current is a special LASTIP variable where current density (A/m)
$ is converted to current (mA) by multiplying by the device length
$ and scaling for symmetry (factor of 2).
$ laser_power is the total output power from both facets and is also
$ scaled for symmetry.
plot_scan scan_var=laser_current_1 variable=laser_power scan_num=2

$ Add marker to subsequent plots
modify_plot show_data_points=yes

$ AC analysis (vs. freq.) at max. current
get_data  main_input=gaas10.sol sol_inf=gaas10.out &&
  xy_data=(12 12)
```

```
ac_voltage log_freq1=6.  log_freq2=10.3 &&
  contact_num=1 freq_point=40 log_scan=no modulate_gain=yes

$ AM response
plot_ac_laser
plot_ac_modal_gain

$ Capacitance derived from AC current on electrode #1
plot_ac_curr variable=capacitance_1

$ AC analysis vs. bias (must start above equilibrium) at 10 GHz
get_data  main_input=gaas10.sol sol_inf=gaas10.out &&
  scan_data=(2 12)
ac_voltage log_freq1=10.  log_freq2=10.3 &&
  contact_num=1 freq_point=2 versus_bias=yes scan_num=2

$ AC data is regarded as xydata and this command would
$ enable plotting AC data versus scan variable
set_xydata_for_scan scan_var=laser_current_1 scan_num=2

plot_ac_laser
plot_ac_curr variable=capacitance_1


end_pstprc
```

Structural data is plotted with **plot_1d** at equilibrium and at maximum bias. We note that when plotting the wave intensity in the .plt system, the confinement factor of the mode is printed in the title. This information can also be found by examining the .sol.msg file.

The I-V and L-I curves are obtained with the **plot_scan** statement. Note the use of scan line numbers which matches the comments of the .sol file: this is not required but it avoids the direct use of the scan_data numbers in **get_data**. The special variables **laser_power** and **laser_current_i** are unique to LASTIP and are used to convert the units of the 2D simulation to something more convenient to the user. However, we note that since this is a 1D simulation, the device has no real impact on the device simulation aside from scaling the current; instead, dividing the **current_i** value (in A/m) by the device width to give a current density value in A/m^2 may be a more useful representation of pumping in a broad-area device.

AC analysis is done by invoking the **ac_voltage** statement. This defines the contact number that is to be modulated as well as the frequency range of the analysis. AC analysis can be done either as a function of the modulation frequency or as a function

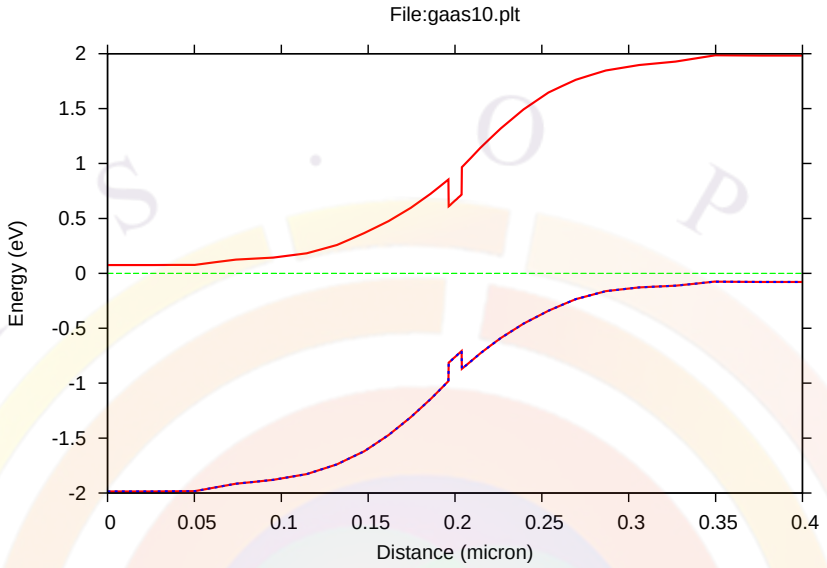


Figure 20.3: Band diagram at equilibrium

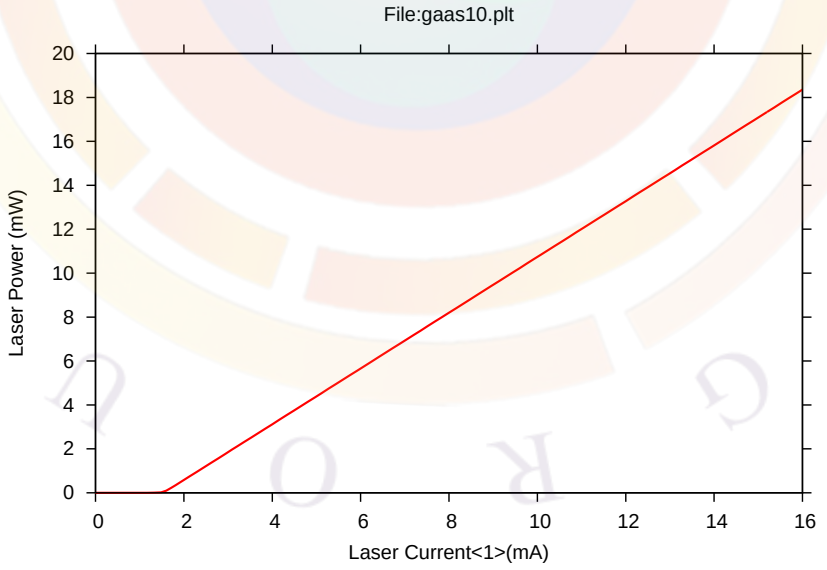


Figure 20.4: LI curve of laser

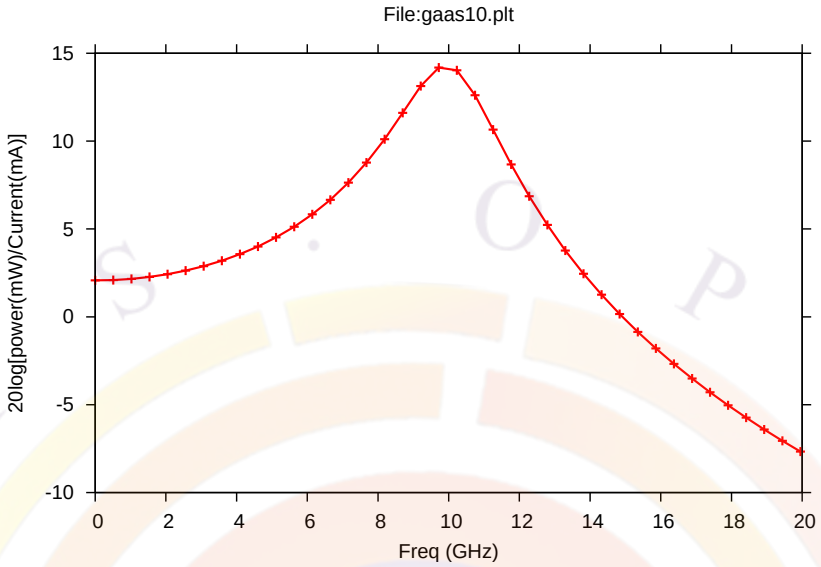


Figure 20.5: AM response of laser at maximum bias

of bias. The two are mutually exclusive so **ac_voltage** must be re-issued if both sets of data are of interest. For further examples of AC analysis, consult Sec. 19.2.

20.3 A__tutorial\1D__therm

This example explains how to include self-heating and thermal effects in a simulation. Proper thermal boundary conditions and the behavior of the substrate are key points to consider.

To model the substrate, some users prefer to abstract it as a fixed thermal resistance boundary condition (see Chapter 11). Although this approach is certainly viable in this simplified example, it could neglect important physical effects in other cases. Therefore, we will also use this example to show how to explicitly include substrate layers.

Layer Structure

We base this example on Sec. 20.2 so the substrate layer will be GaAs; it will be highly n-doped in order to form an ohmic contact. Adding this layer can be done easily in the LayerBuilder GUI or by editing the layer file and adding the appropriate **layer** and **layer__mater** statements. The resulting file is shown below:

```
$file:gaas10.layer
begin_layer
column column_num=1 w=1. mesh_num=2  r=1.
bottom_contact column_num=1 from=0 to=1 contact_num=1 &&
  contact_type=ohmic

layer_mater column_num=1 macro_name=gaas solve_wave=no
layer d=100 n=30 r=0.9 n_doping1=1e24

layer_mater column_num=1 macro_name=algaas &&
  var_symbol1=x grade_var=1 &&
  grade_from=0.0 grade_to=0.71 solve_wave=no
layer d=1.0 n=20 r=0.9 n_doping1=1e24

$
layer_mater column_num=1 macro_name=algaas &&
  var_symbol1=x var1=0.71
layer d=1.35 n=20 r=0.9 n_doping1=1e24
$
layer_mater column_num=1 macro_name=algaas &&
  var_symbol1=x grade_var=1 &&
  grade_from=0.71 grade_to=0.33
layer d=0.1462 n=10 r=0.9 n_doping1=1e23
$
```

```
layer_mater column_num=1 macro_name=gaas &&
  active_macro=AlGaAs/AlGaAs &&
  avar_symbol1=xw avar1=0.0 &&
  avar_symbol2=xb avar2=0.33
layer d=0.0076 n=5 r=1
$
layer_mater column_num=1 macro_name=algaas &&
  var_symbol1=x grade_var=1 &&
  grade_from=0.33 grade_to=0.71
layer d=0.1462 n=10 r=+1.1 p_doping1=1e23
$
layer_mater column_num=1 macro_name=algaas &&
  var_symbol1=x var1=0.71
layer d=1.35 n= 20 r=+1.1 p_doping1=1e24
$
top_contact column_num=1 from=0 to=1 contact_num=2 &&
  contact_type=ohmic
end_layer
```

However, because the substrate layer is GaAs, it creates a few additional problems that might not occur in other material systems (such as devices grown on InP). The first issue we have to deal with is that in this structure, the layer interfacing with the substrate is also heavily doped and has a very different composition than GaAs. This forms an abrupt heterojunction (c.f. Sec. 5.2.3) which will block current and cause convergence problems. There are a few ways of resolving this issue:

- Alter the doping profile of the original device and user a lighter doping in the bottom $Al_{0.71}Ga_{0.29}As$ layer. Since the Fermi level is not pinned on both sides, it avoids forming a barrier.
- Add a thin layer with graded composition: the pinned Fermi level can safely follow the composition grading and a thin layer should not disrupt the simulation too much.
- Introduce quantum tunneling to the simulation so the current can flow past this abrupt junction.

For tutorial purposes, we have chosen the second approach.

The second problem is the behavior of the mode solver in this new device. Since the refractive index of AlGaAs is lower than that of GaAs, adding the substrate means the structure now supports additional modes that peak in this region. These modes have very poor confinement in the active region and do not compete with the lasing mode but they have a higher effective index. Since the mode solver sorts modes

in order of decreasing index, we would have to use an extremely high number of optical modes to ensure that the lasing mode is included in the calculations. This is impractical so we use another approach.

Since we know from Sec. 20.2 that the desired mode is well-confined in the active region and does not leak, we exclude the high-index substrate layers from the mode solver altogether. This is done by using the parameter **solve_wave=no** in the relevant **layer_mater** statement. When the .layer file is processed, this will modify the **wave_boundary** statement in the .mater file and restrict the mode calculations to a range that does not include the offending layers.

Simulation Setup

The .sol file is virtually unchanged from Sec. 20.2. We will concentrate here only on the statements relevant for a thermal simulation:

```
$file:gaas10.sol
$***********
begin
load_mesh mesh_inf=gaas10.msh
output sol_outf=gaas10.out
$ *************************
include file=gaas10.doping
include file=gaas10.mater
more_output ac_data=yes

direct_eigen
init_wave length=400 backg_loss=500. &&
  boundary_type=(2 2 1 1 )  init_wavel=0.83 mirror_ref=0.32 &&
  wavel_range=(0.80, 0.85)

$ Heat flow modeling
temperature temp=300
heat_flow fit_range=500
contact num=1 thermal_type=3 thermal_cond=10
contact num=2 thermal_type=2 heat_flow=0

$ Free carrier losses
passive_carr_loss ncarr_loss=5e-22 pcarr_loss=5e-22 mater=1
passive_carr_loss ncarr_loss=5e-22 pcarr_loss=5e-22 mater=2
passive_carr_loss ncarr_loss=5e-22 pcarr_loss=5e-22 mater=3
passive_carr_loss ncarr_loss=5e-22 pcarr_loss=5e-22 mater=5
set_active_reg ncarr_loss=5e-22 pcarr_loss=5e-22
```

```
newton_par damping_step=5. max_iter=100 print_flag=3

$ scan_num=0
equilibrium
$stop

$ scan_num=1
scan var=voltage_1 value_to=-2 init_step=1e-3 max_step=0.1 &&
  auto_finish=current_1 auto_until=0.1 auto_condition=above

newton_par damping_step=1.0 max_iter=100 opt_iter=50 stop_iter=10

$ print_step can sometimes fail during thermal simulations
$ so use multiple scan statements to print intermediate data

$ scan_num=2
scan var=current_1 value_to=10. &&
  init_step=1.e-2 min_step=1e-8 max_step=0.1

$ scan_num=3
scan var=current_1 value_to=20. &&
  init_step=1.e-2 min_step=1e-8 max_step=0.1

$ scan_num=4
scan var=current_1 value_to=30. &&
  init_step=1.e-2 min_step=1e-8 max_step=0.1

$ scan_num=5
scan var=current_1 value_to=40. &&
  init_step=1.e-2 min_step=1e-8 max_step=0.1

$ scan_num=6
scan var=current_1 value_to=50. &&
  init_step=1.e-2 min_step=1e-8 max_step=0.1


end
```

The first relevant item is the **temperature** statement: this is used to define the temperature at which the equilibrium calculations are done. In an isothermal simulation, this is also the temperature for the rest of the simulation. This statement can safely be omitted in many cases: a value of 300K will be assumed. However, certain

designs call for active cooling which means that before the device is turned on, it will have reached a different thermal equilibrium. If that is the case, this statement must be used to define the new equilibrium point correctly.

The **heat_flow** statement activates the self-heating model from Chapter 11. This model is decoupled from the main electrical solver so after each electrical step, the temperature distribution and material parameters are updated. As a result, the user must ensure the electrical step is always sufficiently small. Not only will this increase the self-consistency of the simulation but it helps with convergence: a sudden temperature change can invalidate the initial guess used for the next electrical step.

Thermal boundaries are set by two different methods. For contact regions, it is part of the **contact** statement. Usually, this statement is generated automatically when processing the .layer file but this sets the thermal boundaries to a default setting: a fixed temperature of 300 K. In all thermal simulations, the user is advised to re-issue the **contact** statements in order to provide the desired thermal boundary. For non-contact regions, the thermal boundary conditions may be defined with the **thermal_inter** statement.

For both these situations, there are several types of boundary conditions available (see Chapter 11). For the bottom contact, we use the more realistic model (type 3) with a fixed thermal conductivity. For the top contact, we wish to present a "worst-case" scenario in which no heat can escape through the top layer: this is done by using an interface of type 2 and explicitly setting the heat flow to zero. We can expect significant heat build-up in the active region as a result of this choice.

If a particular design has different thermal boundaries (e.g. substrate is cooled at 77K but top layers are exposed to air at 300K), then there is an inconsistency in the **equilibrium** calculations. By definition, this calculation assumes the absence of either current or heat flow so if there is an externally applied thermal gradient, the the equilibrium state will need to be revised. The **nstep** parameter can be used to re-run the equilibrium state multiple times so that it stabilizes to take this gradient into account. Note that doing this abuses the concept of equilibrium but we only need require an functioning initial guess to the **scan** statements.

The next element of thermal laser modeling to discuss is not strictly a thermal term but we have found that it is important in modeling the expected power roll-off in lasers at high temperature. This effect is an optical loss term commonly referred to as "free carrier losses": these are carrier-dependent and material-specific loss coefficients that are added to the background loss coefficient defined in the **init_wave** statement.

For passive regions, the free carrier losses may be defined through the **passive_carr_loss** statement in the .sol file. It is also possible to use **elec_carr_loss** and **hole_carr_loss** in the macro file. For active regions, these losses are defined in the .sol file with **set_active_reg**.

The user may also note that multiple scan statements are used in the bias current scan: this is used to generate intermediate data sets for post-processing and plotting purposes. Normally, we could use the **print_step** parameter in the **scan** statement to accomplish this. However, since the thermal and electrical problems are not full coupled in the Newton solver, it is not always possible to exactly match a requested current value. Since **print_step** is designed to always print at specific values, the conflict between the two can sometimes cause problems. By using multiple scan statements, we can generate intermediate data and although the solver may complain that the requested current value at the end of each scan cannot be matched exactly, it will be close enough for most purposes.

Post-Processing

The post-processing is done with the following .plt file:

```
$file:gaas10.plt
$ **************
begin_pstprc
plot_data plot_device=postscript

$ plot structure data at equilibrium
modify_plot show_data_points=no

get_data  main_input=gaas10.sol sol_inf=gaas10.out &&
  xy_data=(1 1)

plot_1d variable=real_index from=(0.5 99.0) to=(0.5, 104.0)
plot_1d variable=wave_intensity from=(0.5 99.0) to=(0.5, 104.0)

plot_1d variable=band from=(0.5 99.0) to=(0.5, 104.0)

$ plot structure data at max. data set (max. current injection)

get_data  main_input=gaas10.sol sol_inf=gaas10.out &&
  xy_data=(7 7)

plot_1d variable=band from=(0.5 99.0) to=(0.5, 104.0)
plot_1d variable=lattice_temp from=(0.5 0.0) to=(0.5, 104.0)


$ plot DC/CW bias dependent quantities.
get_data  main_input=gaas10.sol sol_inf=gaas10.out &&
```

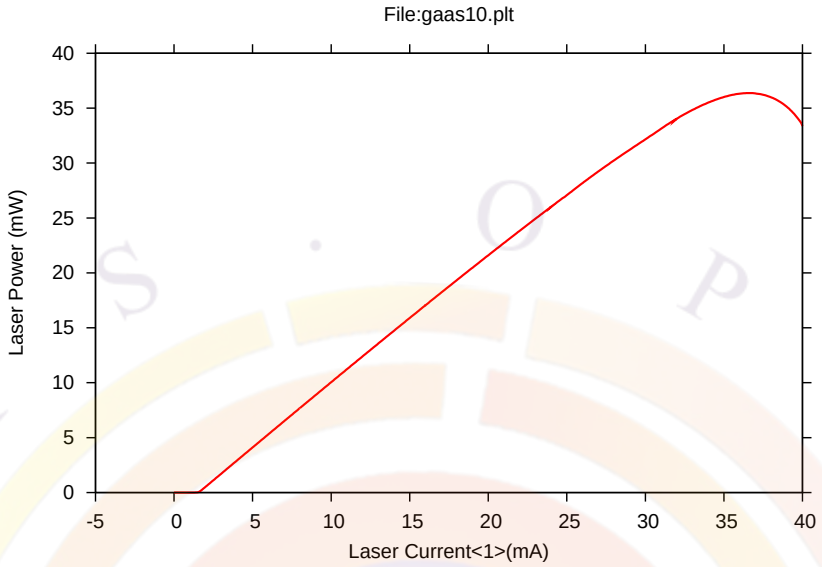


Figure 20.6: L-I curve of laser with thermal effects included

```
  scan_data=(1 7)
plot_scan scan_var=voltage_1 variable=current_1

plot_scan scan_var=laser_current_1 variable=laser_power
plot_scan scan_var=laser_current_1 variable=temp_max

end_pstprc
```

The key points to observe are the roll-over in the L-I curve in Fig. 20.6 and the accompanying increase in maximum temperature in Fig. 20.7.

Application notes for thermal laser simulations

It may be helpful to note that many of the default Crosslight macros do not include the full temperature dependence of the refractive index. While the index change calculations (see. Sec. 8.4) are affected by the temperature, the equilibrium contribution to the refractive index is controlled solely by the **real_index** statement in the macro file.

If this effect is deemed important in a particular design, the user is advised to check the various macros used in the simulation. To include temperature dependence, the **real_index** statement in the macro must be a function of the reserved keyword "temper". If not, the user may override the macro in the .sol file or use a custom macro file.

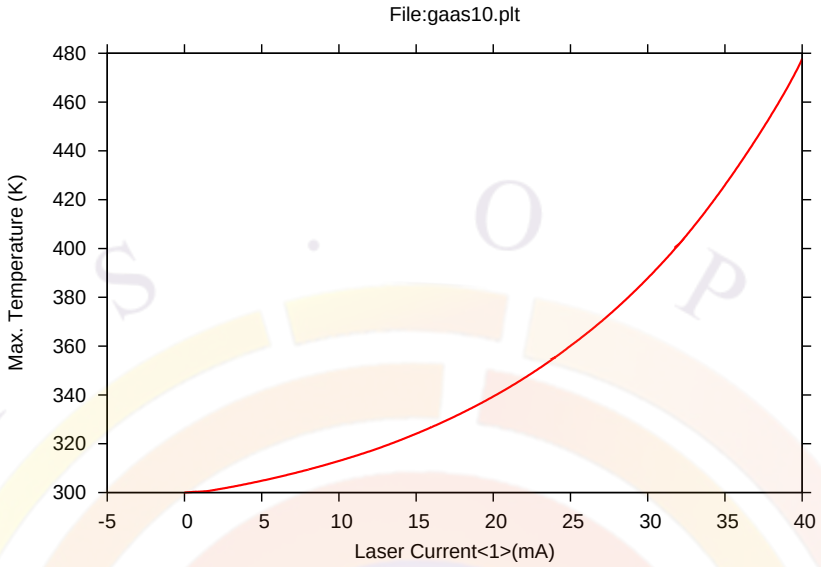


Figure 20.7: Maximum temperature increase vs. bias

Since high-power/thermal laser applications cover a wide variety of wavelength ranges, it is also recommended to check if the macro index values are consistent with the lasing wavelength. In particular, the default Crosslight macros for GaAs-based compounds are set up for the 830 nm range. However, this material is also used in 980 nm applications so the index may have to de redefined to cover a different wavelength range.

For transient analysis in a thermal simulation, the user must be sure to turn on **thm_transient** in the **heat_flow** statement. The user must also specify values for the specific heat (**spec_heat**) and density (**mass_density** of the materials since most of the default macros omit these values.

20.4 A_tutorial\basic_ridge

This example is a basic ridge waveguide laser. It introduces how to work with the mode solver and deal with multiple lateral modes.

Layer Structure

This example is a simple modification of the 1D laser from Sec. 20.2. This time, we want to model a ridge waveguide laser so we will need a second column. Note that by doing this, we are using the symmetry of the device to our advantage in order to save on mesh points and speed up the simulation. However, this means that the user should scale the current and power output during the post-processing stage: unlike older versions of the software LASTIP no longer automatically scales its output by assuming symmetry exists. In current versions of the software, "what you see is what you get" as far as symmetry is concerned.

The simplest way to get our second column is to take the layer file from the 1D laser case and open it in LayerBuilder; it is then possible to copy/paste the column. Some modifications must then be made, either by using LayerBuilder or editing the layer file:

- Define the width of the columns; unlike the 1D case, this is important
- Assign mesh points to the column statements in order to sample the lateral variation of the device
- Split the top layer into two and replace the material in the second column in order to represent the etch depth for the ridge
- Remove the top contact from the second column

In this example, we are using an etch depth of 1 μm. We also decide to use an arbitrary ridge width of 1 μm: since we are using symmetry, this means the first column will be 0.5 μm. The second column needs to be wide enough to not artificially confine the optical mode (because of boundary conditions) without wasting too many mesh points: again, we use a value of 1 μm.

The appearance of the final design in LayerBuilder is shown in Fig. 20.8. The corresponding layer file is shown below:

```
$file:gaas20.layer
begin_layer

column column_num=1 w=0.5 mesh_num=5  r=-1.1
```

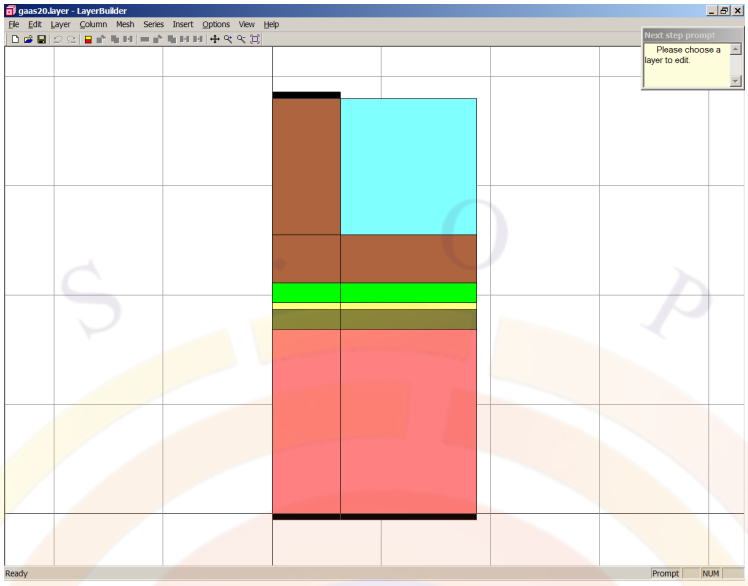


Figure 20.8: Device structure

```
column column_num=2 w=1.  mesh_num=5  r=1.1
bottom_contact column_num=1 from=0 to=1 contact_num=1 contact_type=ohmic
bottom_contact column_num=2 from=0 to=1 contact_num=1 contact_type=ohmic
$
layer_mater column_num=1 macro_name=algaas var_symbol1=x var1=0.71 &&
   n_doping=1e24
layer_mater macro_name=algaas column_num=2 var_symbol1=x var1=0.71 &&
   n_doping=1e24
layer  d=1.35  n=20  r=-1.1
$
layer_mater column_num=1 macro_name=algaas var_symbol1=x grade_var=1 &&
   grade_from=0.71 grade_to=0.33 n_doping=1e23
layer_mater macro_name=algaas column_num=2 var_symbol1=x grade_var=1 &&
   grade_from=0.71 grade_to=0.33 n_doping=1e23
layer  d=0.1462  n=10  r=-1.1
$
layer_mater column_num=1 macro_name=gaas active_macro=AlGaAs/AlGaAs &&
   avar_symbol1=xw avar1=0.0 avar_symbol2=xb avar2=0.33
layer_mater macro_name=gaas column_num=2 active_macro=AlGaAs/AlGaAs &&
   avar_symbol1=xw avar1=0 avar_symbol2=xb avar2=0.33
layer d=0.0076 n=5 r=1
$
layer_mater column_num=1 macro_name=algaas var_symbol1=x grade_var=1 &&
   grade_from=0.33 grade_to=0.71 p_doping=1e23
layer_mater macro_name=algaas column_num=2 var_symbol1=x grade_var=1 &&
```

```
   grade_from=0.33 grade_to=0.71 p_doping=1e23
layer  d=0.1462  n=10  r=-1.1
$
layer_mater column_num=1 macro_name=algaas var_symbol1=x var1=0.71 &&
   p_doping=1e24
layer_mater macro_name=algaas column_num=2 var_symbol1=x var1=0.71 &&
   p_doping=1e24
layer  d=0.35  n=5  r=-1.1
$
layer_mater column_num=1 macro_name=algaas var_symbol1=x var1=0.71 &&
   p_doping=1e24
layer_mater macro_name=air column_num=2
layer  d=1.0  n=15  r=-1.1
$
top_contact column_num=1 from=0 to=1 contact_num=2 contact_type=ohmic
end_layer
```

Note that we use the "air" macro in the etched-away region instead of "void". The difference is that "void" will define a region without mesh points: this means it will not be used in the mode solver and the boundary conditions will be applied at the ridge wall. On the other hand, "air" and "vacuum" macros will define an isolating region with the proper refractive index: this prevents the mode solver from artificially confining the optical mode under the ridge.

Simulation Setup

The .sol file needed to run this simulation is fairly similar to the one from from Section 20.2. However, please note the difference in boundary conditions and the use of the **multimode** statement:

```
$file:gaas20.sol
$**********
begin
load_mesh mesh_inf=gaas20.msh
output sol_outf=gaas20.out
$ *************************
include file=gaas20.doping
include file=gaas20.mater
more_output ac_data=yes

direct_eigen
init_wave length=400 backg_loss=500. &&
```

```
  boundary_type=(2 1 1 1 )  init_wavel=0.83 mirror_ref=0.32 &&
  wavel_range=(0.80, 0.85)

multimode mode_num=6 boundary_type1=(2 1 1 1) boundary_type2=(1 1 1 1)

newton_par damping_step=5. max_iter=100 print_flag=3

$ scan_num=0
equilibrium
$stop

newton_par damping_step=1. print_flag=3

$ scan_num=1
scan var=voltage_1 value_to=-2 &&
  init_step=1e-3 max_step=0.1 &&
  auto_finish=current_1 auto_until=1e-3 auto_within=5e-4

$ scan_num=2
scan var=current_1 value_to=20.  print_step=2. &&
  init_step=1.e-3 max_step=0.2
$
end
```

The two types of boundary conditions are needed to obtain both odd and even modes in this symmetric structure. Since we use two boundary conditions, half of the modes will have the symmetry of **boundary_type1** and the other half will have that of **boundary_type2**. Within each set, modes are sorted in order of decreasing modal index. In this example, the highest order even mode is number #1 and the highest order odd mode is #4; they are respectively shown in Fig. 20.9 and Fig. 20.10.

Please keep in mind that depending on the structure, it is not necessarily the fundamental mode or the mode with the largest confinement factor that lases: the only thing that matters is the modal gain. For example, the .sol.msg output file for this structure reports the following information at maximum bias:

```
 ----Optical data (in 1/m) for segment#          1
 Mode  Modal_gain  Conf._factor Int._loss
   1  0.2848E+04  0.2953E-01  0.4852E+03
   2  0.2843E+04  0.2959E-01  0.4852E+03
   3  0.2843E+04  0.2961E-01  0.4852E+03
   4  0.2840E+04  0.2965E-01  0.4852E+03
   5  0.2845E+04  0.2961E-01  0.4852E+03
   6  0.2840E+04  0.2961E-01  0.4852E+03
```

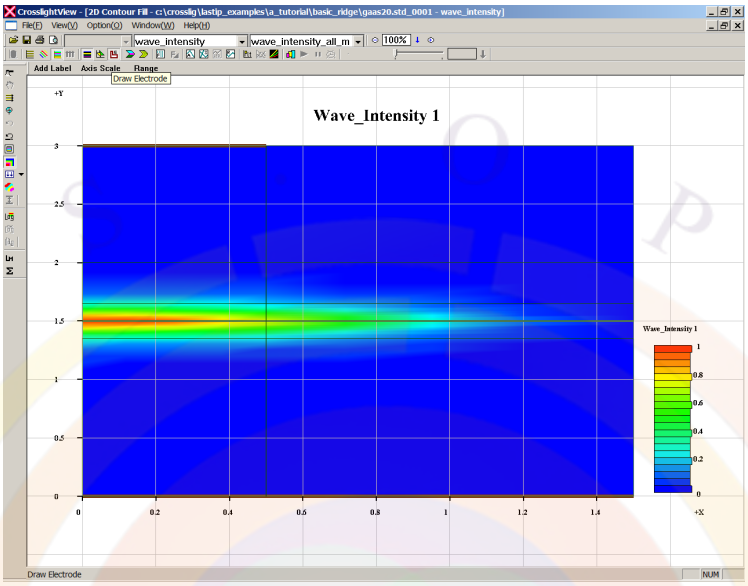


Figure 20.9: Mode # 1

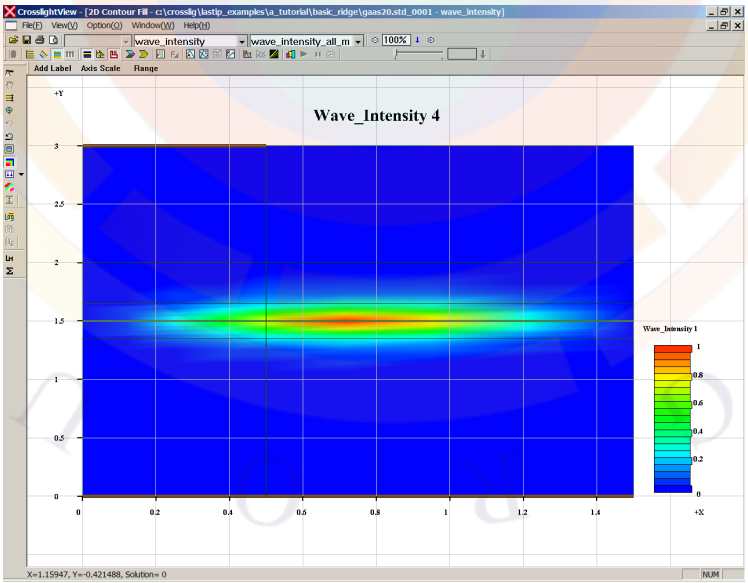


Figure 20.10: Mode # 4

Even though mode #1 has the lowest confinement factor, it has the highest modal gain since it does not peak in the un-pumped region that extends beyond the ridge. In other structures, the opposite can happen and a higher-order mode may lase instead of the fundamental mode.

Post-Processing

The output file above shows that the modal gain values are quite close, which would indicate significant mode competition. This is shown by plotting the total laser power from all modes as well as that of the individual lateral modes in the following .plt file:

```
$file:gaas20.plt
$ **************
begin_pstprc
plot_data plot_device=postscript

$ plot structure data at equilibrium
modify_plot show_data_points=no

get_data  main_input=gaas20.sol sol_inf=gaas20.out &&
  xy_data=(1 1)

plot_2d variable=wave_intensity mode_index=1
plot_2d variable=wave_intensity mode_index=2
plot_2d variable=wave_intensity mode_index=3
plot_2d variable=wave_intensity mode_index=4
plot_2d variable=wave_intensity mode_index=5
plot_2d variable=wave_intensity mode_index=6


plot_1d variable=band from=(0.5 1.3) to=(0.5, 1.7)

$ plot structure data at max. data set (max. current injection)

get_data  main_input=gaas20.sol sol_inf=gaas20.out &&
  xy_data=(12 12)

plot_1d variable=band from=(0.5 1.3) to=(0.5, 1.7)
plot_2d variable = total_curr

$ plot scan results
```

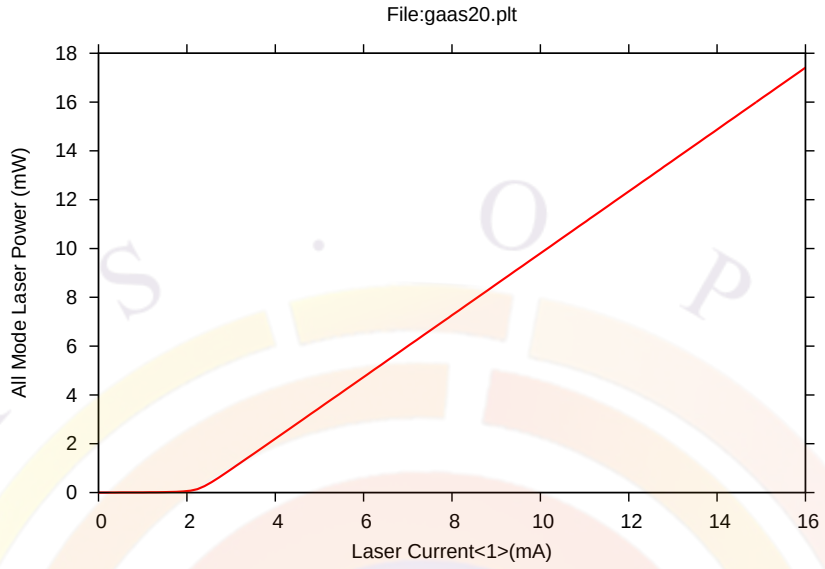


Figure 20.11: Total power from all modes

```
get_data  main_input=gaas20.sol sol_inf=gaas20.out &&
  scan_data=(1 12)
plot_scan scan_var=voltage_1 variable=current_1 scan_num=2

$ Compare the total power vs. power in individual modes
plot_scan scan_var=laser_current_1 variable=all_mode_power scan_num=2
plot_scan scan_var=laser_current_1 variable=laser_power scan_num=2 mode_index=1
plot_scan scan_var=laser_current_1 variable=laser_power scan_num=2 mode_index=2
plot_scan scan_var=laser_current_1 variable=laser_power scan_num=2 mode_index=3
plot_scan scan_var=laser_current_1 variable=laser_power scan_num=2 mode_index=4
plot_scan scan_var=laser_current_1 variable=laser_power scan_num=2 mode_index=5
plot_scan scan_var=laser_current_1 variable=laser_power scan_num=2 mode_index=6

end_pstprc
```

As shown in Figures 20.11 and 20.12, most of the power is carried in mode #1 so it is indeed the main lasing mode. However, it only accounts for about half of the output power which confirms that higher-order modes are significant in this structure.

We also note that the higher-order modes have significant intensity near the right side of the device: this may indicate that these modes are artificially confined by the boundary and are actually leaky/radiative modes. To account for this, the user may need to introduce PML boundary conditions as described in Sec. 12.6. However, this is beyond the scope of this example: users who wish to know more are encouraged to consult some of the other examples included in LASTIP.

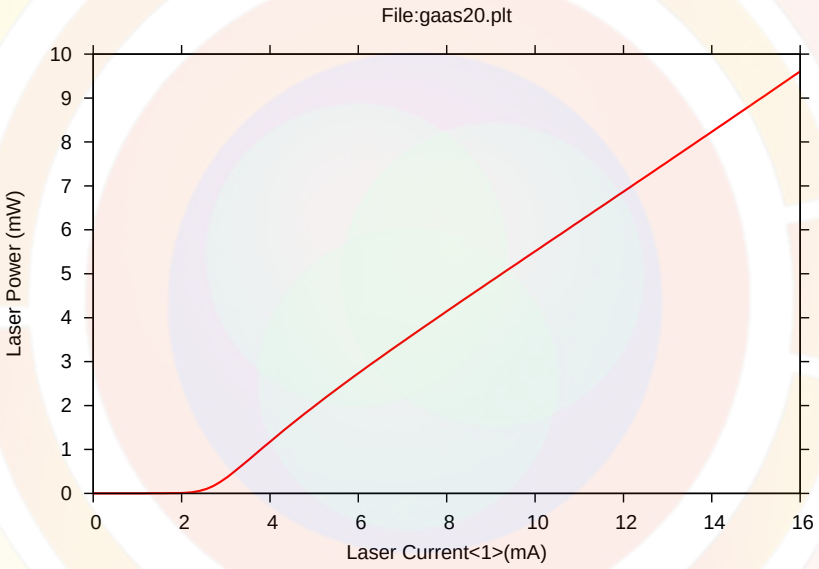


Figure 20.12: Power output from mode #1

Chapter 21

PICS3D EXAMPLES

21.1 Introduction

This chapter will describe a few selected examples of PICS3D simulations. The relevant input files can be found in your local installation directory (default is c:\crosslig\pics3d_examples): the section name corresponds to the examples's location on your hard drive. These examples have been selected to teach the basic operation of the software, explain key concepts essential to device simulation, showcase some popular applications/devices and preemptively answer common user questions.

Note that the examples from this chapter are only a small subset of the included tutorial files. There are a number of other examples that are supported through accompanying README files and comments in the simulation files. Crosslight support staff will also be happy to explain these examples in more detail if you need further assistance.

It is relevant to note that all examples are subject to change as the software is updated. Between each release, there are often bug fixes, material macro changes, new models and other improvements which can affect the results. In most cases, these should result in negligible changes in the results which do not alter the tutorial value of these examples. If any major inconsistencies are found, please report them to Crosslight support staff. Most examples listed here were last updated as part of the 2009 manual release. Whenever possible, we will note examples which have been updated since then.

Throughout this chapter, we will concentrate on the specific simulation statements and physical models that are needed for a particular device. Particular attention will also be given to model parameters that may affect the convergence or reliability of the simulation.

It is also assumed that the reader is familiar with the basic structure of the input files so some of the basic setup steps for will be omitted for the sake of brevity. If you have

not already done so, we strongly urge you to read through Chapter 3. Of particular interest will be the sections dealing with the use of the command-line setup tools as well as their integration into SimuCenter. If you are working using GUI tools such as LayerBuilder, additional online help is available through SimuCenter.

In some cases, the setup tools will only provide the basic framework of an input file and some direct editing may be required to complete the setup. If this is the case, new users are encouraged to use the built-in Wizard option in SimuPICS3D: this will present all the available statements and parameters. New statements may be created by right-clicking on an empty line and selecting the Wizard option: existing statements can be modified in the same way.

Note that users of other Crosslight tools besides PICS3D may also benefit from this chapter. In many cases, the physical models described here are relevant to other devices that are not within the scope this software. While the input files themselves cannot always be reused, most of the syntax will carry over to other Crosslight tools.

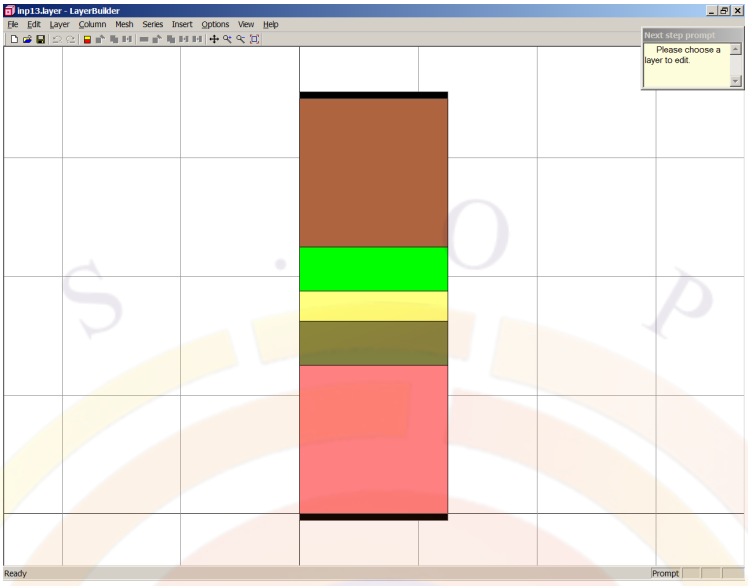


Figure 21.1: 2D cut of quarter-wave-shifted DFB laser

21.2 A_tutorial

This example is an index-coupled distributed feedback (DFB) laser with a quarter-wave phase shift in the middle. This kind of structure is often used to guarantee the device lases on a single longitudinal mode and provides some immunity against external cavity feedback. Users of previous versions of the software will note this example has been altered significantly.

Layer Structure

We start by defining a 2D cut of the device as shown in Fig. 21.1. The device is grown on an InP substrate and a bulk active region tuned to 1.3 μm is used for simplicity:

1.0 μm	InP		n=2e24
0.3 μm	$In_{1-x}Ga_xAs_yP_{1-y}$	x=lattice-matched, y=0.26	n=5e23
0.2 μm	$In_{1-x}Ga_xAs_yP_{1-y}$	x=lattice-matched, y=0.64	undoped
0.3 μm	$In_{1-x}Ga_xAs_yP_{1-y}$	x=lattice-matched, y=0.26	p=5e23
1.0 μm	InP		p=2e24

When defining this structure, care must be taken when selecting the macros: the material properties will be very different if the selected macro is lattice-matched to GaAs or InP. Some macros also allow for strain in which case the exact composition must be provided. The user is advised to consult the comments accompanying

each macro to select the right one: improper macro selection is a common cause of problems with simulations using the InGaAsP system.

An easy way to do this is to use the LayerBuilder GUI: it not only shows a full alphabetized list of available macros but also the comments in the macro header. It is also possible to consult the macros directly by opening the crosslig.mac and more.mac files in your installation directory with a text editor. However, we do not recommend making any changes to the default macro files: it is better to use a custom macro file if you wish to make changes to the material parameters.

For the sake of simplicity, the grating layer has been abstracted away in the above structure. Conceptually, we may assume that part of the top cladding layer is in fact an average material for the grating. We will define an effective grating as part of the longitudinal setup in the .sol file below. More accurate models to define grating layers are available, as will be shown in Sec. 21.4.

For an additional simplification, we neglect any changes in the lateral direction so the 2D cut is actually a 1D device. We have touched on this subject in a previous LASTIP example (Sec. 20.2) so we will omit the rest of the setup steps for the layer file. As can be seen below, it is quite simple:

```
$file:inp13.layer
begin_layer
column column_num=1 w=  0.100000E+01 mesh_num=2  r=1.
bottom_contact column_num=1 from=0 to=  0.100000E+01 &&
  contact_num=1 contact_type=ohmic
$
layer_mater  macro_name=inp column_num=1
layer d=1.0 n= 10 n_doping1=2e24 r=0.9
$
layer_mater  macro_name=ingaasp column_num=1 &&
  var_symbol1=y var1=0.26
layer d=0.3 n=9 n_doping1=5e23 r=0.9
$
layer_mater  macro_name=ingaasp column_num=1 &&
  var_symbol1=y var1=  0.64 &&
  active_macro=InGaAsP &&
  avar_symbol1=yw avar1=0.64
layer d=0.2 n= 12 r=-1.1
$
layer_mater  macro_name=ingaasp column_num=1 &&
  var_symbol1=y var1=0.26
layer d=0.3 n=9 p_doping1=5e23 r=1.1
$
layer_mater  macro_name=inp column_num=1
```

```
layer d=1.0 n= 10 p_doping1=2e24 r=0.9
$
top_contact column_num=1 from=0 to=  0.100000E+01 &&
  contact_num=2 contact_type=ohmic
end_layer
```

Simulation Setup

The following .sol file is used to run this example:

```
$file:inp13.sol
begin

3d_solution_method 3d_flow=yes z_connect=no

z_structure uniform_length=500.0 zplanes=7 zseg_num=1
load_mesh mesh_inf=inp13.msh
$
$ Include the gain file which contains material info.
include file=inp13.gain
include file=inp13.doping
$
output sol_outf=inp13.out
$
$ Set parameters for wave equation
init_wave backg_loss=  0.5000E+03 init_wavel=1.3 &&
  boundary_type=[2 2 1 1] wavel_range=[1.28 1.32]
direct_eigen

$ Set Newton parameters for equilibrium solution
newton_par damping_step=3. var_tol=1.e-8 res_tol=1.e-8 &&
  max_iter=100 opt_iter=15 stop_iter=50 print_flag=3
$
$ Solve equations at equilibrium
equilibrium
$
rtgain_phase density=1.25e24
$
$ You may stop here to examine the round trip gain
$
$stop
$
```

```
$ Set Newton parameters for solution with bias
newton_par damping_step=1. var_tol=1.e-4 res_tol=1.e-4 &&
  max_iter=50 opt_iter=15 stop_iter=9 print_flag=3

$ Ramp up bias voltage
scan var=voltage_1 value_to=-5.0 &&
  init_step=1e-3 max_step=0.1 &&
  auto_finish=current_1 auto_until=1e-3 auto_condition=above

$ Ramp up bias to near threshold as judged by RT-Gain
scan var=current_1 value_to=20e-3 &&
  init_step=1e-3 max_step=5e-3 print_step=10e-3 &&
  auto_finish=rtgain auto_until=0.8 auto_within=0.1

$ Activate 3D C-RTG solver
scan var=current_1 value_to=20e-3 &&
  init_step=1e-5 max_step=1e-4 print_step=10e-3 &&
  solve_rtg=yes
$
end
$ Longitudinal mode session:
begin_zsol

$ Set up grating at 1.3 um
longitudinal ref_wavel=1.3e-6 &&
  left_f_refl=0 right_f_refl=0

$ Quarter-wave phase shift: 90 degress=0.5*pi
section length=250e-6 kappa_real=2e3 &&
  sec_num=1 mesh_points=10 phase_shift=0.5
section length=250e-6 kappa_real=2e3 &&
  sec_num=2 mesh_points=10

$ longitudinal mode search parameters
mode_srch omega_xrange = 20 adjust_range = yes

end_zsol
```

In this section, we will skip over most of the details regarding the waveguide setup and concentrate on the PICS3D-specific aspects of the simulation. The most important point to understand is that there are two aspects to the 3D setup: electrical modeling and optical modeling. These will be described below.

3D Electrical Simulation

The first part of the 3D setup is the electrical simulation and it is identical to what would be used in an APSYS simulation. In both cases, the 3D model is activated with the **3d_solution_method** statement and one or more segments are defined. These segments combine multiple 2D mesh planes and define a 3D volume for the simulation. For a more detailed discussion on this topic, the user is referred to Sec. 6.3.

In this example, the layer file we defined above can represent a 2D cut at any z position of the device. Therefore, we have a single segment and only one **z_structure** statement defines the device for the electrical model. Note that since there is only a single segment, **zseg_num=1** is implicitly defined which simplifies the input file somewhat: we will see in Sec. 21.4 how to deal with multiple segments.

The longitudinal electrical mesh is determined by the number of mesh planes defined in **z_structure**. Since there is only a single segment, a minimum of two planes is needed to define a volume. However, a minimum of three planes is needed to observe a longitudinal variation of material parameters (i.e. longitudinal hole burning). More planes provide better sampling in the z direction but require more memory and processing power so there is always a compromise: we use seven mesh planes in this example.

3D Optical Simulation

The second part of the 3D simulation is the optical propagation: this is tied to the concept of the round-trip gain (RTG) discussed in Sec. 6.4. This longitudinal model is at the very core of PICS3D and involves several setup steps. These require a lengthy explanation so we will divide them into four main categories:

- Optical propagation model
- Longitudinal mode search
- Gain profile definition
- Bias setup

Optical Propagation Model

The optical propagation model is tied to the concept of "sections": these define the optical cavity in the same way that "segments" define the 3D geometry of the electrical simulation. In this case, we have a 500 μm laser but with a phase shift in the middle so we use two sections of equal length and define a phase shift at the end

of the first section. This phase value must be expressed as a multiple of π so $\frac{\lambda}{4}$ is equivalent to 0.5.

Within each section, the same propagation model is used: for edge-emitting devices such as this one, it is the coupled-wave transfer matrix of Sec. 16.5. The detuning term is based on the DFB grating's reference frequency so we must define it with the **longitudinal** statement. This statement also controls the boundary conditions used in the longitudinal propagation model (i.e. mirror reflectivities).

We also define a pure index grating of $\kappa = 2000$ in each section so that the total grating strength for the laser is $\kappa L = 1$. This method of defining the grating is the simplest and most convenient way available in PICS3D. More accurate models are available but they require an explicit definition of the grating composition in the layer file: we will show an example of this method in Sec. 21.4.

We note in passing that the two statements above replace some of the functionality provided by **init_wave** in LASTIP: if the user attempts to use it to define the cavity length or mirror strength, it will be ignored by the software. In PICS3D, the **init_wave** statement is only used to define the background loss of the optical propagation and to control the lateral mode solver.

Longitudinal Mode Search

The longitudinal mode search is controlled by the **mode_srch** statement. This statement defines the search range for the longitudinal modes. This can be done either explicitly with **wavel_xrange** or using a fixed-size window with **omega_xrange**. This window is usually centered on the reference wavelength of the grating but can be allowed to shift in order to center it on the strongest mode with **adjust_range=yes**.

As described in Chap. 16, longitudinal modes are found using the zeros of the Wronskian, which is equivalent to evaluating the RTG. However, evaluating the RTG while considering the photon coupling is tricky: the photon density can only be known by evaluating the RTG but evaluating the RTG requires knowledge of the photon density because of its impact on the longitudinal gain profile (i.e. spatial hole burning). To provide an initial guess, we must assume that the photon coupling is negligible at first and use a flat gain profile to find the longitudinal modes.

As part of its initialization, PICS3D requires a preview of the RTG and longitudinal mode search with the **rtgain_phase** statement. Since this statement only produces a preview of the RTG (i.e. with no photon coupling), it must be used before any actual bias is applied but after the equilibrium calculations.

In addition to its normal use in initialization, **rtgain_phase** can also be useful to troubleshoot device designs and ensure that the cavity modes are as expected. The simulation may be stopped at this point by using the **stop** statement if the user wishes to examine the result before going any further.

The results of the longitudinal mode search will be shown in the simulation log as shown below:

```
Longitudinal modes found:
Wavelength (um)    Round-Trip-Gain
 0.130504E+001  0.976213E-002
 0.130457E+001  0.122080E-001
 0.130398E+001  0.148400E-001
 0.130352E+001  0.201960E-001
 0.130292E+001  0.271156E-001
 0.130247E+001  0.441081E-001
 0.130186E+001  0.760265E-001
 0.130141E+001  0.226975E+000
 0.130061E+001  0.873341E+000
 0.129981E+001  0.171932E+000
 0.129937E+001  0.505705E-001
 0.129874E+001  0.235128E-001
 0.129832E+001  0.126577E-001
 0.129769E+001  0.754646E-002
 0.129728E+001  0.487527E-002
 0.129664E+001  0.318803E-002
 0.129624E+001  0.224836E-002
 0.129559E+001  0.153203E-002
 0.129521E+001  0.113569E-002
```

However, more information is also available by using the **plot_rtgain** statement in the .rtgain file as shown in Fig. 21.2.

We note in passing that RTG ≥ 1 is an unphysical situation and should be ignored. It usually indicates a point above threshold, which means that the photon density is too strong to be ignored and the evaluation of the RTG is not realistic (i.e. hole burning and carrier clamping effects were neglected or solution is not properly converged yet). To be more accurate, RTG $= 1$ is a limit corresponding to an ideal cavity without spontaneous emission: the RTG only approaches 1 in a real device and there is no clear-cut value of the RTG at threshold.

Gain profile definition

During the RTG preview, a carrier density value must be provided since no bias has been applied yet: this is used to obtain the propagation constant from tabulated gain values. A convenient ways to provide these tables is to include the .gain file instead of the .mater file in the .sol: the .mater file will still be included recursively.

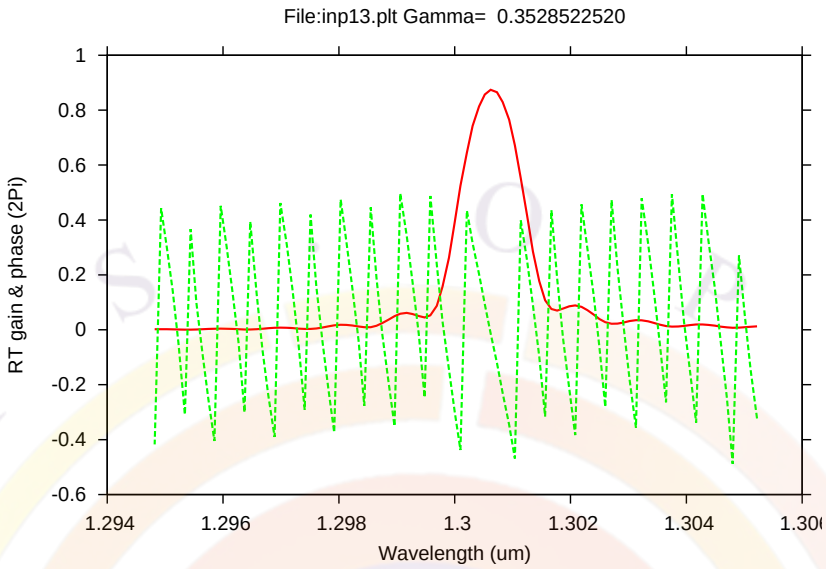


Figure 21.2: Round-trip gain (RTG) of quarter-wave-shifted DFB laser below threshold

The actual gain tables for the preview are generated by the **gain_wavel** statement. If it has not been defined already, a simple .gain template with the necessary statements can be generated by right-clicking the .mater file in SimuCenter. It is a good idea to process this file before the main simulation to optimize the gain curve peak vs. the grating's reference wavelength.

The user may also find it useful to manually add a **gain_density** statement to the .gain file before processing: this will generate a material gain vs. carrier density curve that can help locate a reasonable carrier density at which to preview the RTG.

During the full simulation (i.e. **scan** statements), PICS3D does not need to use tabulated gain values. Instead, it uses the information from the electrical simulation to obtain local gain values: these values will be linearly interpolated from the electrical mesh planes onto the optical mesh points defined by the **section** statements and used to evaluate the RTG.

Bias setup

When applying bias, we still have the same problem of initializing the photon coupling that we described earlier: we need to provide an accurate initial guess of the photon density and the wavelength of the longitudinal modes to ensure the convergence of the Newton solver. As we discussed earlier, this means selecting a bias point below threshold so that the starting photon density is negligible and the flat gain profile approximation is realistic.

However, it is well-known that the longitudinal mode wavelengths can shift below threshold since increasing bias produces carrier-induced index changes. This means that we cannot select a bias point that is too far below threshold since we might risk missing the lasing mode during the mode search. This is a commonly encountered problem in devices with many longitudinal modes such as very long Fabry-Perot cavities.

A reasonable compromise is to select a bias point where the modal gain is positive but not yet lasing. To do this, use the **auto_finish** parameter of the **scan** statement. The RTG value selected to terminate the scan should be larger than what is provided by the mirror feedback but still less than 1.

This **auto_finish** parameter is also required by PICS3D to initialize the coupled RTG solution in the Newton solver. The photon coupling can be turned on immediately afterwards by using **solve_rtg=yes** in the next **scan** statement. Once photon the coupling is turned on, the user must take care to always use small bias steps since the solution state will shift strongly as the threshold region is crossed: although mode wavelengths tend to stabilize as the carrier density is clamped, the photon density of the lasing modes will increase by several orders of magnitude.

Additional hints on how to configure the bias steps in PICS3D are discussed in Chap. 4.

Post-Processing

Once the simulation has been run, results can be plotted using the following .plt file:

```
$file:inp13.plt
begin_pstprc
plot_data plot_device=postscript
get_data  main_input=inp13.sol &&
  sol_inf=inp13.out &&
  xy_data=[5 5]  scan_data=[1 5]
$
plot_scan scan_var=current_1 &&
  variable=rtg_2facet_power_allmode &&
  scan_num= 3

plot_scan scan_var=current_1 &&
  variable=rtg_peak_wavelength &&
  scan_num= 3

modify_plot longitudinal_mode=10
plot_scan scan_var=current_1 &&
```

```
  variable=wavelength &&
  scan_num= 3


gain_spectrum variable=rtg_spectrum


$ Plot at xy cross sections
$
lplot_xy variable=band xy_from=[  0.1000E+00  0.0000E+00] &&
  xy_to=[  0.1000E+00  0.2800E+01] z=0
$
lplot_xy variable=elec_curr_y xy_from=[  0.1000E+00  0.0000E+00] &&
  xy_to=[  0.1000E+00  0.2800E+01] z=0
lplot_xy variable=hole_curr_y xy_from=[  0.1000E+00  0.0000E+00] &&
  xy_to=[  0.1000E+00  0.2800E+01] z=0



vplot_xy variable=total_curr &&
  z=0  grid_sizes=[35 35]
$
cplot_xy variable=wave_intensity &&
  z=0  grid_sizes=[35 35]
splot_xy variable=elec_conc &&
  grid_sizes=(35, 35) &&
  view_xrot=0 view_zrot=30 z=0
splot_xy variable=hole_conc &&
  grid_sizes=(35, 35) &&
  view_xrot=0 view_zrot=30 z=0
$
$ Plot for all dimensions of xyz [for edge laser only].
$
splot_xyz variable=wave_intensity &&
  xy_from=[  0.1000E+00  0.0000E+00] &&
  xy_to=[  0.1000E+00  0.2800E+01] &&
  grid_sizes=[50, 20] view_zrot=20.

modify_plot show_data_points=yes
lplot_xyz variable=wave_intensity xy_point=(0.0 1.4)
$
end_pstprc
```

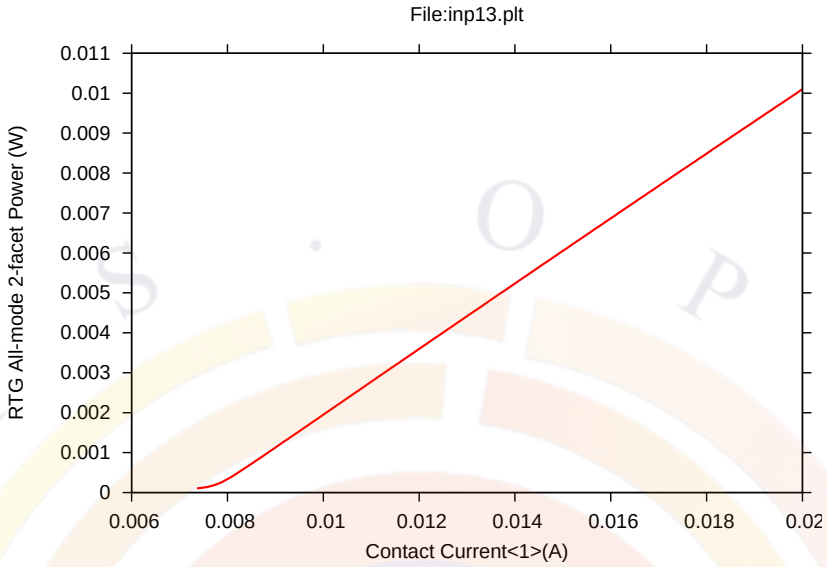


Figure 21.3: L-I curve of DFB laser

When plotting results, there are several differences between LASTIP and PICS3D. The most important one of these is that results are always in 3D so no special variables are needed for unit conversion. Another important point is that unlike LASTIP, PICS3D does not apply any scaling factors due to symmetry: this is a software design choice prompted by the fact that PICS3D must handle many devices with different kinds of symmetry: for example unlike ridge waveguides, VCSELs use rotational symmetry. The user should keep this in mind and apply appropriate scaling of the current and power when comparing PICS3D and LASTIP results for the same structure. Many variable names are also different than in LASTIP: for example, most of the variables obtained from the coupled RTG calculations are prefixed by "rtg_".

To plot the L-I curve for the laser, we use the **plot_scan** statement as in Fig. 21.3 . Power from individual modes can also be plotted by specifying the longitudinal mode index with **modify_plot** before the plot command. Note that the **mode_index** parameter in **plot_scan** is used to specify lateral mode index and cannot be used for this purpose.

plot_scan can also be used to plot other variables vs. bias. For example, Fig. 21.4 shows the peak emission wavelength which confirms our earlier comment about the shift in longitudinal mode positions. The change in photon number can also be plotted in this way.

To plot spectral data, the **gain_spectrum** statement has been re-purposed to plot different physical quantities. One of these is the mode spectrum as shown in Fig. 21.5 which shows the strong side-mode suppression and single-mode behavior that is

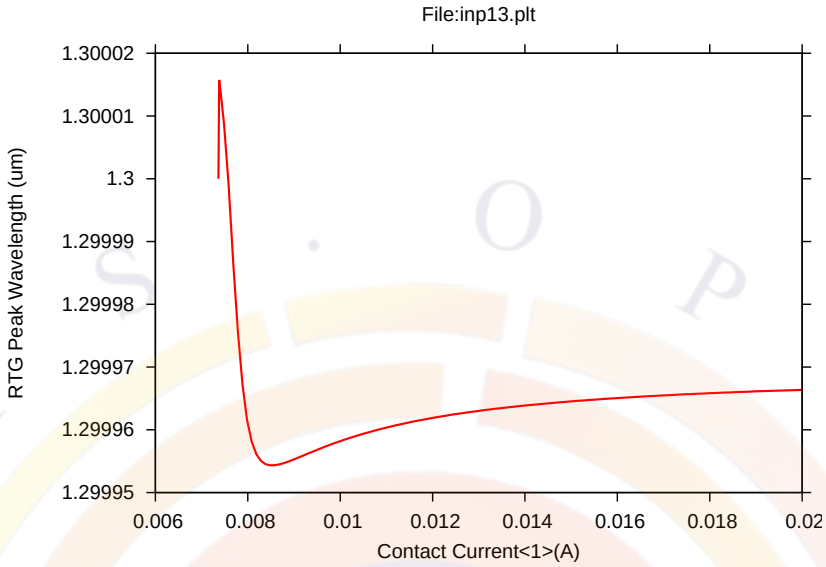


Figure 21.4: Peak emission wavelength of DFB laser vs. bias

expected from this structure.

When plotting structural data, special plotting commands must be used to deal with the 3D data but most of the plotting commands from 2D have a corresponding command in 3D. For example, **plot_1d** is replaced by **lplot_xy** or **lplot_xyz** depending on the direction of the line cut.

As an example of this, Fig. 21.6 shows the longitudinal distribution of the wave intensity: the cut is taken in the center of the active region. As we discussed earlier, we only used seven mesh planes since there is a compromise between the amount of longitudinal sampling and the speed & resources required for the simulation. However, we see that even this rough amount of sampling is enough to obtain good results for the L-I curve and spectral properties of the laser.

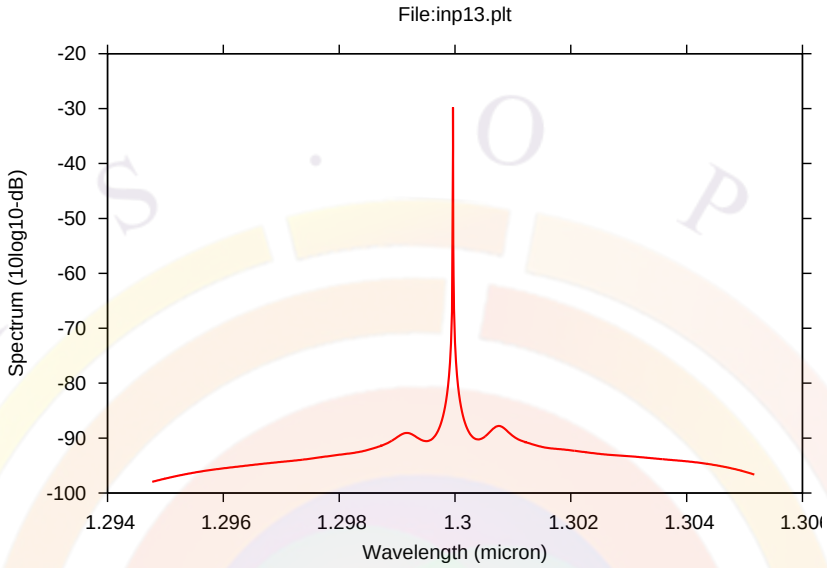


Figure 21.5: Mode spectrum of DFB laser

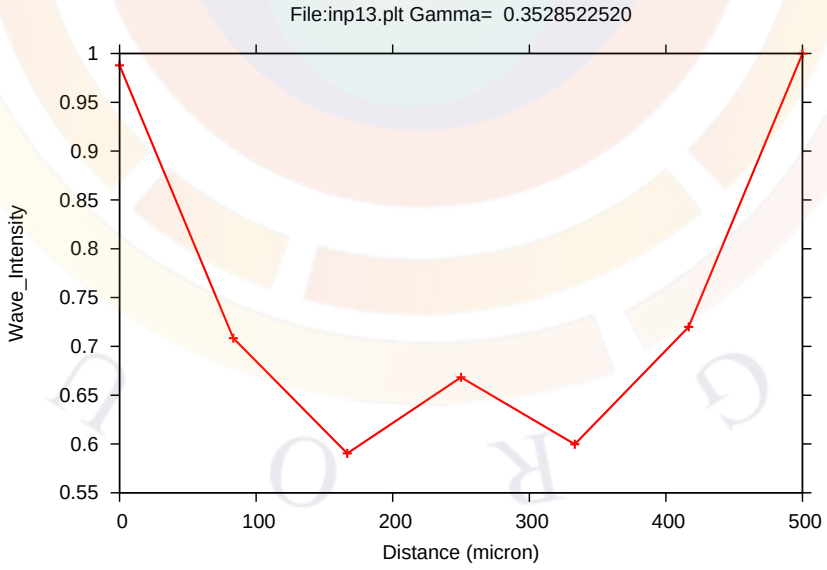


Figure 21.6: Longitudinal mode profile of DFB laser

21.3 amplifier_3D\inp13amp

This example is a semiconductor optical amplifier (SOA): we define incoming power on the left facet and study the outgoing power and amplified spontaneous emission (ASE) of the device. The modeling techniques described here can also be applied to electro-absorbing modulators (EAM) and superluminescent diodes (SLED) with a few simple modifications.

Layer Structure

The layer structure used here is based on the one used in Sec. 21.2. The only difference is that we do not assume there is a grating layer that is abstracted away and defined later in the simulation: in most SOA applications, optical feedback must be minimized so this device really has no grating.

In practical terms, it means that the .layer input file for this example is exactly the same as in Sec. 21.2. In order to save time, we will simply reuse this file instead of creating a new one from scratch. However, for the sake of consistency, we will rename this file to inp13amp.layer so it matches the rest of our project files.

Simulation Setup

Defining the 3D model in an SOA is a bit different than for an edge-emitting laser. The electrical part is the same as before and we use multiple mesh planes to define a simulation volume. Depending on the precision required, more mesh planes may be required: SOA saturation at high power is clearly related to longitudinal spatial hole burning so the quality of the sampling in the z direction may be important.

We use the following .sol file for this simulation:

```
$file:inp13amp.sol
begin
z_structure uniform_zseg_from=0.  uniform_zseg_to=500.  zseg_num=1  &&
  zplanes=5
3d_solution_method 3d_flow=yes

load_mesh mesh_inf=inp13amp.msh
$
include file=inp13amp.gain
include file=inp13amp.doping
$
output sol_outf=inp13amp.out
```

```
more_output pics3d_ase=yes wave_phase=yes
$
$ Set parameters for wave equation
init_wave backg_loss=500 boundary_type=[2 2 1 1] &&
  init_wavel=1.29 wavel_range=[1.28 1.32]
direct_eigen
$
$ ASE model assumes input single wavelength is close to gain peak
$
3d_amplifier_model wavelength=1.29 input_watt=1.e-4
$
$ Set Newton parameters for equilibrium solution
newton_par damping_step=3. var_tol=1.e-8 res_tol=1.e-8 &&
max_iter=100 opt_iter=15 stop_iter=50 print_flag=3
$
equilibrium
$
rtgain_phase  density=1.4e24
$
$
$ Set Newton parameters for solution with bias

newton_par damping_step=1. var_tol=1.e-4 res_tol=1.e-4 &&
max_iter=50 opt_iter=15 stop_iter=9 print_flag=3

$ Ramp up bias voltage (adjust 1 volt for your case)
scan var=voltage_1 value_to=  -1. print_step= 1. &&
  init_step=0.1 min_step=1.e-5 max_step=0.5


$ auto_finish=rtgain is actually used to initialize parameters
scan var=current_1 value_to=18.5e-3 max_step=1.e-3 &&
  auto_finish=rtgain auto_until=0.9


newton_par damping_step=1. var_tol=1.e-4 res_tol=1.e-4 &&
max_iter=50 opt_iter=25 stop_iter=9 print_flag=3


scan var=light value_to=1 init_step=1.e-3 &&
var2=current_1 value2_to=18.5e-3 max_step=0.02 solve_rtg=yes

end
```

```
$ Longitudinal mode session:
begin_zsol
$ this is to control the data points in ASE spectrum
freq_control data_points=200
longitudinal ref_wavel=  0.12900E-05 &&
  left_f_refl=  0.0000E+00 &&
  right_f_refl=  0.0000E+00
section length=  0.5000E-03 &&
  sec_num=1 mesh_points=20
mode_srch omega_xrange=10.
end_zsol
```

3D Optical Simulation

The longitudinal part of the 3D problem is a bit different since the longitudinal behavior is caused by an external light input rather than being due to the existence of cavity modes. The **3d_amplifier_model** statement is used to define the wavelength and power of this input light. Alternatively, **waveguide_input** can also be used.

We will consider here an ideal SOA with no optical feedback so the **section** statements do not define any grating and the **longitudinal** statement defines facet reflectivities of zero.

In the absence of optical feedback, the longitudinal mode solver will not find any modes. However, the solution method is based on a variation of the coupled RTG equations so we need to define the optical propagation correctly. This means that the **longitudinal** statement needs to define a reference frequency that will be used to calculate the detuning term: this can be set as the SOA's input wavelength. We also need to initialize the mode search in the same way as for a DFB laser: the user is referred to Sec. 21.2 for details.

When calculating the ASE spectrum, we need to do two things. The first is to use the **freq_control** to define the number of points in the spectrum. The second is to use **more_output** to force PICS3D to include the ASE spectrum in the output data: it is not included by default since it is not usually relevant in lasers.

Bias setup

When applying bias, we refer to our solar cell example in APSYS (Sec. 19.4) and note that the equilibrium calculations produce a state where no bias exists. The input light to the SOA is a form of bias so it is turned off at the beginning of the simulation.

Just like in Sec. 19.4, we use the "light" scan variable to turn on this input light. However, instead of being a multiple of **light_power**, this variable now represents a fraction of the value defined in **3d_amplifier_model**. A starting point of zero light is also consistent of our usual strategy for the photon coupling: in order to ensure convergence of the coupled RTG equations, we assume a flat gain profile in the initial guess.

As in previous examples, the photon coupling is turned on by using **solve_rtg=yes** in the **scan** statement. Again, this must be preceded by a **scan** statement using **auto_finish=rtgain**. However, we note that in the absence of optical feedback, the round-trip gain value is meaningless: this parameter will only initialize the RTG model and will not terminate the current scan.

Post-Processing

After the simulation has been run, the results can be shown with the following .plt file:

```
$file:inp13amp.plt
begin_pstprc
plot_data plot_device=postscript
get_data  main_input=inp13amp.sol sol_inf=inp13amp.out &&
xy_data=(4 4) scan_data=(4 4)

modify_plot show_data_points=yes

plot_scan scan_var=light variable=rtg_wave_phase
plot_scan scan_var=light variable=rtg_right_power_allmode
plot_scan scan_var=light variable=rtg_ase_left
plot_scan scan_var=light variable=rtg_ase_right

gain_spectrum variable=rtg_asespec_left
gain_spectrum variable=rtg_asespec_right

modify_plot show_data_points=no
$
$ Plot for all dimensions of xyz [for edge laser only].
$
splot_xyz variable=wave_intensity &&
  xy_from=[  0.1000E+00  0.0000E+00] &&
  xy_to=[  0.1000E+00  0.2800E+01] &&
  grid_sizes=[50, 20] view_zrot=20. wave_option=right
$
```

```
$
$ Plot at xy cross sections first
$
lplot_xy variable=band xy_from=[  0.1000E+00  0.0000E+00] &&
  xy_to=[  0.1000E+00  0.2800E+01] z=5
splot_xy variable=elec_conc z=5. grid_sizes=(35, 35) &&
  xrange=[  0.0000E+00  0.1000E+01] &&
  yrange=[  0.0000E+00  0.2800E+01] &&
  view_xrot=0 view_zrot=30
$
vplot_xy variable=elec_curr &&
  xrange=[  0.0000E+00  0.1000E+01] &&
  yrange=[  0.0000E+00  0.2800E+01] &&
  z=5  grid_sizes=[35 35]
$
cplot_xy variable=wave_intensity &&
  xrange=[  0.0000E+00  0.1000E+01] &&
  yrange=[  0.0000E+00  0.2800E+01] &&
  z=5  grid_sizes=[35 35]
end_pstprc
$
```

Again, we use **plot_scan** to show results vs. bias. In Fig. 21.7, we show the expected signal saturation of the SOA as the input power (i.e. "light" variable) is varied. We also see in Fig. 21.8 that this is accompanied by a reduction in the amount of amplified spontaneous emission. Note that the amount of ASE depends on which facet it is being observed.

The ASE spectrum can be plotted in the same way as the RTG spectrum in a laser: a specific variable has been defined to use in conjunction with the **gain_spectrum** statement as shown in Fig. 21.9.

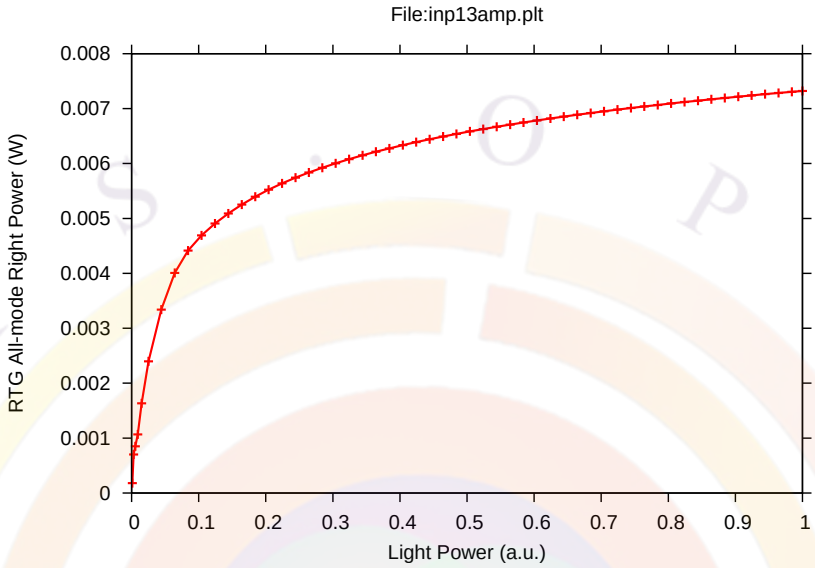


Figure 21.7: Output power of SOA vs. input light

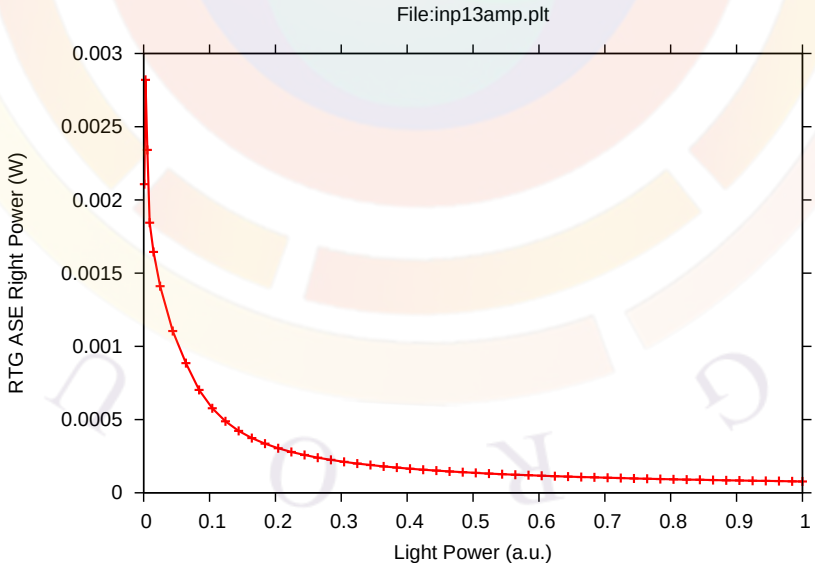


Figure 21.8: ASE on right facet of SOA vs. input light

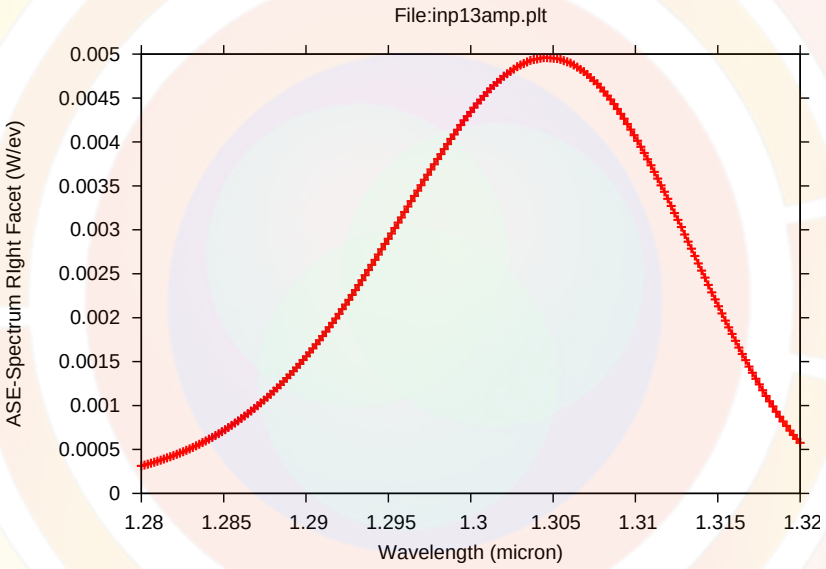


Figure 21.9: ASE spectrum on right facet of SOA

21.4 DBR\3section_tunable

This example[1] is a tunable edge-emitting laser with a Distributed Bragg Reflector (DBR). It consists of three segments: a gain region on the left, a tuner in the middle and a Bragg mirror on the right. We use this example to show how PICS3D handles complex structures where the device structure changes in the longitudinal direction.

We also use this example to show how to explicitly define grating layers. Unlike the simplified method shown in Sec. 21.2, here PICS3D will calculate the coupling coefficient (κ) by using the index profile of the grating and its overlap with the optical mode.

Layer Structure

Since the device structure varies longitudinally in this example, we need multiple 2D cuts and .layer files: each of these cuts represents a single segment that will be defined later in the .sol file.

The different .layer files can be created independently from each other using the same techniques discussed in our other examples. However, in order to process these files correctly and generate correct material numbers, the **previous_layer** statement should be used after the first layer.

The user can also automatically generate all the .layer files for this project at once by using the Layer3D GUI program (shown in Fig. 21.10). Each of the 2D cuts are built using the same rules as the LayerBuilder GUI but the **previous_layer** statement will be added automatically.

The gain segment is very similar to previous devices we have considered:

```
begin_layer

column column_num=1 w=1.5  mesh_num=2  r=1.0
bottom_contact column_num=1 from=0.0 to=1.5 contact_num=1

layer_mater macro_name=inp column_num=1
layer d=1.5 n=8  r=0.8 n_doping1=1e24

layer_mater macro_name=ingaasp var1=0.568 column_num=1 &&
  var_symbol1=y
layer d=0.08 n=4  r=0.9

$ total thickness of the following MQW is 0.15 um
```

[1] Last updated Nov. 2011

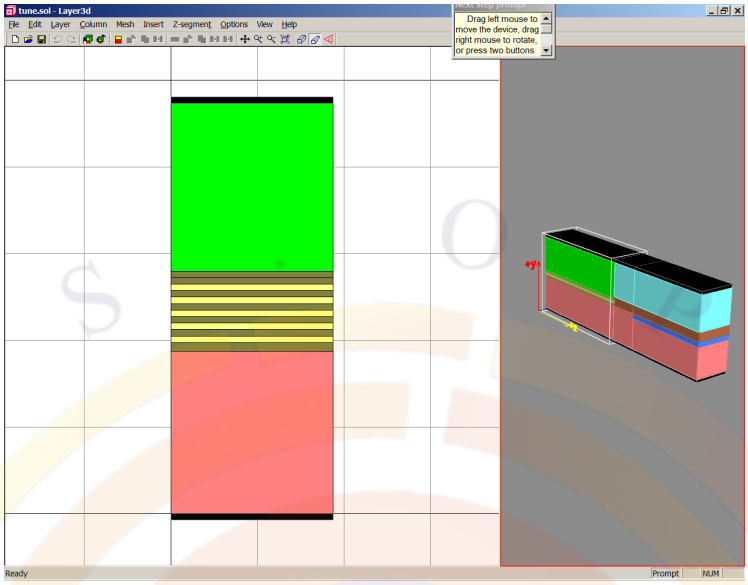


Figure 21.10: 2D & 3D view of tunable DBR laser

```
include file=tune1.well
include file=tune1.bar
include file=tune1.well
include file=tune1.bar
include file=tune1.well
include file=tune1.bar
include file=tune1.well
include file=tune1.bar
include file=tune1.well
include file=tune1.bar

layer_mater macro_name=ingaasp var1=0.568 column_num=1 &&
  var_symbol1=y
layer d=0.05 n=4  r=1.1

layer_mater macro_name=inp column_num=1
layer d=1.55 n=9  r=1.2 p_doping1=5e23

top_contact column_num=1 from=0.0 to=1.5 contact_num=2
end_layer
```

The active region is an unstrained MQW InGaAsP region tuned to peak around 1.55 μm. Again, we use external files and the **include** statement to simplify this declaration:

```
layer_mater macro_name=ingaasp var1=0.568 column_num=1 var_symbol1=y
layer d=0.0225 n=3  r=1.


layer_mater macro_name=ingaasp var1=0.950 column_num=1 &&
  active_macro=InGaAsP/InP avar1=0.443 avar2=0.95 &&
  avar3=0.262 avar4=0.568 &&
var_symbol1=y avar_symbol1=xw avar_symbol2=yw avar_symbol3=xb avar_symbol4=yb
layer d=0.0075 n=3  r=1.
```

Users working with the LayerBuilder GUI program can also copy & paste layers to easily create a MQW region.

For the tuning section, we need to consider what is the mechanism involved. In this case, we wish to consider bias-induced index change due to interband transitions: this is modeled by an active region and an active macro just like in the gain segment.

However, this particular active region has a very different bandgap and only the tail edge of the the gain curve is visible in the 1.55 μm range. We call this a "mildly active" layer for historical reasons: the gain is calculated but it is so small that it has no practical effect beyond contributing to the index change.

The resulting layer file for the tuning section is shown below:

```
begin_layer

$ Make sure the program knows there is a segment before this structure.
$ The programs needs this information to generate the material numbers
$ correctly.

previous_layer file=tune1.layer

column column_num=1 w=1.5  mesh_num=2  r=1.0
bottom_contact column_num=1 from=0.0 to=1.5 contact_num=1

layer_mater macro_name=inp column_num=1
layer d=1.5 n=8  r=0.8 n_doping1=1e24

$ Passive waveguide material declared to be "active"
$   so that interband optical model for index change can be used.

layer_mater macro_name=ingaasp var1=0.5 column_num=1 &&
  active_macro=InGaAsP avar1=0.5 &&
  var_symbol1=y avar_symbol1=yw
layer d=0.38 n=9  r=1.
```

```
layer_mater macro_name=inp column_num=1
layer d=1.45 n=9  r=1.2 p_doping1=5e23

top_contact column_num=1 from=0.0 to=1.5 contact_num=3

end_layer
```

For the DBR segment, we use the same "mildly active" region as the tuner (unstrained InGaAs(0.5)P) as the base of the grating. Conceptually, we can imagine that we partially etch this region and regrow with InP to form an average material (unstrained InGaAs(0.25)P).

We quickly note that the regrowth region is below the original layer so things look upside down. This is simply due to our usual convention of defining devices n-side down which can be confusing if the original design is on a p-type substrate. However, this convention is merely for the sake of consistency and the user is free to change it.

When defining the grating layer, we start by defining a layer made of the average material in **layer_mater**. However, this is overridden immediately afterwards by the **grating_compos** statement: the original layer material is replaced by a virtual embedded structure and two sets of material macros are used for the same material. This is shown in the following excerpt of the tune3.mater file, which generated when all the layer files are processed:

```
$ -->grating region based on composition variation
grating_model grating_order=1 &&
  use_active_mater=   5 &&
  d_high=  0.100000000000E+000 d_low=  0.100000000000E+000 &&
  d_fall=  0.000000000000E+000 d_rise=  0.000000000000E+000
$
$ -->high index grating region
load_macro name=ingaasp mater=   5 &&
  var_symbol1=y  var1=  0.5000E+00
get_active_layer name=InGaAsP mater=   5 &&
  var_symbol1=yw  var1=  0.5000E+00
active_reg embedded_structure=1 mater=   5 &&
  thickness=  0.200000000000E+000
$
$ -->low index grating region
load_macro name=ingaasp mater=   5 &&
  var_symbol1=y  var1=  0.0000E+00
get_active_layer name=InGaAsP mater=   5 &&
```

```
  var_symbol1=yw  var1=  0.0000E+00
active_reg embedded_structure=1 mater=   5 &&
  thickness=  0.200000000000E+000
```

When defining the grating composition, we specify both passive and active macros for the low and high index regions. This means that the grating is also a "mildly active" layer: the index step will be affected by interband transitions. Note that the bandgap values for the high and low regions are still far away from 1.55 μm: otherwise, this would a gain- or loss-coupled laser, which a very different kind of device. The user can also choose to omit the active macro: this means that the index step of the grating will be frozen at its equilibrium value.

Alternatively, one can also define the grating using the **grating_model** statement in the .layer file. In this case, the refractive index of the high and low regions must be provided instead of the macro parameters. Also, the average material defined in **layer_mater** will be used in the electrical simulation instead of being overridden with the grating macros.

The layer file used for the DBR segment is shown below:

```
begin_layer

$ Make sure the program knows there is a segment before this structure.
$ The programs needs this information to generate the material numnbers
$ correctly.

previous_layer file=tune2.layer

column column_num=1 w=1.5  mesh_num=2  r=1.0
bottom_contact column_num=1 from=0.0 to=1.5 contact_num=1

layer_mater macro_name=inp column_num=1
layer d=1.3 n=8  r=0.8 n_doping1=1e24

$
$ 0.2 um corrugation grating is made between InP and InGaAs(0.5)P
$ So the average material is InGaAs(0.25)P. Let us declare it
$ as mild region to let the program treat it mildly.
$
layer_mater  macro_name=ingaasp var1=  0.25 &&
  active_macro=InGaAsP avar1=  0.25 &&
  column_num=1 embedded_structure=1 &&
  var_symbol1=y avar_symbol1=yw
grating_compos column_num=1  &&
```

```
  d_high=  0.1 d_low=  0.1 &&
  hi_macro_name=ingaasp hi_var1=0.5 &&
  hi_active_macro=InGaAsP hi_avar1=0.5 &&
  lo_macro_name=ingaasp lo_var1=0. &&
  lo_active_macro=InGaAsP lo_avar1=0. &&
  hi_var_symbol1=y  &&
  hi_avar_symbol1=yw &&
  lo_var_symbol1=y &&
  lo_avar_symbol1=yw

layer d=  0.2 n=  9 &&
  n_doping1=  1.E+23 &&
  r=  0.9

$ Passive waveguide material declared to be "active"
$   so that interband optical model for index change can be used.

layer_mater macro_name=ingaasp var1=0.5 column_num=1 &&
  active_macro=InGaAsP avar1=0.5 &&
  var_symbol1=y avar_symbol1=yw
layer d=0.38 n=9  r=1.


layer_mater macro_name=inp column_num=1
layer d=1.45 n=9  r=1.2 p_doping1=5e23

top_contact column_num=1 from=0.0 to=1.5 contact_num=4
end_layer
```

Simulation Setup

The following .sol file is used in this simulation:

```
$***********
begin
use_macrofile macro1=my.mac

3d_solution_method 3d_flow=yes z_connect=no
z_structure uniform_zseg_from=0. uniform_zseg_to=400. zseg_num=1 zplanes=3
z_structure uniform_zseg_from=400. uniform_zseg_to=500. zseg_num=2
z_structure uniform_zseg_from=500. uniform_zseg_to=800. zseg_num=3 zplanes=3
```

```
load_mesh mesh_inf=tune1.msh zseg_num=1
load_mesh mesh_inf=tune2.msh zseg_num=2
load_mesh mesh_inf=tune3.msh zseg_num=3

output sol_outf=tune.out

begin_zmater zseg_num=1
include file=tune1.gain
include file=tune1.doping
end_zmater

begin_zmater zseg_num=2
include file=tune2.gain
include file=tune2.doping
end_zmater

begin_zmater zseg_num=3
include file=tune3.gain
include file=tune3.doping
end_zmater


optical_field profile=effective_index x_segment=4
init_wave backg_loss=500 &&
  boundary_type=(2 2 1 1 )  init_wavel=1.55 &&
  wavel_range=(1.5, 1.60)

prop_constant_model  zseg_num=1 precalculated_index=yes
prop_constant_model  zseg_num=2 precalculated_index=yes
prop_constant_model  zseg_num=3 precalculated_index=yes

newton_par damping_step=10. var_tol=1.e-9 res_tol=1.e-9 &&
  max_iter=100 opt_iter=15 stop_iter=50 print_flag=3 mf_solver=2
equilibrium
rtgain_phase  density=1.6e24 zseg_num=1
rtgain_phase  density=1.6e24 zseg_num=2
rtgain_phase  density=1.6e24 zseg_num=3

$parallel_linear_solver
newton_par damping_step=10.0 var_tol=1.e-4 res_tol=1.e-4 &&
  max_iter=30 opt_iter=16 stop_iter=17 print_flag=3 mf_solver=2

scan var=voltage_1 value_to=-0.8
```

```
scan var=current_2 value_to=-2.4e-3 init_step=1e-6 &&
min_step=1e-8 max_step=0.1e-3 &&
var2=current_3 value2_to=-0.3e-3 &&
var3=current_4 value3_to=-0.9e-3 &&
  auto_finish=rtgain auto_until=0.9 auto_within=0.09


scan var=current_2 value_to=-7.2e-3 &&
var2=current_3 value2_to=-0.3e-3 &&
var3=current_4 value3_to=-0.9e-3 &&
init_step=1e-6  min_step=1e-8 max_step=0.5e-3 solve_rtg=yes

newton_par damping_step=10.0 var_tol=1.e-3 res_tol=1.e-1 &&
  max_iter=30 opt_iter=16 stop_iter=17 print_flag=3 mf_solver=2

scan var=current_4 value_to=-18.e-3 &&
var3=current_2 value3_to=-7.2e-3  &&
var2=current_3 value2_to=-0.3e-3 &&
init_step=1e-6  min_step=1e-8 max_step=0.4e-3 solve_rtg=yes

$ alternatively, one may tune the laser by changing the middle phase section
$scan var=current_3 value_to=-10.e-3 &&
$var3=current_2 value3_to=-7.2e-3  &&
$var2=current_4 value2_to=-0.9e-3 &&
$init_step=1e-6  min_step=1e-8 max_step=0.4e-3 solve_rtg=yes


end
$ ********************************************************
begin_zsol
longitudinal ref_wavel=1.55e-6 left_f_refl=0.3 right_f_refl=0.3
section length=400.e-6 sec_num=1 mesh_points=10
section length=100.e-6 sec_num=2 mesh_points=10
section length=300.e-6 sec_num=3 mesh_points=10
mode_srch omega_xrange=16.  adjust_range=yes
end_zsol
```

Like all edge-emitting devices in PICS3D, this is a 3D simulation with multiple mesh planes so we must use the **3d_solution_method** statement with **3d_flow=yes**. We also use three different **z_structure** statements to define the segment lengths as well as the number of mesh planes within each segment.

The 2D mesh used at the various z points in each segment is loaded by a series of **load_mesh** statements. Material properties are loaded in the same way as in Sec. 21.2 but since there is more than one segment, the **include** statements for each segment must be bracketed by **begin_zmater** and **end_zmater** to identify the segment being defined.

Optical Propagation Model

To define the optical cavity, we use three sections to match the segments defined above. Note that no grating is defined at this stage: the earlier grating definition will be used calculate the coupling coefficient automatically. Based on the index step and wave profile, the solver prints this kind of information in the simulation log at each bias step:

```
At lambda=   1.55379871355004
Grating Reg. Opt. Conf.=  0.180108918999968
Real kappa (1/m)=   36386.4705291671
Imag kappa (1/m)= -5.748666808201413E-013
```

As we can see, the gain coupling coefficient is negligible, which is consistent with our "mildly active" model. The real part of the grating indicates a highly reflective mirror ($\kappa L \approx 10.9$ for this segment).

To calculate the round-trip gain (RTG), we need a reference point in the cavity. For best results, this point should be in the middle of the gain region so we modify the default value of **ref_midpoint** in the **longitudinal** statement. We also use this statement to specify the facet boundary conditions and the reference wavelength for the longitudinal mode search.

To save on computation time, we also use **prop_constant_model** to instruct the software to use precalculated index change tables from the .gain preview. This may be less accurate then calculating the index automatically but often helps solve certain convergence issues.

Bias setup

When defining the .layer files above, we note that we used the same electrode number (1) for the n-side bottom electrode but we used a different number (2,3,4) for the top p-side electrode. This means that the bottom electrode is shared across all segments whereas the top electrodes are split and can be used to independently bias each individual segment.

Since at the equilibrium point, we define a situation where all electrodes are grounded at 0 V, this can create unexpected bias configurations. For example, if one were to

ramp up the bias on electrode (#2) only, the current would naturally flow towards the other top electrodes since that is the shortest electrical path. This is not the desired situation so we apply a small bias to the bottom shared electrode (#1) instead. This method shifts the reference ground and is equivalent to biasing all the top electrodes simultaneously.

After this initial voltage scan, there is a trickle of current flowing down in each segment so we can use a current bias on the top electrodes and bias each individual segment separately. We note that by convention, current flowing into an electrode is positive (and vice versa) so that the bias currents on the p-side electrodes are negative in a forward bias situation.

When applying bias on the top electrodes, we use multiple scan variables: this is because we want to control multiple currents simultaneously. If we omit a particular electrode from the scan (i.e. electrode #1), then only its voltage value would be controlled: it would be kept at its previous value during the new scan. This means that as we apply current on the top electrodes, the voltage of these electrodes changes; meanwhile, the voltage on the bottom electrode stays fixed at -0.8 V but the current on this electrode changes.

The current scans are split into three steps: the first step initializes the coupled RTG model at a point below threshold, as described in Sec. 21.2. The second one ramps the gain segment above threshold to produce the L-I curve of the laser. The last scan keeps the gain segment current at its previous value but applies a tuning current to the middle segment. In the last two scans, the photon coupling is turned on with **solve_rtg=yes**.

Post-Processing

Once the simulation has been run, results can be plotted using the following .plt file:

```
$file:tune.plt
$ **************
begin_pstprc
plot_data plot_device=postscript


get_data sol_inf = tune.out main_input=tune.sol scan_data=(1 4)

plot_scan scan_var=current_1 variable = rtg_left_power_allmode

get_data sol_inf = tune.out main_input=tune.sol scan_data=(5 5) &&
xy_data=(5 5)
```

```
plot_scan scan_var = current_4 variable = rtg_left_power_allmode &&
 scale_horizontal=-1
plot_scan scan_var = current_4 variable =rtg_peak_wavelength &&
 scale_horizontal=-1

get_data sol_inf = tune.out main_input=tune.sol &&
xy_data=(4 4)
gain_spectrum variable=rtg_spectrum
get_data sol_inf = tune.out main_input=tune.sol &&
xy_data=(5 5)
gain_spectrum variable=rtg_spectrum

lplot_xy variable=band xy_from=(0.5 1.1) &&
  xy_to=(0.5 2.2) z=200.
splot_xyz variable=wave_intensity xy_from=(0.05, 0.0) xy_to=(0.05 3.38) &&
  grid_sizes=(80, 20) view_zrot=20.
end_pstprc
```

As before, we use the **plot_scan** statement to plot bias-related data. However because of our current convention, we need to use a scaling factor of -1 to plot curves in the usual direction when using the current of the top electrodes.

The L-I curve during bias ramp is shown in Fig. 21.11. We plot versus the current on electrode #2 since the bottom electrode has current contributions from other segments. We also plot the power on the left facet since we have already determined that the DBR section on the right is highly reflective.

To plot the results of the tuning, we do the same with current #3. As can be seen in Fig. 21.12 a), the tuning current produces small variations in the output power of the device. This is due to changes in the position of the longitudinal modes with respect to the peaks of the gain curve and DBR reflection spectrum. As the power of one mode goes down, a new mode becomes favored by the cavity and a mode hop occurs. This can be shown by plotting the changes in the peak mode wavelength vs bias, as seen in Fig. 21.12 b).

For structural data, the same rules as in Sec. 21.2 apply.

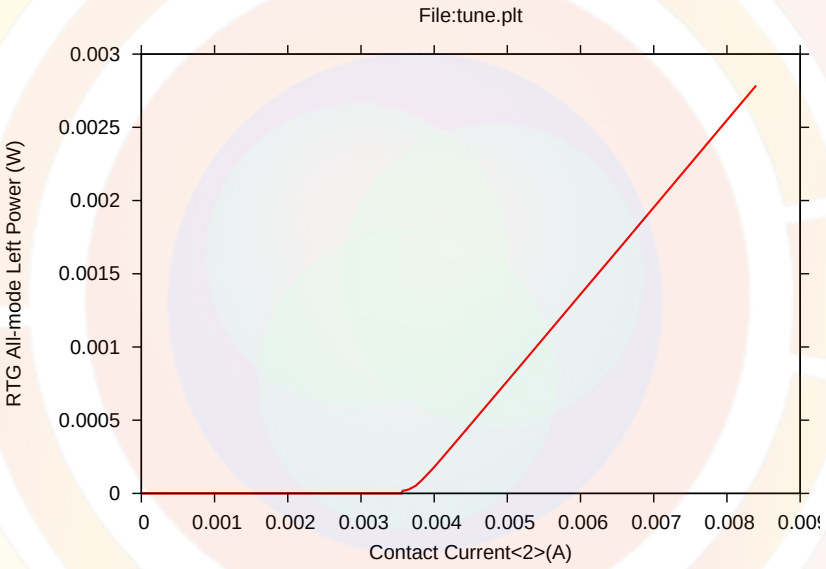
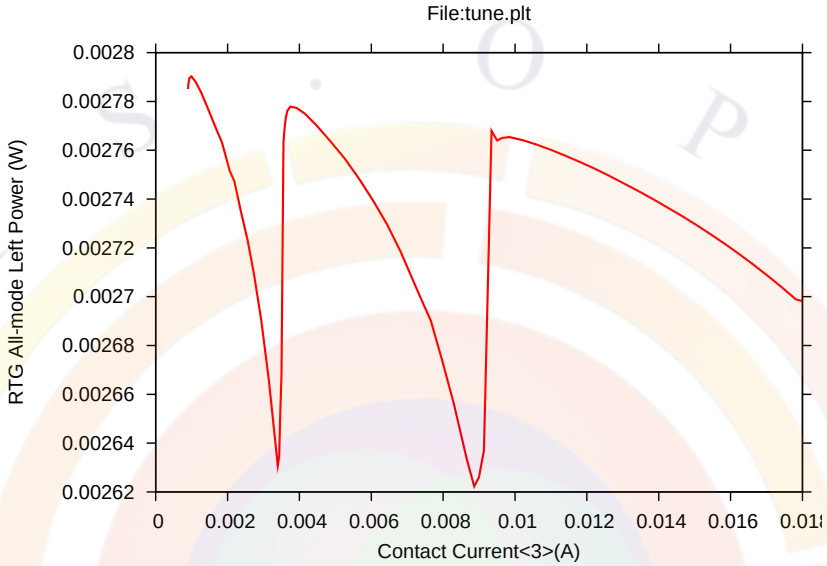
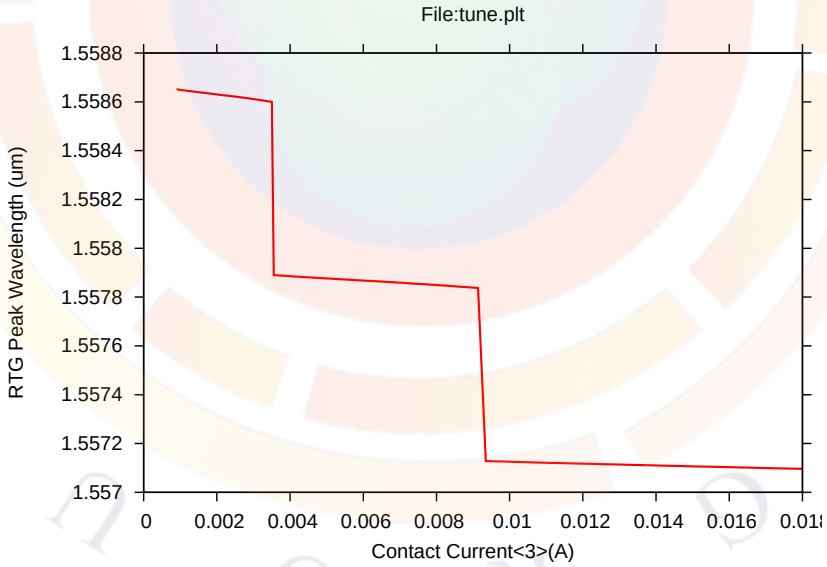


Figure 21.11: Bias ramp in gain segment of tunable DBR laser



(a)



(b)

Figure 21.12: Tuning characteristics of DBR laser

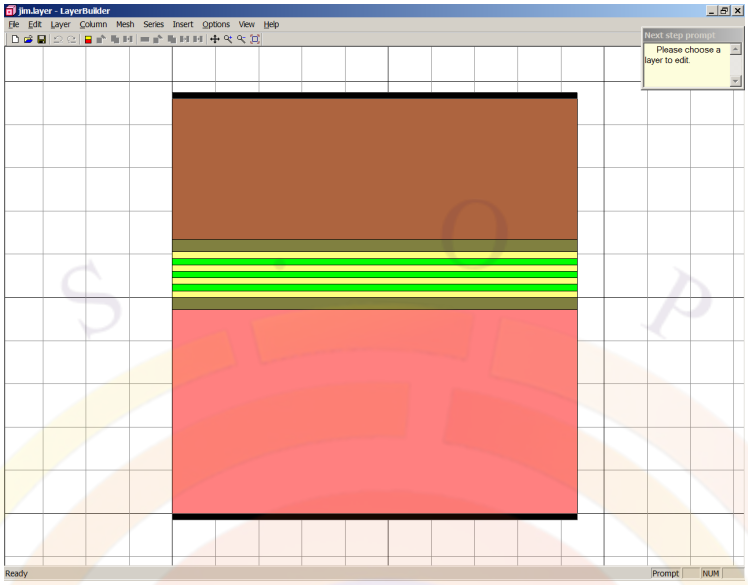


Figure 21.13: 2D cut of VCSEL

21.5 VCSELS\jim_vcsel

This example is a basic vertical cavity surface emitting laser (VCSEL). Since the optical propagation takes place perpendicularly to the active region, we will use a different method to define segments and sections.

Layer Structure

This VCSEL is a GaAs-based MQW device designed to lase at 0.835 μm. The DBRs consist of alternating $Al_{0.25}Ga_{0.75}As$/AlAs quarter-wave layers. However, since it would require a lot of mesh to model all the layers of the DBR stack, we approximate this region by an average material ($Al_{0.625}Ga_{0.375}As$) as shown in Fig. 21.13. This is only for the purposes of the electrical simulation though: to model the optical propagation, we must explicitly consider the alternating layers of the DBR.

Optical Propagation Model

Since the light propagation takes place in the vertical direction, the .layer file of a VCSEL must also define the optical cavity. Strictly speaking, each individual layer should be considered as an individual section for the purposes of propagation but that would require too much computation time.

Instead, groups of layers are assigned to the same section by assigning them a **vcsel_type** label in the **layer** statement. This label refers to a section specified

by **vcsel_section** which defines the optical propagation model used for a particular layer or group of layers. New users of PICS3D may find it easier to use the Layer-Builder GUI to define VCSEL sections: right-click on a given layer and select the VCSEL tab.

For example, the average layer used to represent the DBR electrically is assigned a model that describes propagation in a periodic stack of layers. The spacer region is a single layer and is treated as such optically. For reasons that are explained below, the MQW region is also approximated by a single layer.

When defining VCSEL sections, care must be taken not to interleave section labels: it is not supported by the software at this time. For example, assigning a particular section label (e.g. "b") to the barrier region and another to the well region (e.g. "w") may result in the label sequence "b/w/b" which would be an error and produce incorrect section lengths. Therefore, we recommend that a single VCSEL section label be used for the entire MQW region.

In a simple VCSEL design such as this one, the user needs to define only five VCSEL sections: n-DBR, n-spacer, mqw_active, p-spacer, p-DBR. This is shown in the .layer file below:

```
begin_layer

$layer_input_convention layer_unit=relative ref_wavelength=0.84

$
column column_num=1 w=7.5  mesh_num=4  r=1.
$
bottom_contact column_num=1 from=0 to=7.5 contact_num=1
$
$ to save mesh, use effective media:  AlGaAs Al=0.625
$ n-mirror
$

vcsel_section vcsel_type=n-dbr &&
  dbr_period_from_macro=yes &&
  active=no mesh_points=10

$ this is the effective medium
layer_mater macro_name=algaas var1=0.625 column_num=1 var_symbol1=x

$ let's define a DBR period using macro like this (use column 1 only)
$ (also possible to define grading within a DBR period)
vertical_dbr_layer_mater macro_name=algaas var_symbol1=x var1=0.25 &&
  thick=0.0595
```

```
vertical_dbr_layer_mater macro_name=algaas var_symbol1=x var1=1. &&
  thick=0.0706

$ thickness here is actually determined by DBR periods above
layer d=1.  n=15  r=0.9 &&
  n_doping1=2.e24  vcsel_type=n-dbr use_dbr_period=29
$
$ add a spacing layer Al(0.6) GaAs of 1 lambda:  0.825/3.2=0.26
$
$ We choose the spacer carefully because the emission
$   wavelength depends on it.
$
vcsel_section vcsel_type=n-spacer &&
  grating_model=1layer active=no &&
  mesh_points=5
layer_mater macro_name=algaas var1=0.6 column_num=1 var_symbol1=x
layer d=0.23  n=6  r=0.9 vcsel_type=n-spacer
$
$ MQW system:
$
vcsel_section vcsel_type=mqw_active &&
  grating_model=1layer active=yes &&
  mesh_points=9
include file=jim.bar
include file=jim.well
include file=jim.bar
include file=jim.well
include file=jim.bar
include file=jim.well
include file=jim.bar
$
$
$ add a spacing layer Al(0.6) GaAs of 1 lambda:  0.825/3.2=0.26
$
$ We choose the spacer carefully because the emission
$   wavelength depends on it.
$
vcsel_section vcsel_type=p-spacer &&
  grating_model=1layer active=no &&
  mesh_points=5
$
layer_mater macro_name=algaas var1=0.6 column_num=1 var_symbol1=x
layer d=0.23  n=6  r=0.9  vcsel_type=p-spacer
```

```
$
$ to save mesh, use effective media:  AlGaAs Al=0.625
$

vcsel_section vcsel_type=p-dbr &&
  dbr_period_from_macro=yes &&
  active=no mesh_points=10

$ effective medium layer
layer_mater macro_name=algaas var1=0.625 column_num=1 var_symbol1=x

$ let's define a DBR period using macro like this (use column 1 only)
$ (also possible to define grading within a DBR period)
vertical_dbr_layer_mater macro_name=algaas var_symbol1=x var1=1. &&
  thick=0.0706
vertical_dbr_layer_mater macro_name=algaas var_symbol1=x var1=0.25 &&
  thick=0.0595

$ thickness here is actually determined by DBR periods above
layer d=1.  n=12  r=1.1 &&
  p_doping1=3.e24  vcsel_type=p-dbr use_dbr_period=20
$
top_contact column_num=1 from=0 to=7.5 contact_num=2
$
end_layer
```

Note that we used the **include** statement above to repeat the MQW more easily. The well and barrier regions are defined below:

```
layer_mater macro_name=algaas var1=0.25 &&
  column_num=1 var_symbol1=x
layer d=0.0100  n=3  r=1. xp1=1 xp2=1 vcsel_type=mqw_active

layer_mater macro_name=gaas &&
  column_num=1 active_macro=AlGaAs/AlGaAs &&
  avar1=0. avar2=0.25 &&
  avar_symbol1=xw avar_symbol2=xb
layer d=0.0070  n=3  r=1. xp1=1 xp2=1 vcsel_type=mqw_active
```

When using the LayerBuilder GUI, MQW regions can also be created easily by copying and pasting layers. However, when using this method, it is recommended that the VCSEL sections be defined only after all layers have been copied. The

reason for this is that VCSEL section labels get copied right along with the layer itself which can inadvertently produce alternating section labels.

The thickness of the n and p spacer regions determines the cavity length and thus, the lasing wavelength. Since longitudinal modes are found where the round-trip phase is a multiple of π, it is common to use a quick rule (thickness $= \frac{\lambda}{n}$) when designing VCSELs.

However, this approximation does not include the phase contribution from the DBR mirrors (penetration depth) and propagation in the active region (non-zero thickness). Therefore, choosing the spacer thickness is an iterative process: the estimate above can be used as a starting guess but the optimal thickness will likely be a bit smaller. We recommend using the round-trip gain (RTG) preview from .sol to facilitate the procedure; see below or Sec. 21.2 for details.

When the .layer file is processed, it creates a .vcsel file in addition to the usual output files. This contains various statements such as **section** that define the longitudinal cavity model that will be used by PICS3D in the .sol file. The starting and ending points of these sections should correspond to the layer positions defined above:

```
$
$  type:n-dbr
section length=  0.377290000000E-005 &&
  sec_num= 1 mesh_points= 10 &&
  active=no dbr_period_mater=   1
passive_3d nref_type2=2.e24 index_nref=2.e24 &&
  sec_num = 1
$
$  type:n-spacer
section grating_model=1layer &&
  length=  0.230000000000E-006 &&
  sec_num= 2 mesh_points=  5 &&
  active=no
passive_3d nref_type2=2.e24 index_nref=2.e24 &&
  sec_num = 2
$
$  type:mqw_active
section grating_model=1layer &&
  length=  0.610000000000E-007 &&
  sec_num= 3 mesh_points=  9 &&
  active=yes
$
$  type:p-spacer
section grating_model=1layer &&
  length=  0.230000000000E-006 &&
```

```
  sec_num= 4 mesh_points=  5 &&
  active=no
passive_3d nref_type2=2.e24 index_nref=2.e24 &&
  sec_num = 4
$
$  type:p-dbr
section length=  0.260200000000E-005 &&
  sec_num= 5 mesh_points= 10 &&
  active=no dbr_period_mater=   5
passive_3d nref_type2=2.e24 index_nref=2.e24 &&
  sec_num = 5
section_location start=  0.000000000000E+000 end=  0.689590000000E+001
```

Simulation Setup

The following .sol file is used to run the simulation:

```
$file:jim.sol
$***********
begin
load_mesh mesh_inf=jim.msh
include file=jim.gain
include file=jim.doping
output sol_outf=jim.out
more_output rtgain_scan=yes

$
$ *************************
$ VCSEL parameters:
$
vcsel_model index_core=3.2 index_cladding=1.0 &&
  core_radius =7.5  bessel_order=0
sor_par  max_iter=0
init_wave backg_loss=500 init_wavel=0.83  wavel_range=(0.75, 0.90)
cylindrical axis=y
$
newton_par damping_step=5. var_tol=1.e-9 res_tol=1.e-9 &&
  max_iter=100 opt_iter=15 stop_iter=50 print_flag=3
$restart
equilibrium
rtgain_phase  density=4.5e24
$stop
```

```
$
newton_par damping_step=2. var_tol=1.e-2 res_tol=1.e-3 &&
  max_iter=50 opt_iter=25 stop_iter=10 print_flag=3 change_variable=yes

scan var=voltage_1 value_to=-1.3 print_step=1.3 &&
  init_step=0.2 min_step=1.e-5 max_step=0.5

$ better to start with low RTG and progress slowly
$ auto_finish=rtgain is mandatory to get RTG ready
$
scan var=current_1 value_to=8.e-3  print_step=0.15e-3 &&
  init_step=0.1e-4 min_step=1.e-6 max_step=0.5e-3 &&
  auto_finish=rtgain auto_until=0.95 auto_condition=above


$ it is wise to start with a small step here.

scan var=current_1 value_to=8.e-3  solve_rtg=yes  &&
  init_step=0.01e-3 max_step=0.1e-3

$
end
$*****************************
begin_zsol
longitudinal ref_wavel=0.825d-6 left_f_refl=0.32 right_f_refl=0.32

$
include file=jim.vcsel
$
mode_srch  iter_num=75  wavel_xrange=(0.75e-6 0.9e-6)
bias3d step_num=10
end_zsol
$
```

As we previously discussed in Sec. 6.4, all PICS3D simulations are in 3D. However, since this VCSEL has rotational symmetry, only a single mesh plane is required: this means that most of the setup steps are the same as in a 2D simulation. What turns this example into a 3D simulation is the simple addition of the **cylindrical** statement: the y axis of the .layer file becomes the z axis of the cylindrical system defined in 6.2.

Since this simulation is using cylindrical coordinates, a different mode solver must be used. In this example, we use the fiber-like EIM model from Sec. 17.2 as defined in the **vcsel_model** statement. As a result, many of the parameters from **init_wave** are

inoperative. Here, this statement should only be used to define an initial wavelength estimate for the mode solver, a wavelength range for material properties as well as the background propagation loss coefficient.

Optical Propagation Model

The RTG is calculated by using a reference point in the cavity. For best results, this point should be in the middle of the gain region so we modify the default value of **ref_midpoint** in the **longitudinal** statement. We also use this statement to specify the facet boundary conditions that lie beyond the DBR layers and the reference wavelength for the longitudinal mode search.

The sections that define the optical cavity were generated by the layer.exe program earlier. By adding the jim.vcsel file to the simulation with the **include** statement, the .sol file can use this information.

Longitudinal Mode Search

Since a DBR cavity is very short, it must be tuned correctly to make the most of the available material gain: the spacer thickness must be adjusted to get the desired emission wavelength and the DBR stacks have to be optimized for this particular wavelength.

To do this, we use the RTG preview we introduced in Sec. 21.2. This previews the round-trip gain of the cavity using the tabulated gain from the .gain file and does a preliminary search for longitudinal modes. The output is shown in the .log file:

```
Longitudinal modes found:
Wavelength (um)    Round-Trip-Gain
 0.895948E+000  0.447049E+000
 0.889152E+000  0.615152E+000
 0.834478E+000  0.997190E+000
 0.785883E+000  0.437601E+000
 0.780535E+000  0.344013E+000
 0.771981E+000  0.330875E+000
 0.760833E+000  0.500443E+000
 0.752508E+000  0.136641E+000
Printing standing wave pattern near the active
  region along with MQW information
End of Printing
Longitudinal standing wave gain
  enhancement factor gfactor_stdwave=   1.70915643886328
```

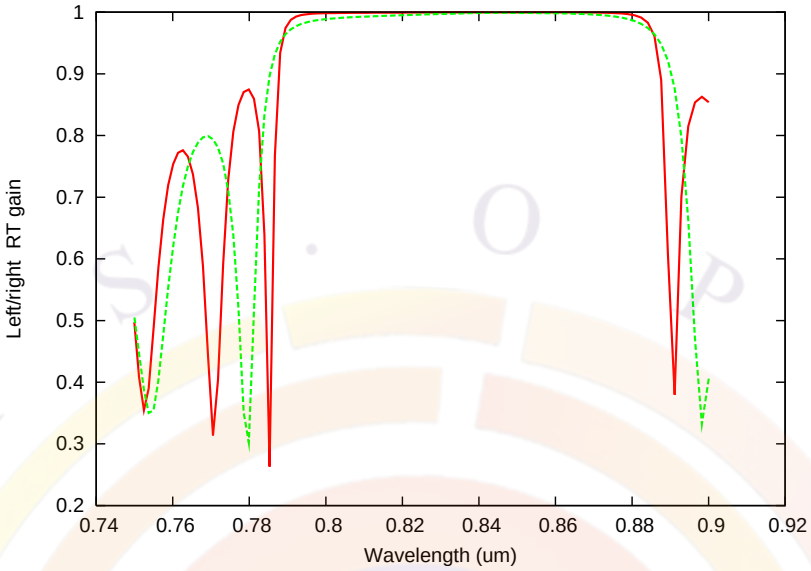


Figure 21.14: Feedback provided by bottom(left) and top(right) DBR stacks

In addition to this information, more data is available by using the **plot_rtgain** statement in the.rtgain file. For example, we can verify the DBR stacks individually by comparing the left (bottom) and right (top) RTG values with respect to the cavity reference point defined earlier. This is shown in Fig. 21.14.

For VCSELs, additional information is available from the standing wave file (.stw) generated by the RTG preview. This information is provided by adding **standing_wave** to the **plot_rtgain** statement.

For example, the standing wave overlap with the quantum well region is shown in Fig. 21.15. In edge-emitting lasers, the standing wave is in the plane of the MQW region so its contribution usually averages out. In a correctly tuned VCSEL cavity however, the standing wave produces significant gain enhancement: the value is shown in the .log printout above.

Bias setup

When applying bias, we follow the guidelines of Sec. 4.2. Since VCSELs have very low threshold currents, the initial voltage scan not only terminates on a current condition, it also initializes the coupled RTG model. That model is turned on by **solve_rtg=yes** in the following **scan** statement.

In previous sections, we tried to emphasize the fact that the RTG initialization should not take place at too low a bias value. This general rule does not apply to VCSELs since the cavity is very short: even with a wide search window, there are few longitudinal modes so it is unlikely that the lasing mode will be missed in the

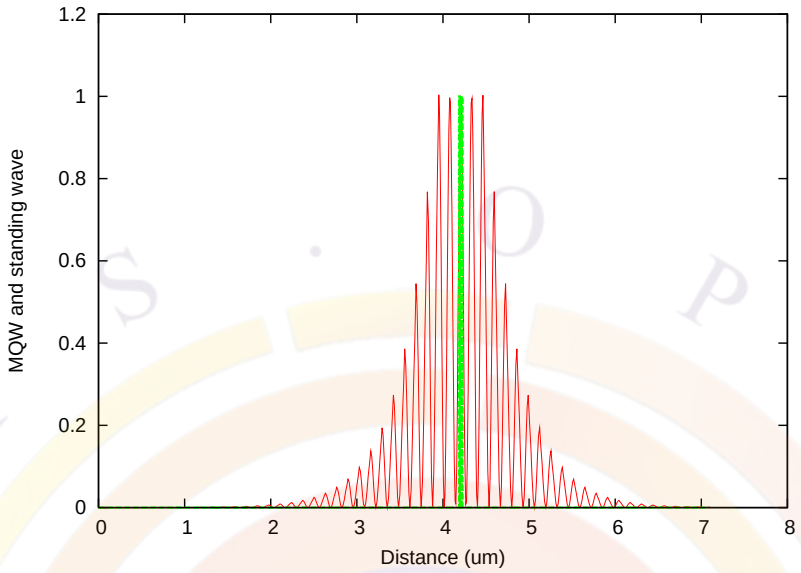


Figure 21.15: Standing wave pattern of VCSEL and overlap with MQW region

below-threshold search. However, we still need to turn on the photon coupling before threshold is reached: the **auto_finish=rtgain** scan must use small bias steps in order not to overshoot the desired bias point.

Post-Processing

Once the simulation has been run, results can be plotted using the following .plt file:

```
$ **************
$file:jim.plt
$ **************
begin_pstprc
plot_data plot_device=postscript
get_data  main_input=jim.sol sol_inf=jim.out &&
  xy_data=(4 4) scan_data=(1 4)
$
plot_scan scan_var=current_1 &&
  variable=rtg_2facet_power_allmode

plot_scan scan_var=current_1 &&
  variable=rtg_left_power_allmode

plot_scan scan_var=current_1 &&
  variable=rtg_right_power_allmode
```

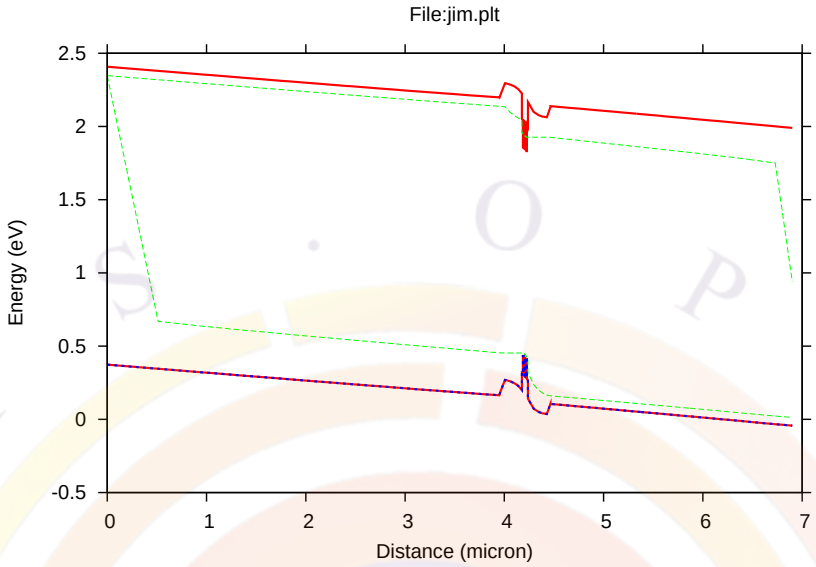


Figure 21.16: Band diagram of VCSEL

```
gain_spectrum variable=rtg_spectrum

$
$ Plot at xy cross sections first
$
plot_1d variable=band from=(0.5 0.0)  to=(0.5 6.9)
plot_2d variable=total_curr grid_sizes=(35, 35)
plot_2d variable=wave_intensity grid_sizes=(35, 35)
end_pstprc
```

Since the simulation uses only a single mesh plane, structural data is plotted using the same commands as for 2D simulations. For example, the band diagram in Fig. 21.16 was generated with the **plot_1d** statement. The (x,y) coordinates are equivalent to (r,z) in the cylindrical system.

For bias-dependent data, the same **plot_scan** statement is used. The 3D nature of the simulation is already taken into account so unlike 2D simulations, the current is plotted in Amperes. We show in Fig. 21.17 the power from the "right" side of the device which corresponds to the top side of the VCSEL: there are fewer DBR layers on that side of the device so it is where most of the output power comes out.

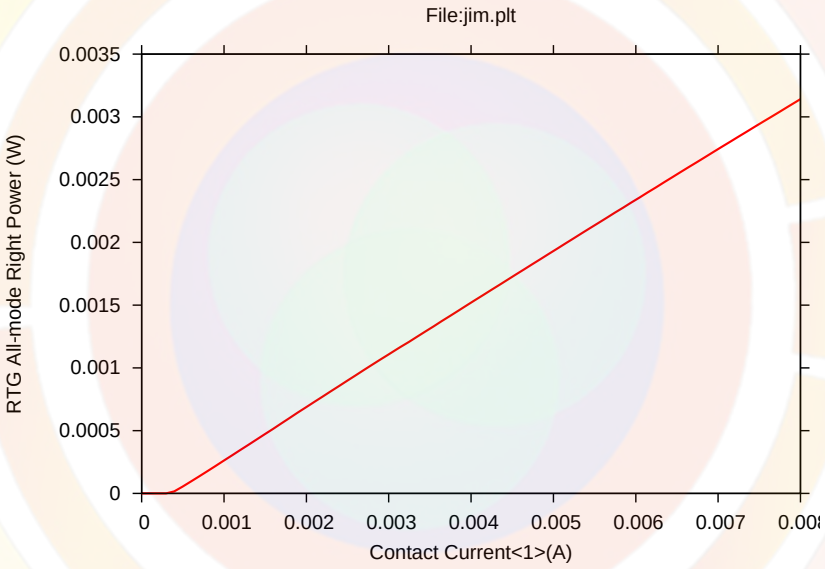


Figure 21.17: Emission power from top facet

PARS OPTICS GROUP

Part V

REFERENCE

PARS OPTICS GROUP

Chapter 22

COMMAND SYNTAX

22.1 Introduction

This chapter explains the syntax of the various statements used in our software. Each statement also supports a number of parameters: they are detailed in the relevant section below.

22.2 2nd_harmonic

parameter	data type	values [defaults]
freq_shd1	real	[0.1e9] (Hz)
freq_shd2	real	[0.1e9] (Hz)
freq_shd3	real	[0.1e9] (Hz)
freq_shd4	real	[0.1e9] (Hz)
freq_shd5	real	[0.1e9] (Hz)
freq_shd_num	intg	[1]

The statement **2nd_harmonic** is used to specify the the modulation frequencies at which the second harmonic distortion ratio is computed.

Parameters

- **freq_shd#** is the modulation frequency of channel number #.
- **freq_shd_num** is the number of modulation channels being considered for calculation.

Examples

```
2nd_harmonic freq_shd1=1.e7 freq_shd2=1.e8 freq_shd3=2.e8 &&
  freq_shd4=6.e8 freq_shd_num=4
```

22.3 3d_amplifier_model

parameter	data type	values [defaults]
input_spec_shape	char	
interband_bulk_absorption	char	
wavelength	real	[1.55] (μm)
input_watt	real	[0.001] (Watt)
linewidth_ghz	real	[1.] (GHz)
more_wavelength_pair	intg	[0]

The statement **3d_amplifier_model** is used to specify the input power and wavelength of the semiconductor optical amplifier (SOA) or photo-absorbing waveguide/modulator. It is assumed that input light is incident from the left facet and output power is from the right facet.

Parameters

- **input_spec_shape** defines the spectral shape of the light input to the SOA or modulator. The input file defined here should consist of a text file with two columns: wavelength (in μm) and normalized intensity.

- **interband_bulk_absorption** is needed to turn on the optical generation due to the SOA/modulator light input when the device consists only of passive bulk regions. This step is necessary due to the separation (in the code) of the optical generation related to optical pumping (i.e. **light_power**) and the generation that occurs as a result of negative stimulated recombination.

  3d_amplifier_model and **waveguide_input** are both designed with an amplifier model in mind and use this second term to amplify/absorb the input light. If no active material is present to trigger the calculation of the stimulated recombination term, the simulation may not behave correctly. Using this parameter substitutes the bulk absorption coefficient for the computed gain/absorption value in the stimulated recombination term.

- **wavelength** is the input light wavelength in μm.

- **input_watt** is the input light power in Watt. This value is scaled by the *light* variable in the **scan** statement. This scaling factor is equal to zero under equilibrium conditions.
- **linewidth_ghz** can be used to quickly define an input spectrum having a flat profile and a certain width. It does not correspond to the linewidth of a laser line input to the device.
- **more_wavelength_pair** defines the number of additional mode peaks in the spectral window being solved. Each peak will be solved self-consistently to get the photon number and amplified spontaneous emission spectra.

Examples

```
3d_amplifier_model wavelength=1.3 input_watt=1.e-4

$ Bias steps omitted ...
scan var=light value_to=1 init_step=1.e-3 &&
  var2=current_1 value2_to=18.5e-3 max_step=0.02 solve_rtg=yes
```

22.4 3d_attachment

parameter	data type	values [defaults]
xy	realx2	[0 0] (μm^2)
to_xy	realx2	[0 0] (μm^2)
zeg_num	intg	[1]
to_zseg_num	intg	[1]

In a conventional 3D setup, z-segments are stacked on top of each other to form a long back to back sequence of 3D objects. The **3d_attachment** command is used to break this sequence and attach any z-segment to any previously defined z-segment: this is used to design complicated 3D structures.

Parameters

- **xy** is a reference point on the segment being attached. This point will be connected to **to_xy** during the attachment process.
- **to_xy** is a reference point on the segment that is the target of the attachment.

- **zeg_num** is the segment being attached.
- **to_zeg_num** is the segment that is the target of the attachment.

Examples

```
z_structure uniform_zseg_from=0 uniform_zseg_to=1  &&
  zseg_num=2 zplanes=4
z_structure uniform_length=0.3  &&
  zseg_num=3 zplanes=4
3d_attachment xy=(0.4 0.4) to_xy=(0.5 0.5) &&
  zseg_num=3 to_zseg_num=2
```

22.5 3d_plane_control

parameter	data type	values [defaults]
z_link_same_mater_only	char	[no]
min_cyl_plane_distance	real	[1.e-4](um)
scale_z_coupling	real	[1.]

The statement **3d_plane_control** is similar to **3d_solution_method** and is used to control the connectivity between stacked mesh planes in a 3D simulation.

Parameters

- **z_link_same_mater_only** would force mesh connectivity only between the same material. This can help avoid undesirable current leakage through different materials.
- **min_cyl_plane_distance** specifies a minimum distance between planes stacked in a cylindrical system. At $r = 0$, the distance between planes is also zero and allowing mesh connectivity can cause problems in some cases.
- **scale_z_coupling** would be used to artificially scale the coupling coefficient (in units of areas) between z-planes.

Examples

```
3d_plane_control z_link_same_mater_only=yes
```

22.6 3d_solution_method

parameter	data type	values [defaults]
3d_flow	char	[yes],no
z_connect	char	yes,[no]
z_connect_method	char	[gpc1],gpc2
xy_compatible_mesh	char	yes,[no]
traveling_wave_model	char	yes,[no]
suprem_insert_mesh	char	yes,[no]
save_z_connect_data	char	yes,[no]
load_z_connect_data	char	yes,[no]
skip_fv_data	char	yes,[no]

The statement **3d_solution_method** is used to define the solution method for a 3D structure. It is related to the statement **z_structure**.

Note that this is used only for devices with more than one mesh plane. Cylindrical devices with a single mesh plane rely on rotational symmetry to define the third dimension and should use the **cylindrical** statement instead.

Parameters

- **3d_flow** is obsolete and used for historical reasons only. All current versions of the software should use **3d_flow**=*yes* when modeling devices with more than one mesh plane. If necessary, the electrical connection between mesh planes can be turned off with **z_connect**.
- **z_connect** is used when modeling multiple mesh planes at once is required (as in the coupled-RTG method of PICS3D) but it is not necessary to allow current to flow between planes. This should not be used in devices where the main current flow direction is perpendicular to the mesh planes.
- **z_connect_method** controls the algorithm used to evaluate the connection between mesh points on neighboring planes; this connection is a function of the overlap between the polygon boxes surrounding each mesh points. Two methods are available: *gpc1* which favors speed and *gpc2* which favors accuracy.
- **xy_compatible_mesh** can be used to simplify the problem when all mesh planes use the same identical layout. The software will alter the way is positions planes at z-segment boundaries and will not calculate the mesh point overlap between planes. **Warning: using this option when the planes are not compatible is a serious error so this option should be used with care.**

- **traveling_wave_model** is used in 3D amplifier and SLED models in PICS3D. It is used to calculate the round-trip propagation based on power flux rather then optical fields.

- **suprem_insert_mesh** is used only when importing 3D mesh data from CSUPREM where additional bent mesh planes have been added.

- **save_z_connect_data** instructs the software to save the connection data between mesh planes. In subsequent simulations, **load_z_connect_data** may be used to speed up the simulation by re-using these results.

- **skip_fv_data** instructs the software not to store filling volume data. This statement should be used in simulations where multiple 2D devices are solved using a single 3D mesh but the devices are not actually monolithically integrated.

Examples

```
3d_solution_method 3d\_flow=yes
```

This setting is used for full 3D simulations in APSYS.

```
3d_solution_method 3d\_flow=yes z_connect=no
```

In PICS3D the above would mean decoupling all plane interactions except for the optical wave propagation (coupled round-trip gain).

22.7 3drayplot_angle

parameter	data type	values [defaults]
data_file	char	void
smooth	char	[yes] no
view_xrot	real	[0.]
view_zrot	real	[0.]
theta_range	realx2	[0. 180.]
phi_range	realx2	[0. 360.]
smooth_width	real	[8.](degree)

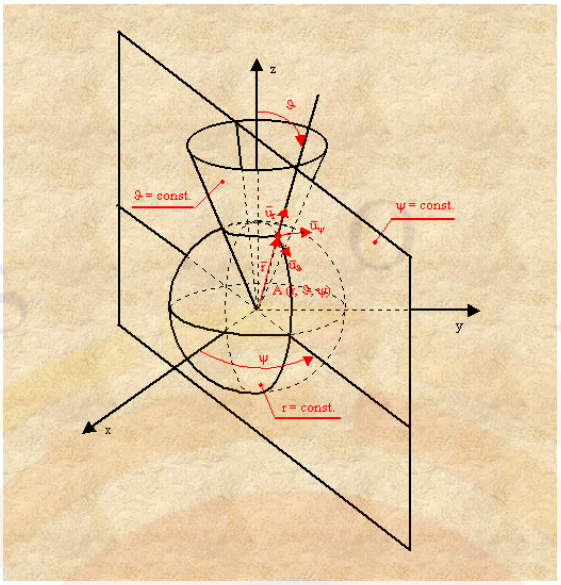


Figure 22.1: The spherical coordinates used for raytracing commands

3drayplot_angle is used after a 3D raytracing simulation to plot the emitted power distribution. The power is shown in the form power(phi,theta) where the two angles (in degrees) are defined in the spherical coordinate system of Fig. 22.1.

In order to save time, it is common to use separate .plt files to do the raytracing simulation and plot its results. If that is the case and symmetry conditions have been enforced with **set_3dray_mirror** during the raytracing simulation, then this statement should be reused to guarantee the symmetry of the plot.

Note that 1D cuts of this data are available with the **3d_rayplot_phi** and **3d_rayplot_theta**. This should be used when plotting results from a 2D simulation since the raytracing program will only emit rays in a plane: **3drayplot_angle** cannot be used in that case.

Parameters

- **data_file** copies the plot results to the specified data file.
- **view_xrot** rotates the plot around the phi-axis.
- **view_zrot** rotates the plot around the power-axis.
- **theta_range** narrows the area plot within theta angle range.
- **phi_range** narrows the area plot within phi angle range.
- **smooth** indicates if data smoothing is performed. This is usually recommended since there are only a finite number of rays in a given simulation.

- **smooth_width** is the width of the data broadening/smoothing. A Gaussian function is used for smoothing and this parameter is the standard deviation. The larger the standard deviation, the smoother the data curve appears to be.

Examples

```
3drayplot_angle phi_range=(180. 360.)
```

22.8 3drayplot_bias

parameter	data type	values [defaults]
variable	char	[transmitted]
relative	char	[no] yes
bias_variable	char	[current]voltage
data_file	char	[void]
bias_electrode	intg	[1]

3drayplot_bias is used to plot bias-dependent results from a 3D raytracing simulation.

In order to save time, it is common to use separate .plt files to do the raytracing simulation and plot its results. If that is the case and symmetry conditions have been enforced with **set_3dray_mirror** during the raytracing simulation, then this statement should be reused to scale the results properly.

Parameters

- **variable** is the physical variable to be plotted. It takes one of the following values:

variable	description
total_source	Total power of the emission source.
transmitted	Total transmitted power.
semicond_absorb	Power absorbed in LED, excluding losses due to metal contacts.
contact_absorb	Power absorbed in LED due to metal contacts.
upper_half	Power emitted towards the upper half semi-sphere.
lower_half	Power emitted towards the lower half semi-sphere.
left_half	Power emitted towards the left half semi-sphere.

right_half	Power emitted towards the right half semi-sphere.
front_half	Power emitted towards the front half semi-sphere.
back_half	Power emitted towards the back half semi-sphere.
upper_side	Power emitted through the upper side facet(s).
lower_side	Power emitted through the lower side facet(s).
left_side	Power emitted through the left side facet(s).
right_side	Power emitted through the right side facet(s).
front_side	Power emitted through the front side facet(s).
back_side	Power emitted through the back side facet(s).

- **relative** indicates whether relative or absolute values are being plotted.
- **bias_variable** is the bias variable to appear in the horizontal axis of the plot.
- **data_file** copies the plot results in the specified data file.
- **bias_electrode** is the electrode/contact number of parameter **bias_variable**.

Examples

```
3drayplot_bias variable=transmitted bias_variable=current
```

The above statement plots the total transmitted LED power versus the current from electrode number 1.

22.9 3drayplot_phi

parameter	data type	values [defaults]
data_file	char	[void]
smooth	char	[yes] no
polar	char	[yes] no
theta	real	[0.] (degrees)
smooth_width	real	[8.](degree)

3drayplot_phi is used after a 3D raytracing simulation to plot the emitted power distribution. The power is shown as a 1D of power(phi,theta) at a specific theta

value. Both angles are in degrees and are defined in the spherical coordinate system of Fig. 22.1.

In order to save time, it is common to use separate .plt files to do the raytracing simulation and plot its results. If that is the case and symmetry conditions have been enforced with **set_3dray_mirror** during the raytracing simulation, then this statement should be reused to guarantee the symmetry of the plot.

Parameters

- **theta** is the fixed-theta angle for the plot of power versus phi. It ranges from 0 to 180 degrees.
- **data_file** copies the plot results to the specified data file.
- **polar** determines whether or not the results are shown in polar coordinates.
- **smooth** indicates if data smoothing is performed. This is usually recommended since there are only a finite number of rays in a given simulation.
- **smooth_width** is the width of the data broadening/smoothing. A Gaussian function is used for smoothing and this parameter is the standard deviation. The larger the standard deviation, the smoother the data curve appears to be.

Examples

```
3drayplot_pho theta=0
```

22.10 3drayplot_project

parameter	data type	values [defaults]
view_xrot	real	[0.]
view_zrot	real	[0.]
distance	real	[1.]
radius	real	[2.]
position_phi	real	[270.]
position_theta	real	[90.]

3drayplot_project is a post-processing statement for a 3D raytracing simulation. It shows the power distribution as projected onto a screen some distance away from

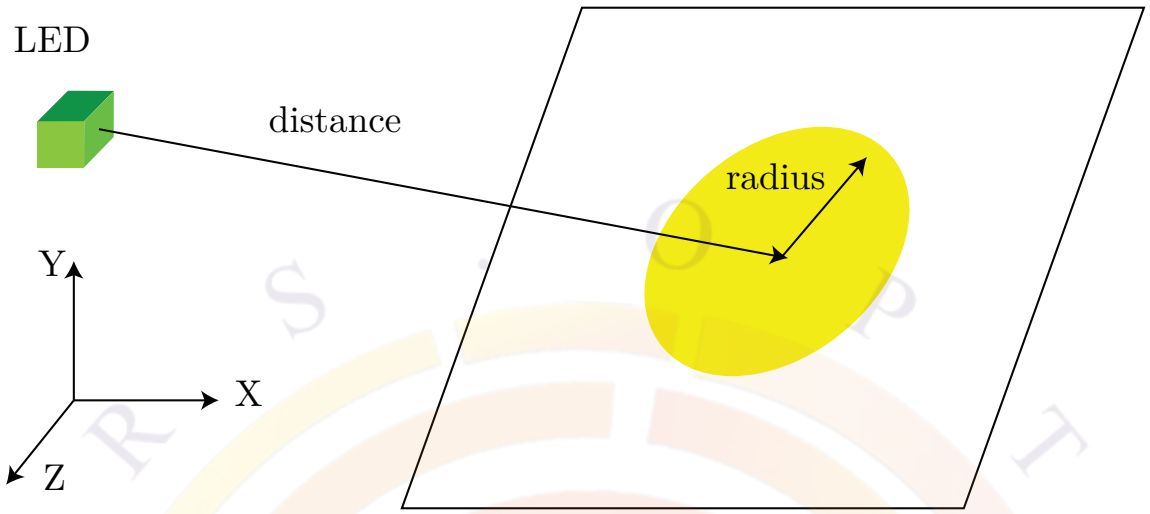


Figure 22.2: The variables **distance** and **radius** of **3drayplot_project** statement

the LED as shown in Fig. 22.2. Position on the screen is determined by the angles phi and theta (see Fig. 22.1) of a spherical coordinate system centered on the LED.

Parameters

- **view_xrot** rotates the plot around the phi-axis.
- **view_zrot** rotates the plot around the power-axis.
- **distance** is the distance between the screen center and the center of the LED.
- **radius** is the radius of the circle area of projection.
- **position_phi** is a screen center position defined by the angle phi.
- **position_theta** is a screen center position defined by the angle theta.

The angle values are given in degrees. The distance and radius are in units of meter.

Examples

```
3drayplot_project radius=3. position_phi=90.
```

22.11 3drayplot_spectrum

parameter	data type	values [defaults]
variable	char	[transmitted]
data_file	char	
relative	char	yes,[no]

3drayplot_spectrum is used after a 3D raytracing simulation to plot various wavelength-dependent results.

Parameters

- **variable** is the variable being plotted. The definitions are the same as in 3drayplot_bias.
- **data_file** copies the plot results to the specified data file.
- **relative** is used to show plots using absolute power units or as a percentage.

Examples

```
3drayplot_spectrum variable=transmitted relative=yes
```

22.12 3drayplot_surfpower

parameter	data type	values [defaults]
data_file	char	[void]
smooth	char	[yes] no
side	char	-x,+x,-y,+y,-z,+z
x_range	realx2	[-9999. , -9999.] (um)
y_range	realx2	[-9999. , -9999.] (um)
smooth_width	real	[1.](um)

3drayplot_surfpower is used to plot the surface emission power in a 3D ray tracing simulation.

Parameters

- **data_file** copies the plot results to the specified data file.
- **side** determines which side is to be plotted.
- **smooth** indicates if data smoothing is performed.
- **smooth_width** is the width of the data broadening/smoothing. A Gaussian function is used for smoothing and this parameter is the standard deviation. The larger the standard deviation, the smoother the plot.
- **x_range** and **y_range** can be used to zoom in on a particular section of the plot. Note however that the original data is still limited by **surface_power_grid** in **do_raytrace_3d**.

Examples

```
3drayplot_surfpower side=+y
```

22.13 3drayplot_theta

parameter	data type	values [defaults]
data_file	char	[void]
smooth	char	[yes] no
polar	char	[yes] no
phi	real	[90.] (degrees)
smooth_width	real	[8.](degree)

3drayplot_theta is used after a 3D raytracing simulation to plot the emitted power distribution. The power is shown as a 1D of power(phi,theta) at a specific phi value. Both angles are in degrees and are defined in the spherical coordinate system of Fig. 22.1.

In order to save time, it is common to use separate .plt files to do the raytracing simulation and plot its results. If that is the case and symmetry conditions have been enforced with **set_3dray_mirror** during the raytracing simulation, then this statement should be reused to guarantee the symmetry of the plot.

Parameters

- **phi** is the fixed-phi angle for the plot of power versus theta. It ranges from 0 to 360 degrees.
- **data_file** copies the plot results to the specified data file.
- **polar** determines whether or not the results are shown in polar coordinates.
- **smooth** indicates if data smoothing is performed. This is usually recommended since there are only a finite number of rays in a given simulation.
- **smooth_width** is the width of the data broadening/smoothing. A Gaussian function is used for smoothing and this parameter is the standard deviation. The larger the standard deviation, the smoother the data curve appears to be.

Examples

```
3drayplot_theta phi=90
```

22.14 a_bar

The material statement **a_well** is an active layer macro statement used for zincblende materials to define the total hydrostatic deformation potential (eV) in the barrier of a quantum well. This value is split between the conduction and valence bands ($a_c + a_v = a$). The exact ratio between these values is determined by **av_over_a** in **active_reg**.

Please note that for wurtzite, a more complex strain model is required so this parameter is ignored. See **ac_bar** for details.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.15 a_bulk

This statement is obsolete. Please see **ac_bulk** for details.

22.16 a_well

The material statement **a_well** is an active layer macro statement used for zincblende materials to define the total hydrostatic deformation potential (eV) in a quantum well or bulk active region. This value is split between the conduction and valence bands ($a_c + a_v = a$). The exact ratio between these values is determined by **av_over_a** in **active_reg**.

Please note that for wurtzite, a more complex strain model is required so this parameter is ignored. See **ac_well** for details.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.17 absorption

This material statement is used to defined the optical absorption or loss (in units of 1/m) in passive materials of the device. For active materials, this value is overridden by the internal gain calculations.

The default macro setting for this parameter is zero which means that a default background loss will be applied for this material. This background loss will be determined either by a hard-coded value in the software or from the **backg_loss** value in **init_wave** or **set_wavelength**. Note that conversely, a non-zero value of **absorption** will override the background loss for this material.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.18 ac_bar

ac_bar is an active layer macro statement that is used for wurtzite materials [62]. It defines the hydrostatic deformation potential (in eV) applied to the conduction band in the barrier of a quantum well. This parameter is applied to the lateral strain coefficient ϵ_{xx}; see **acz_bar** for the matching coefficient which is applied to ϵ_{zz}.

This value is used to position the band edges based on the value of **band_offset** and the strained bandgap difference, as discussed in Sec. 10.1

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.19 ac_bulk

The material statement **ac_bulk** is a passive macro statement used to define the hydrostatic deformation potential (eV) applied to the conduction band in wurtzite bulk regions[62].

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

Important Note

This parameter is used internally to solve the bulk Wurtzite Hamiltonian and compute band edges as part of the fitting procedure for the effective masses in Sec. 13.1. ***However, it is not used to actually shift the conduction band that is used for transport.***

Users who wish to see strain effects affect the position of bulk band edges are advised to modify the **affinity** statement to include the hydrostatic shift. It may also be necessary to use **bulk_strain_exist**=*yes* in **wurtzite_offset_model**.

22.20 ac_light

parameter	data type	values [defaults]
output_file	char	[void]
current_distr	char	[no] yes
versus_bias	char	yes [no]
log_freq1	real	[6.]
log_freq2	real	[12.]
freq_point	intg	[20]
scanline	intg	[-9999]
scan_num	intg	[-9999]

---

ac_light is a post-processing statement used to apply a small AC light signal to a photosensitive device being simulated. It is otherwise similar to **ac_voltage** and the required AC data must be gathered during the main simulation with the **more_output** statement.

Parameters

- **output_file** is the output file where AC small signal analysis data for this statement will be saved.
- **current_distr** is used to indicate if AC current distribution data is to be saved in the output file for later use.
- **versus_bias** indicates if the plot is against bias (or scanned variable) or vs. the modulation frequency.
- **log_freq1** is the starting frequency in log10 scale for the frequency scan.
- **log_freq2** is the ending frequency in log10 scale for the frequency scan.
- **freq_point** is the number of frequency points for the AC scan.
- **scanline** is used only when **versus_bias**=*yes*. It is the same as in **plot_scan**.
- **scan_num** equals **scanline**-1.

Examples

```
ac_light output_file=testlight.ac log_freq1=6.  log_freq2=12. &&
  freq_point=20
```

22.21 ac_parameters

parameter	data type	values [defaults]
output_file	char	[void]
log_scan	char	[yes] no
versus_bias	char	yes [no]
scan_label	char	[]
modulate_gain	char	[no], yes
mixmode	char	[no], yes
log_freq1	real	[6.]
log_freq2	real	[12.]
scale_lit	real	[1.]
scale_curr	real	[1.]
input_contact	intg	
output_contact	intg	
freq_point	intg	[20]
scanline	intg	[]
scan_num	intg	[]

ac_parameters is used to set up the AC analysis for a 2-port high frequency model. Once the input and output contacts are defined, the analysis is performed with an AC voltage signal at the input and output contacts; the resulting current responses are saved in a data file so that AC characteristics (such Y- and S-parameters) can be plotted afterwards.

This statement is used in .plt files as a post-processing statement after the DC results are obtained. To use this statement, **more_output** should be used in the .sol file to instruct the program to save the necessary extra AC data while performing the DC simulation.

To apply the AC bias to a single electrode, use **ac_voltage**.

Parameters

- **output_file** is the output file where AC small signal analysis data for this statement will be saved for later use. The simulator will assign an internal file name if this parameter is not defined.

- **log_scan** indicates whether frequency points are spaced equally on a logarithmic or linear scale.

- **versus_bias** indicates if the plot is against bias (or scanned variable) or vs. the modulation frequency.

  In the first case, data sets at multiple bias values are used so the input range for the AC analysis is defined by the **scan_data** parameter of the preceding **get_data** statement. A single frequency value is used for all bias points.

  In the latter case, a single bias value is used so the input range is instead set by **xy_data** in the same command. Multiple frequency points are used to analyse this single bias point. Note that the equivalence between the data set number and the exact voltage/current bias value is shown in the .log and .sol.msg files for each simulation.

- **log_freq1** is the starting frequency in log10 scale for the frequency scan; the end of the range is given by **log_freq2**. If **versus_bias**=*yes*, these two values should be identical to set the single frequency at which the AC analysis is done.

- **scale_lit** is applicable for laser simulation and scales the light power to account for the symmetry of the device; **scale_lit** is similar but scales the laser current. Previous versions of LASTIP assumed symmetry and set both of these values to 2 by default; see **plot_scan** for more information.

- **input_contact** is the input contact in a 2-port system; similarly, **output_contact** is the output contact. For example, in the common emitter configuration of a BJT, the input contact would be the base while the output contact would be the collector.

  Please note that during the AC analysis, all other contacts are assumed to have a zero AC voltage bias unless otherwise specified or controlled in some way. To define a floating electrode that can still supply a DC bias, an external circuit with a large inductance may be used.

- **freq_point** is the number of frequency points for the AC scan when **versus_bias**=*no*.

- **scanline** is used only when **versus_bias**=*yes*; the definition is the same as in **plot_scan**. This parameter filters the data sets defined in **get_data** so that only some of them get processed in the AC analysis.

  Alternatively, **scan_num** = **scanline**-1 may be used to select data sets. A previously-defined scan label may also be used with **scan_label**.

- **modulate_gain** is used in laser simulations to include the effects of the photon rate equation on the AC response.

- **mixmode** is used in mixed-mode (**minipsice**) simulations to include the effects of external circuit elements in the AC response.

Examples

To plot y, s or h-parameters vs. the input voltage at 1 MHz:

```
$ For purpose of demonstrating the plotting,
$ let us plot y parameters assuming input=output=electrode #1
get_data  main_input=camel.sol sol_inf=camel.out &&
  scan_data=(3 13)

ac_parameters versus_bias=yes log_freq1=6.  log_freq2=6. &&
  input_contact=1 output_contact=1 freq_point=2 scanline=3

plot_ac_parameters parameter_type=y smith_chart=no
```

For a frequency analysis between 1 kHz and 10 GHz for data set #6:

```
$ contact 1 = collector
$ contact 2 = emitter
$ contact 3 = base
$
$ AC results
get_data  main_input=bi_mi.sol sol_inf= bi_mi.out &&
  xy_data=(6  6)

ac_parameters log_freq1=3.  log_freq2=10. &&
  freq_point=20 input_contact=3 output_contact=1

plot_ac_parameters parameter_type=y smith_chart=no

plot_ac_parameters parameter_type=s smith_chart=no

plot_ac_parameters parameter_type=s smith_chart=yes
```

22.22 ac_sparse_solver

parameter	data type	values [defaults]
use_mf	char	[yes],no

sparse_eigen_solver is used to control the sparse solver used in the .plt file for AC small signal analysis. The default setting is usually optimal and should be used unless there is a convergence problem.

Parameters

- **use_mf** is used to indicate whether multi-frontal solver will be used.

Examples

```
ac_sparse\_solver use_mf=yes
```

22.23 ac_voltage

parameter	data type	values [defaults]
output_file	char	[void]
current_distr	char	yes, [no]
log_scan	char	[yes], no
versus_bias	char	yes, [no]
scan_label	char	[]
modulate_gain	char	[no], yes
mixmode	char	[no], yes
log_freq1	real	[6.]
log_freq2	real	[12.]
scale_lit	real	[1.]
scale_curr	real	[1.]
contact_num	intg	
freq_point	intg	[20]
scanline	intg	[]
scan_num	intg	[]

ac_voltage is used to apply a small AC bias voltage on one of the electrodes; the resulting current responses are saved in a data file so that AC characteristics can be plotted afterwards.

This statement is used in .plt files as a post-processing statement after the DC results are obtained. To use this statement, **more_output** should be used in the .sol file to instruct the program to save the necessary extra AC data while performing the DC simulation.

For devices with more than two electrodes, **ac_parameters** may be used to perform a similar small-signal analysis using a two-port model.

Parameters

- **output_file** is the output file where AC small signal analysis data for this statement will be saved for later use. The simulator will assign an internal file name if this parameter is not defined.

- **current_distr** instructs the software to output the AC current distribution instead of data related to the AC current on the electrodes. Since this current flow is shown vs. x-y coordinates, the AC analysis can only be done at a single frequency rather than over a range.

- **log_scan** indicates whether frequency points are spaced equally on a logarithmic or linear scale.

- **versus_bias** indicates if the plot is against bias (or scanned variable) or vs. the modulation frequency.

  In the first case, data sets at multiple bias values are used so the input range for the AC analysis is defined by the **scan_data** parameter of the preceding **get_data** statement. A single frequency value is used for all bias points.

  In the latter case, a single bias value is used so the input range is instead set by **xy_data** in the same command. Note that the equivalence between the data set number and the exact voltage/current bias value is shown in the .log and .sol.msg files for each simulation.

- **log_freq1** is the starting frequency in log10 scale for the frequency scan; the end of the range is given by **log_freq2**. If **versus_bias** or **current_distr** are set to *yes*, these two values should be identical to set the single frequency at which the AC analysis is done.

- **scale_lit** is applicable for laser simulation and scales the light power to account for the symmetry of the device; **scale_lit** is similar but scales the laser current. Previous versions of LASTIP assumed symmetry and set both of these values to 2 by default; see **plot_scan** for more information.

- **contact_num** is the input contact for the AC voltage; all other contacts are assumed to have a zero AC voltage bias unless controlled in some other way.

  Please note that during the AC analysis, all other contacts are assumed to have a zero AC voltage bias unless otherwise specified or controlled in some way. To define a floating electrode that can still supply a DC bias, an external circuit with a large inductance may be used.

- **freq_point** is the number of frequency points for the AC scan when **versus_bias**=*no*.

- **scanline** is used only when **versus_bias**=*yes*; the definition is the same as in **plot_scan**. This parameter filters the data sets defined in **get_data** so that only some of them get processed in the AC analysis.

  Alternatively, **scan_num** = **scanline**-1 may be used to select data sets. A previously-defined scan label may also be used with **scan_label**.

- **modulate_gain** is used in laser simulations to include the effects of the photon rate equation on the AC response.

- **mixmode** is used in mixed-mode (**minipsice**) simulations to include the effects of external circuit elements in the AC response.

  If this setting is used then the input source for the AC voltage are defined in the SPICE circuit file and the **contact_num** setting is ignored: all TCAD electrode voltages become unknowns that must be solved for.

Examples

This example shows analysis vs. frequency for data set #2. The real and imaginary parts of the AC current for electrode #1 is used to obtain G-f and C-f plots, respectively.

```
$ AC analysis at max. voltage
get_data  main_input=camel.sol sol_inf=camel.out &&
  xy_data=(2 2)

$ AC analysis vs. freq.
ac_voltage log_freq1=6. log_freq2=10. contact_num=1
plot_ac_curr variable=capacitance_1
plot_ac_curr variable=conductance_1
```

We wish to emphasize that in the above, capacitance is obtained directly from the AC electrode current ($I = VY$, with $Y = G + j\omega C$). Different electrodes may have different AC current values, in which case the capacitance values will also be different; this may occur in current-blocking device simulations with numerical accuracy difficulties or in devices where multiple electrodes split the current. This definition of capacitance is thus similar to what is obtained experimentally and lumps together all effects which add a "lag" to the signal response including, but not limited to, junction capacitance and carrier transit time in thick layers.

This second example is the same as above except that instead of looking at data from the electrodes, the physical distribution of AC current inside the device is shown.

```
$ AC analysis at max. voltage
get_data  main_input=camel.sol sol_inf=camel.out &&
  xy_data=(2 2)

$ Physical distribution of AC current @ 1 MHz
ac_voltage current_distr=yes log_freq1=6. log_freq2=6.
plot_1d_ac_curr variable=elec_curr_y imag_part=no from=(0.5 0.) to=(0.5  2.3)
plot_1d_ac_curr variable=hole_curr_y imag_part=no from=(0.5 0.) to=(0.5  2.3)
```

The third example shows C-V and G-V plots at 1 MHz. The electrode used to input the AC signal (#1) serves as the horizontal axis of the C-V plot which means that it was also used as the original DC bias electrode.

```
$ We plot capacitance-voltage and conductance-voltage here at 1 MHz
$ To generate reasonably smooth AC vs. bias plots, please save a sufficient
$ number of data sets.  AC analysis plots cannot include equilibrium data.
get_data  main_input=camel.sol sol_inf=camel.out &&
  scan_data=(3 13)

ac_voltage log_freq1=6. log_freq2=6. contact_num=1 &&
  freq_point=2 versus_bias=yes scanline=3

set_xydata_for_scan scan_var=voltage_1
plot_ac_curr variable=capacitance_1
plot_ac_curr variable=conductance_1
```

Our final example is a mixed-mode AC simulation, new to v.2015 of the software.

```
get_data  main_input=tt.sol &&
  sol_inf=tt.out xy_data=[  2  2]

ac_voltage mixmode=yes log_freq1=6.  log_freq2=10 &&
      current_distr=no freq_point=20
plot_ac_curr variable=t_real_2
plot_ac_curr variable=d_real_2
plot_ac_curr variable=capacitance_2

plot_ac_minispice variable=voltage node=2 element=void
plot_ac_minispice variable=current node=2 element=Dtcad1
plot_ac_minispice variable=current node=2 element=R2
plot_ac_minispice variable=current node=2 element=Dtcad1 imag_part=yes
plot_ac_minispice variable=current node=2 element=R2 imag_part=yes
```

As in the first example, capacitance is extracted from the electrode's (#2) current response. However this time, the input AC voltage is located outside the device and is applied to a SPICE node, as shown in the following circuit file:

```
# test circuit
R1 0 1 1k
Vcc 1 0  0.65V  AC_INPUT
R2 1 2 1k
R3 0 2 0.5k
Dtcad1 2 0  tcadmesh
.END
```

22.24 ac_well

ac_well is an active layer macro statement that is used for wurtzite materials [62]. It defines the hydrostatic deformation potential (in eV) applied to the conduction band in a quantum well or in a bulk active region. This parameter is applied to the lateral strain coefficient ϵ_{xx}; see **acz_well** for the matching coefficient which is applied to ϵ_{zz}.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

Important Note

In quantum wells, this parameter is used, among other things, to compute the band edges based on the value of **band_offset** and the strained bandgap difference, as discussed in Sec. 10.1.

However, in a bulk active region the position of the conduction band is inherited from the underlying passive macro. As such, this parameter ***is not used to actually shift the conduction band that is used for transport***. This parameter is only used internally as part of the band fitting procedure for the effective masses in Sec. 13.1.

Users who wish to see strain effects affect the position of bulk active band edges are advised to modify the **affinity** statement (in the underlying passive macro) to include the hydrostatic shift. It may also be necessary to use **bulk_strain_exist**=*yes* in **wurtzite_offset_model**.

22.25 active_macro_override

parameter	data type	values [defaults]
complex_macro	char	[yes]

The statement **active_macro_override** is used to control how parameters that are defined in both active and passive macros co-exist in the simulation. For example, bandgap is necessary for both the gain calculations (active macro, can be used in stand-alone gain preview mode) and the electrical solver (passive macro, aligns the band edges). By default, the passive macro is always overridden by the active macro.

In the conventional active macros, passive parameters belonging to both the quantum well and barrier may be overridden. More complex situations can occur when using complex MQW macros where the distinction between barrier and well is fuzzier so this statement allows for a convenient way of going back to a simpler model and disabling the active macro override.

See also **use_bulk_affinity**, **use_bulk_bandgap** and **use_bulk_property** for a partial disabling of the active macro override system.

Parameters

- The **complex_macro** flag turns on (default) or off the override of passive material parameters by complex MQW macros.

22.26 active_reg

parameter	data type	values [defaults]
mode	char	[te],tm
broadening	char	[lorentzian], landsberg, landsberg_2
axial_approx	char	[yes],no
valence_mixing	char	[no],yes
analytical_recomb	char	[no],yes
bandgap_renorm	char	[no],yes
gain_coulomb	char	[no],yes
exciton	char	[no],yes
coulomb_dim	char	[quasi2d],2d,3d
coulomb_screen	char	[yes],no

franz_keldysh	char	[no],yes
mater_label	char	
bulk_active	char	[no],yes
strained_bulk_active	char	[no],yes
uniaxial_strain	char	[no],yes
exch_coef	real	[0.0] (eV/m)
tau_scat	real	[2.e-13] (sec)
a_scat	real	[0.0] ($eVm^{3/2}$)
thickness	real	[0.05] (μm)
ncarr_loss	real	[0.0] (m^2)
pcarr_loss	real	[0.0] (m^2)
dip_factor	real	[1.]
k_range	real	[0.06]
tau_bkg	real	[5.e-13](sec)
expo_mix	real	[0.01] (eV)
active_loss	real	[0.] (1/m)
av_over_a	real	[0.33333]
k_range_wurt	real	[0.1]
diel_av	real	[13.1]
plasma_coeff	real	[1.0]
sommerfeld_fac	real	[1.7]
rmesh_fd	real	[0.2]
two_photon_loss	real	[0] (m^2)
two_photon_carr	real	[0] (m^5)
mater	intg	[1]
qw_print	intg	[1]
level_srch	intg	[30]
fd_mesh	intg	[80]
qw_plane	intg	[100]
coulomb_int_acc	intg	[6]
embedded_structure	intg	[0],1,2

The statement **active_reg** is used to define active regions where optical gain/absorption is calculated according to the usual formulas (i.e. Fermi's golden rule). This statement should be used in conjunction with **get_active_layer** to load active macro parameters such as composition.

active_reg also defines various gain calculation settings on a per-material basis. In practice, users who wish to alter these settings in their simulation want to change

them for all active regions in the device: this means using **set_active_reg** is often preferred to change these settings. With a few exceptions, the two statements share the same set of input parameters.

For wurtzite structures, additional relevant parameters are controlled by the **modify_wurtzite** statement.

Parameters

- **mode** defines whether TE or TM modes are assumed in the gain calculation. The spontaneous emission is not affected by this choice because spontaneous emission is randomly polarized.
- **broadening** defines the type of energy level broadening used in the quantum well gain calculation. Bulk optical gain is not affected by this option since this effect is much smaller than in quantum wells. The following choices are available:
  - *lorentzian* uses the ideal Lorentzian function with constant half-width Γ over all frequencies. Refer to Sec. 8.3.3 for details.
  - *landsberg* uses a broadening function of the Landsberg type, i.e., a Lorentzian function with an energy and carrier concentration dependent Γ. Note that this model does not include any broadening for optical gain below transparency (or loss). Therefore the gain spectrum contains step-like jumps in the negative gain region. Refer to Sec. 8.3.4 for details.
  - *landsberg_2* is a modified version of the ideal Landsberg model that combines useful features of the previous two options. The original Landsberg model assumes no broadening below transparency and results in sharp edges in the quantum well spectrum. The modified Landsberg model replaces the spectrum below transparency by a Lorentzian broadened gain with a different broadening tau equal to **tau_bkg** (for tau of background). Above the transparency, the gain function is a mixture of Landsberg broadened gain and Lorentzian broadened gain controlled by **expo_mix**. Physically the background Lorentzian broadening can be considered as contributions from scattering mechanisms other than carrier-carrier interaction.
- **axial_approx** turns on/off the axial approximation in k.p band dispersion calculation.
- **valence_mixing** turns on/off the valence mixing model in the k.p quantum well theory. The 4×4 Luttinger-Kohn Hamiltonian is solved for the quantum well system based on the formulas of S.L. Chuang. For the wurtizte structure system, a 6×6 Hamiltonian is used: refer to Chap. 13 for details.

- **analytical_recomb** is used to indicate if the analytical recombination term Bnp defined with the **radiative_recomb** statement in the passive macro is used to model the spontaneous recombination term. In bulk active regions, this is always the case but the default setting for quantum wells is to use the integral of the spontaneous recombination spectrum.

- **override_barrier** is here for historical reasons. It determines whether active layer macros from the quantum well can override passive layer macros for the barrier. This is the default behavior in newer versions of the software although setting this to *no* can recover the behavior of older versions.

- **bandgap_renorm** turns on the internal bandgap renormalization model when considering many-body Coulomb interaction. If this is used, **exch_coef** should be set to zero to avoid double-counting the bandgap renormalization term.

- **gain_coulomb** turns on many-body Coulomb interactions in the optical gain model. The shape of the gain and spontaneous emission spectrum may become more symmetric as a result of this model and the magnitude of the gain/PL spectrum may be enhanced.

- **exciton** turns on the exciton model.

- **coulomb_dim** specifies the dimension of the Coulomb potential used for the derivation of the exciton binding energy. The most appropriate for excitons confined to thin (but not zero thickness) semiconductor layers is the *quasi2d* model. *3d* is a normal Coulomb potential without any spatial confinement; *2d* is a Coulomb potential acting only in a 2-dimensional plane of charges (zero thickness) and is only used here for comparative studies.

- **coulomb_screen** means two options of Coulomb potential model for the purpose of finding binding energy of an exciton with screening ("yes") or without screening ("no").

- **franz_keldysh** turn on/off the Franz-Keldysh effect for bulk active layer.

- **exch_coef** is the bandgap renormalization coefficient due to exchange effects:

$$\Delta E_g = A_x \left(\frac{n+p}{2} \right)^{1/3} \tag{22.1}$$

  The value of this parameter sets A_x; a typical value is $\sim 10^{-10}$ eV.m. Care should be taken not to double-count this effect if **bandgap_renorm** and **gain_coulomb** are used.

- **tau_scat** and **a_scat** are parameters defining the gain broadening due to carrier scattering. For the Lorentzian model, this defines an effective broadening

constant:

$$\frac{\hbar}{\tau} = \frac{\hbar}{\tau_{scat}} + a_{scat}\sqrt{\frac{n+p}{2}} \tag{22.2}$$

For the Landsberg model, we simply have:

$$\Gamma_0 = \frac{\hbar}{\tau_{scat}} \tag{22.3}$$

Refer to Sec. 8.3 for more details.

- **thickness** defines the thickness of the active quantum well and is used to compute the subband structures of the quantum well in the absence of mesh information. Note that this parameter should be consistent with the device geometry given in the .geo input file. This parameter is not used for bulk active regions.
- **ncarr_loss** and **pcarr_loss** are the loss coefficients due to free carrier absorption for n and p carriers. **pcarr_loss** can also be used to model the intervalence band absorption effects because it is also proportional to the p-carrier densities. Note that this only applies for active regions. For passive regions, the equivalent statements **elec_carr_loss** and **hole_carr_loss** should be used in the material macro.
- **dip_factor** is an artificial scaling factor for the dipole moment of the interband optical transition.
- **k_range** specifies the k-space range within which the in-plane effective mass is fitted from k.p theory for a strained material. It is given in units of $2\pi/a_0$, where a_0 is the lattice spacing of the substrate material. Large values tend to average out the details of the k.p band structure and produce lower optical gain. Small values tend to over-estimate the optical gain and run the risk of producing invalid fitted mass values. **Caution: zone center masses can be negative in some material compositions and this may cause numerical trouble**.
- **tau_bkg** is the background scattering lifetime. It is used with **broadening**=*landsberg_2* to broaden the negative part of the gain spectrum.
- **expo_mix** defines the amount of mixing of Landsberg and Lorentzian broadening, used in an exponential factor

$$Gain = Gain_{lorentzian} exp(-\Delta E_{imref}/expo\_mix)$$
$$+Gain_{landsberg}(1 - exp(-\Delta E_{imref}/expo\_mix))$$

where ΔE_{imref} is the difference between Fermi level splitting and transition bandgaps. This formula indicates that when ΔE_{imref} is greater than **expo_mix**,

the Lorentzian component is negligible and the Landsberg component takes over. **expo_mix** may be on the order of kT to signal the onset of thermally-activated carrier-carrier scattering.

- **active_loss** is used to add a background loss to the active region to account for loss mechanisms other than interband transition and free carrier absorption.
- **av_over_a** is the fraction of hydrostatic potential appearing in the valence band. For wurtzite, refer to **ac_bar** and **ac_well** instead.
- **k_range_wurt** is the same as **k_range** except it applies to the wurtzite material system.
- **diel_av** is the average dielectric constant of the well and the barrier for the many-body gain model.
- **plasma_coeff** is the plasma coefficient used in the many-body gain model.
- **sommerfeld_fac** typically takes a value between unity and two. Often called the Sommerfeld enhancement factor, it characterizes the degree of contribution of continuum state excitons into the exciton absorption. The continuum state excitons give some background in absorption without absorption peak, which is characteristic for bound excitons.
- **rmesh_fd** is a mesh ratio used to control the uniformity of the finite difference mesh for the gain calculations. A schematic diagram illustrating the effect of various ratios is given in Fig. 22.22.
- **two_photon_loss** is a two-photon absorption term in S^2 for this active region. See **two_photon_loss** for details.
- **two_photon_carr** is a free carrier loss coefficient related to the two-photon absorption process: these two effects combine to create an S^3 loss term for this active region. See **two_photon_carr** for details.
- **mater** is the material number assigned to a specific material. This number should always be consistent with the value used in the mesh generator and .geo file. If a label has previous been assigned to the material, **mater_label** may be used instead.
- **qw_print** is a flag number controlling the amount of information on quantum well models printed to the screen and to the message file with extension ".msg". When the value is "1", only limited or no amount of information is printed to the screen. When it is "2", detailed information on the quantum well model such as subband levels and transition matrix elements is printed to a file with extension " .msg" or as a screen runtime message.

- **level_srch** is a parameter used in the case where the well width is very large (500 Åor greater) and the subband level finder can miss some quantum levels. To fix this problem, **level_srch** may be used to sample the boundary condition function to search for the subband levels. When subband levels are very close to the bottom of the energy bands, a large **level_srch** should be used.

- **fd_mesh** is the finite difference mesh points used in computing the active region MQW subbands.

- **qw_plane** takes 100, 111 or 110. These are crystallographic planes used in the quantum MOS model. For non-polar or semi-polar wurtzite plane orientations, see **modify_wurtzite**.

- **coulomb_int_acc** is a selection number on the scale 1-9, which determines the order of Gauss-Legendre quadrature used in calculating integrals of electron-hole Coulomb interaction matrix element. The result is subsequently use in calculating exciton binding energy by Ritz variational method.

  - 1 corresponds to 2nd order quadrature.
  - 6 corresponds to 12th order quadrature; this is the default value.
  - 9 corresponds to 24th order quadrature.

- **embedded_structure** takes types 1 and 2. If it is set to zero, no embedded structures are present for the present layer/region. It is used to define an active material embedded inside a regular active layer.

  For type 1, it is used to define a corrugation grating in a DFB/DBR waveguide. A corrugation structure is regarded as two different kinds of active material embedded within an active layer material which has a material property equal to the overall average of the grating layer. The two embedded layers are considered active because interband model is used to calculate refractive index change. This kind of material was formerly referred to as “mildly active” because the gain at the lasing wavelength is close to zero: only the index change really matters.

  The 2nd type of embedded structure involves a quantum dot (QDOT) embedded within the wetting layer (regular quantum well). For quantum dot material, material style "cx-" is required. The solver will combine the quantum levels of the dot and and wetting layer to calculate the optical gain and spontaneous emission.

- **bulk_active** switches the gain model to use bulk calculations rather than those for a QW. This parameter is intended to be used as part of the new shared material library system in the 2014 version.

- **strained_bulk_active** is used to enable the use of strain in zincblende bulk active macros: normally, zincblende macros only support strain in QW active macros. This parameter is currently of limited use and was developed to model SiGe lasers.

- **uniaxial_strain** switches the default strain calculations from biaxial strain to uniaxial strain in zincblende active macros. This parameter is currently of limited use and was developed to model SiGe lasers.

  Note that in both models, the lateral strain from the epitaxial growth (ϵ_{xx}) is defined by commands such as **strain_well**. For biaxial strain, we further set $\epsilon_{yy} = \epsilon_{xx}$ while for uniaxial strain, we use $\epsilon_{yy} = \epsilon_{zz}$.

  The remaining strain component is obtained by enforcing a zero stress condition along the growth axis:

$$\sigma_{zz} = 0 = C_{12} * (\epsilon_{xx} + \epsilon_{yy}) + C_{11} * \epsilon_{zz}$$

  where C_{11} and C_{12} are stiffness coefficients. Shear strain terms such as ϵ_{xy} are not considered.

Examples

```
active_reg exch_coef=3.e-10 &&
   tau_scat=1.e-13  thickness=0.010  mater=3
```

22.27 active_reg_zdim

parameter	data type	values [defaults]
zdim	real	0.01(um)

---

active_reg_zdim is a statement used in the .layer file to define an active region thickness in the z direction. This is used in 3D simulations when each layer file represents a single x-y plane and the quantum well normal is along the z direction. The solver still needs to know the quantum well thickness to find the energy levels.

Parameters

- **zdim** is the z-dimension thickness in microns.

Examples

```
active_reg_zdim zdim=0.01
```

22.28 active_temper

parameter	data type	values [defaults]
mater_label	char	
ref_temper	real	[300.] degree
delta_tau_scat	real	[0.] (sec/K)
delta_ncarr_loss	real	[0.] (m^2/K)
delta_pcarr_loss	real	[0.] (m^2/K)
delta_active_loss	real	[0.] (m^{-1}/K)
delta_two_photon_loss	real	[0.] (m^2/K)
delta_two_photon_carr	real	[0.] (m^5/K)
mater	intg	[1]

The statement **active_temper** is used to specify temperature dependence in some of the important parameters for the active region of a laser diode. This statement may be used in the .gain or .sol files.

The linear temperature dependence of the parameters controlled by this statement is given by the following formula:

$$apar = apar0 + \Delta apar(T - T_{ref})$$

where $apar0$ is the initial value of the parameter at the reference temperature T_{ref} and $\Delta apar$ is the rate of temperature variation.

Note that for parameters defined in a macro file (see section 22.456), the temperature dependence can also be written as a function in the macro itself.

Parameters

- **ref_temper** is the reference temperature.
- **delta_tau_scat** is the rate of temperature change for the intra-band scattering lifetime.
- **delta_ncarr_loss** is the rate of temperature change for the intra-band electron free carrier absorption loss.

- **delta_pcarr_loss** is the rate of temperature change for the intra-band hole free carrier absorption loss.
- **delta_active_loss** is the rate of temperature change for the additional loss mechanism in the active region.
- **delta_two_photon_loss** is the rate of temperature change for the two-photon absorption coefficient in the active region.
- **delta_two_photon_carr** is the rate of temperature change for the carrier-induced two-photon loss absorption coefficient in the active region.
- **mater** is the material number assigned to the active region and must match the value in the **active_reg** statement. If a label has previously been assigned to this material, **mater_label** may be used instead.

Examples

```
active_temper ref_temper=300 delta_tau_scat=-0.01e-13  mater=3
```

22.29 acz_bar

Like ac_bar, this command defines the hydrostatic deformation potential in the barrier region of wurtzite quantum wells. However, it applies to the perpendicular strain ϵ_{zz}.

22.30 acz_well

Like ac_well, this command defines the hydrostatic deformation potential in wurtzite quantum wells. However, it applies to the perpendicular strain ϵ_{zz}.

22.31 add_acmemory

parameter	data type	values [defaults]
megbyte	real	[0.]

add_acmemory is used to add memory to the ac solver. This is used only for special cases where the default memory allocation is insufficient.

Parameters

- **megbyte** is the amount of memory in megabyte to be added.

Examples

```
add_acmemory megbyte=2
```

22.32 add_arnoldimemory

add_arnoldimemory is used to add memory to the direct Arnoldi eigenmode sparse solver. This is used only for special cases where the default memory allocation is insufficient. For more details, see the **add_acmemory** statement.

22.33 add_boundary

parameter	data type	values [defaults]
polygon_name	char	
edge_points	charx2	(void void)
limits	realx2	(0. 1.0) (μm)
boundary_num	intg 1	

add_boundary is used in the mesh generator to define a boundary region on the edge of a given polygon. It should be used instead of the **polygon** statement's own boundary definition parameters when there are multiple boundaries on the same polygon edge.

Parameters

- **polygon_name** is the name of the polygon.
- **edge_points** are the point labels (defined with the **point** statement) used to identify which edge of a polygon is being affected. Point labels should be given in counter-clockwise order.
- **limits** are the beginning and ending coordinates of the boundary. The first point label in **edge_points** is the zero reference.

- **boundary_num** is a number identifying the boundary region being applied to this polygon.

Examples

```
add_boundary polygon_name=poly1 edge_points=(a1 a2) &&
  limits=(0.0 1.0) boundary_num=1
```

22.34 add_mainmemory

parameter	data type	values [defaults]
megbyte	real	[0.]
sparse_fill	real	[0.] (megabyte)
symbolic	real	[0.] (megabyte)

add_mainmemory is used to add memory to the main solver. This is used only for special cases where the default memory allocation is insufficient. Usually, there will be error messages suggesting the use of this statement.

The most frequent case is 3D simulations where the connection between mesh elements on different planes result in non-zero terms that are far from the sparse matrix diagonal. This requires using more fill-in terms.

Parameters

- **megbyte** is the memory added to main sparse solver at the numerical factorization stage.
- **sparse_fill** is the memory added to the sparse filling space.
- **symbolic** is the memory added to the symbolic factorization stage of the sparse solver.

Examples

```
add_mainmemory sparse_fill=50
```

22.35 add_rcled_lambertian

parameter	data type	values [defaults]
mix_fraction	real	[0.]
extraction	real	[0.1]

This statement replaces **rcled_mix_lambertian** from previous versions of the software. Like its predecessor, it is used to artificially mix the angle-dependent optical power from the RCLED model with a background component having a Lambertian form. These two components can be understood as light being emitted from different parts of the LED.

Unlike its predecessor, this statement is used in the .plt post-processing rather than in the .sol file.

Parameters

- **mix_fraction** is the fraction of the light having a Lambertian emission profile.
- **extraction** is the extraction efficiency for the Lambertian light component. The extraction for the RCLED component is determined internally.

Examples

```
add_rcled_lambertian mix_fraction=0.2 extraction=0.1
```

22.36 add_thermalmemory

add_thermalmemory is used to add memory to the thermal solver. This is used only for special cases where the default memory allocation is insufficient. See add_acmemory for further examples and details.

22.37 adjust_active_reg

parameter	data type	values [defaults]
valence_mixing	char	[void]
exch_coef	real	
tau_scat	real	
ncarr_loss	real	
pcarr_loss	real	
dip_factor	real	
active_loss	real	
two_photon_loss	real	
two_photon_carr	real	
qw_print	intg	

adjust_active_reg is used in the .layer file to adjust the parameters of the **active_reg** statement that is automatically generated as part of the active material declaration when this file is processed. In general, this is not used and a global override of gain calculation settings with **set_active_reg** is preferred.

Refer to **active_reg** for further details.

22.38 adjust_column_screening

parameter	data type	values [defaults]
screening	real	[0.5]

adjust_column_screening is used in the .layer file to locally adjust the screening coefficient of the **set_polarization** command. It can be used to account for local defects and surface states which compensate the fixed charge due to piezoelectric effects.

The modified screening coefficient will apply to all layer interfaces in a given column. See also **adjust_layer_screening**.

Parameters

- **screening** is the local value of the screening coefficient for all layer interfaces in this column.

Examples

```
column column_num=1 w=0.04 mesh_num=10 r=-1.1
adjust_column_screening screening=0.0
```

This will force all the interface charges in column #1 to be zero.

22.39 adjust_doping

parameter	data type	values [defaults]
impurity	char	[void]
pf_model	char	void
level	real	

adjust_doping is used in the .layer file to adjust parameters in the **doping** statement that is automatically generated when processing the file. See **doping**for further details and examples.

22.40 adjust_layer_screening

parameter	data type	values [defaults]
interface	char	[top],bottom
screening	real	[0.5]

adjust_layer_screening is used in the .layer file to locally adjust the screening coefficient of the **set_polarization** command. It can be used to account for local defects and surface states which compensate the fixed charge due to piezoelectric effects.

See also **adjust_column_screening**.

Parameters

- **interface** defines the location of the interface whose screening coefficient is modified. The position is defined relative to the preceding **layer** statement.
- **screening** is the local value of the screening coefficient.

Examples

```
layer_mater macro_name=gan
layer  d=0.0025  n=20  r=-1.1
adjust_layer_screening interface=top screening=0.44
```

This will modify the interface charges on top of the GaN layer.

22.41 affinity

affinity is a passive macro material statement defining the electron affinity (in eV) of a material. In metal and resistors, this value is equal to the work function: it is the difference between the vacuum level and the conduction band edge. For active layers, **affinity** will be overridden by band offset parameters from the active macro.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.42 a1_bar

ai_bar,i=1...6 are a set of parameters used to define barrier properties in wurtzite quantum well active macros. They define the effective valence mass terms of the Hamiltonian, labelled as (A_i) in Ref [62]. The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.43 a2_bar

See Sec. 22.42.

22.44 a3_bar

See Sec. 22.42.

22.45 a4_bar

See Sec. 22.42.

22.46 a5_bar

See Sec. 22.42.

22.47 a6_bar

See Sec. 22.42.

22.48 a1_bulk

ai_bulk,i=1...6 are a set of parameters used to define material properties in bulk wurtzite active macros. They define the effective valence mass terms of the Hamiltonian, labelled as (A_i) in Ref [62].

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.49 a2_bulk

See Sec. 22.48.

22.50 a3_bulk

See Sec. 22.48.

22.51 a4_bulk

See Sec. 22.48.

22.52 a5_bulk

See Sec. 22.48.

22.53 a6_bulk

See Sec. 22.48.

22.54 a1_well

ai_well,i=1...6 are a set of parameters used to define well properties in wurtzite quantum well active macros. They define the effective valence mass terms of the Hamiltonian, labelled as (A_i) in Ref [62].

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.55 a2_well

See Sec. 22.54.

22.56 a3_well

See Sec. 22.54.

22.57 a4_well

See Sec. 22.54.

22.58 a5_well

See Sec. 22.54.

22.59 a6_well

See Sec. 22.54.

22.60 align_complex

parameter	data type	values [defaults]
x_range	realx2	(um)

This statement defines the limits of a complex quantum-coupled region along the x axis. It is primarily used when importing mesh from CSUPREM.

See also **y_from_bottom** in **begin_complex**.

22.61 alignment_ref

This statement defines a reference energy level used to compute Schottky contact barrier heights. It is commonly used when a single contact touches different materials.

This energy level is defined via a material macro and composition so all parameters are the same as in **layer_mater**.

22.62 alpha_n

See **electron_mobility**

22.63 alpha_p

See **hole_mobility**.

22.64 alpha_wavel

alpha_wavel is the same as **gain_wavel** except that it plots the linewidth enhance factor α.

22.65 analytical_gain

parameter	data type	values [defaults]
import_fit_data	char	[no]
fit_data_file	char	[fitgain.txt]
import_spectrum_only	char	[no]
fit_density	real	[2.e24] (m^{-3})
fit_density_p	real	[-9999.] (m^{-3})
fit_gain	real	[10000] (m^{-1})
dg_dn	real	[1.e-19] (m^2)
dg_dp	real	[1.e-19] (m^2)
dg2_dlambda2	real	[3.e8] ($m^{-1}/\mu m^2$)
peak_wavelength	real	[0.83] (μm)
peak_shift_n	real	[0.] ($m^{-1}/\mu m^{-3}$)
peak_shift_p	real	[0.] ($m^{-1}/\mu m^{-3}$)

This statement overrides the internal gain calculations and replaces them with a simple analytical model often found in textbooks:

$$\begin{aligned} g_{peak} &= g_0 + \frac{\partial g}{\partial n}(n - n_0) + \frac{\partial g}{\partial p}(p - p_0) \\ \lambda_{peak} &= \lambda_0 + \frac{\partial \lambda}{\partial n}(n - n_0) + \frac{\partial \lambda}{\partial p}(p - p_0) \\ g(\lambda) &= g_{peak} - \frac{1}{2}\frac{\partial^2 g}{\partial \lambda^2}(\lambda - \lambda_{peak}) \end{aligned} \tag{22.4}$$

These values may be obtained by using the **fit_gain_wavel** statement in the .gain file.

Parameters

Most of the parameters are defined in Eq. 22.4 above.

- **import_fit_data** uses fitting parameters from **fit_data_file** instead of the parameters from this statement.
- **import_spectrum_only** imports only the spectral shift information.
- **fit_density** and **fit_density_p** are equal to n_0 and p_0, respectively. By default, $p_0 = n_0$.

- **fit_gain** is g_0.
- **dg_dn** and **dg_dp** are $\frac{\partial g}{\partial n}$ and $\frac{\partial g}{\partial p}$, respectively.
- **dg2_dlambda2** is equal to $\frac{\partial^2 g}{\partial \lambda^2}$. It can be related to the gain width.
- **peak_wavelength** is λ_0.
- **peak_shift_n** and **peak_shift_p** are $\frac{\partial \lambda}{\partial n}$ and $\frac{\partial \lambda}{\partial p}$, respectively.

Examples

```
analytical_gain fit_density=1.5e24 peak_wavelength=1.55
```

22.66 auger_n

auger_n is a passive macro material statement defining the Auger coefficient for electrons. See Sec. 5.1.1 for details.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.67 auger_p

auger_p is a passive macro material statement defining the Auger coefficient for holes. See Sec. 5.1.1 for details.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.68 auto_tunneling

parameter	data type	values [defaults]
carrier_type	char	[both], electron, hole
max_range	real	[0.02] (μm)
min_barrier	real	[0.1] (eV)
ratio_division	real	[1.]
shift_division	real	[0.] (μm)
division	intg	[1]

auto_tunneling instructs the software to automatically look for barriers that block current flow and define an appropriate intraband tunneling command. In effect, this command replaces manual usage of the **tunneling** command and defines a propagation matrix model for the automatically-detected tunneling range.

Note that this command only looks for barriers present under equilibrium conditions: barrier heights may increase or decrease with applied bias.

Parameters

- **carrier_type** determines whether the software looks for barriers that block the flow of a particular kind of charge carrier.
- **max_range** determines the maximum distance allowed for the automatic tunneling profile.
- **min_barrier** is the minimum barrier height required during the search to trigger the declaration of an automatic tunneling range. Smaller barriers will be ignored and will default back to the usual thermionic emission model.
- **division** is a factor used to oversample the existing mesh for the purposes of the tunneling model. **ratio_division** and **shift_division** control the distribution profile of this new mesh, following the usual rules outlined in the **put_mesh** commands.

22.69 az_bar

az_bar is the same as **a_bar** except in the z-direction (c-plane axis).

22.70 az_well

az_well is the same as **a_well** except in the z-direction (c-plane axis).

22.71 b_bar

This active layer statement is identical to **b_well** except that it refers to the barrier material in a simplified quantum well macro. Note that this parameter is not used in complex MQW macros.

22.72 b_well

The material statement **b_well** is an active layer macro statement used to define the shear deformation potential (eV) in a quantum well and thus, the separation between the heavy and light hole bands. Note that this parameter only applies to materials with zincblende symmetry. For wurtzite materials, a set of parameters define in **ai_well,i=1...6** are used instead.

See Sec. 10.1 for additional details on band alignment rules.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.73 back_index

parameter	data type	values [defaults]
real_index	real	[-9999.]
imag_index	real	[0.]

back_index is an outside index value used as a boundary condition for optical pumping. The default value assumes a continuous index to prevent back reflections. This command has no practical effect if **back_reflection** is used.

See the theory section in **front_reflection** for more details.

22.74 back_reflection

See **front_reflection**.

22.75 band_discont

band_discont is an active layer macro statement that defines the conduction band discontinuity (in eV) between an active layer and the neighboring passive layer. This value overrides both the **band_offset** statement from active macros and the **affinity** statement from passive macros. A negative value can be used to define a type II band alignment.

Note that **band_discont** will define the discontinuity on both sides of the layer unless **band_discont_right** is used. In that case, it only defines the discontinuity

on the left. This should be used in complex MQW regions (cx-style macros) so the discontinuity is set correctly on the edges of the complex region.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.76 band_discont_right

See **band_discont**.

22.77 band_distance

parameter	data type	values [defaults]
data_file	char	
use_fermi_level	char	[no]
conc_logscale	char	[no]
prop_file	char	[void]
ignore_cband	char	[no]
ignore_vband	char	[no]
plot_qcl_3levels	char	[no]
band_edge_file	char	[void]
use_sheet_density	char	[yes]
conc	real	[1.e15] (m^{-2} or m^{-3})
pn_ratio	real	[1.]
qw_wave_ht	real	[0.1]
fermi_n	real	[0.]
fermi_p	real	[0.]
band_range	realx2	[-1.5 1.5]
data_point	intg	[90]
cond_valley	intg	[1]
val_valley	intg	[1]
cond_valley_prop	intg	[0]
val_valley_prop	intg	[0]

---

band_distance plots the band and subband structure versus distance. This statement is only used in the gain preview module or preview session before a full-blown simulation.

By default, a flat band model is used. A local field can be defined **gain_module** but **self_consistent** should also be used in this case.

Parameters

- **data_file** is an output file containing a copy of the plot data in ASCII format.
- **use_fermi_level** tells the software to directly set the Fermi levels according to **fermi_n** and **fermi_p**.
- **conc_logscale** used to indicate whether log scale is used in plotting any concentrations.
- **prop_file** is a file to store the property results from this statement. All results will be appended to this file so that multiple simulations can be summarized within this file.
- **ignore_cband** will ignore conduction band so that only the valence band will appear.
- **ignore_vband** will ignore valence band so that only the conduction band will appear.
- **plot_qcl_3levels** lets the program plot only the 3 critical quantum levels related to intersubband transition in a quantum cascade laser.
- **band_edge_file** instruct the program to save the band edge profile to this file so that an upper level project can use it later.
- **conc** is the total electron concentration used to determine the Fermi levels and subband population. The units of this parameter change depending on the **use_sheet_density** setting. If set to *yes*, sheet density in m^{-2} is used; otherwise, bulk density in m^{-3} is used. As of the 2012 version, the new default is to use the sheet density.

  Sell also the **gain_module** statement which also enables the use of sheet density in other gain preview commands.
- **pn_ratio** is the ratio of hole over electron concentrations. Note that this ratio can be set to an arbitrary number in the gain preview. In the main solver, this ratio is determined automatically by the simulator according to the local Fermi levels.
- **qw_wave_ht** is the height of the quantum wave amplitude as plotted on the band diagram.
- **fermi_n** is the electron Fermi level (eV) from the conduction band edge.

- **fermi_p** is the hole Fermi level (eV) from the HH valence band edge.
- **band_range** is the energy range (in units of eV) in a band diagram plot.
- **data_point** is the number of data points in one curve.
- **cond_valley** directs the program to plot properties related to a particular conduction band valley with this valley index. The valley index means band valleys with labels such as Gamma, L, X, or Delta2, Delta4, etc.
- **val_valley** directs the program to plot properties related to a particular valence band valley with this valley index. The valley index means band valleys with labels such as HH, LH, SO, etc.
- **cond_valley_prop** is kth the conduction band property as defined by statement **condj_valley_propk**. The valley number j is defined by the parameter **cond_valley**.
- **val_valley_prop** is kth the conduction band property as defined by statement **valj_valley_propk**. The valley number j is defined by the parameter **val_valley**.

22.78 band_gap

band_gap is a passive macro material statement used the define the energy bandgap (in eV) of a bulk semiconductor. For quantum well barriers and active regions, this value will be overridden with the values of **eg0_bar** or **eg0_well** from the active macro.

For wurtzite materials, the bulk unstrained bandgap is given by **eg0_bulk** instead of **band_gap**.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.79 band_offset

band_offset is an active layer macro statement defining the band offset of a quantum well as defined in Sec. 10.1. This value overrides the **affinity** statement from the passive macro and may itself be overridden by **band_discont**.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.80 bandgap_narrow

parameter	data type	values [defaults]
model	char	[slotboom]

The statement **bandgap_narrow** is used to include the bandgap narrowing effect. Currently, only the Slotboom model is supported.

22.81 bandgap_optical_gen

parameter	data type	values [defaults]
allowed	char	[no],yes
bandgap_gen_iqe	real	[0.1]

bandgap_optical_gen is used to enable optical generation at photon energies smaller than the bandgap of the material.

Parameters

- **allowed** turns this effect on or off.
- **bandgap_gen_iqe** is the internal quantum efficiency (IQE) for the below-bandgap optical generation.

Examples

```
bandgap_optical_gen allowed=yes bandgap_gen_iqe=0.01
```

22.82 barrier_correction_by_qw_model

This statement overrides the band alignment for the last barrier of a complex MQW region so that it follows the same rules that are used for the inner wells (c.f. Sec. 10.1). It also instructs the software to use other aspects of the quantum well model for that region, such as the carrier masses.

This statement is only used for the simplified complex library system (c.f. Sec. 3.5.1); it has no additional parameters.

22.83 basic_var_symbol

parameter	data type	values [defaults]
var_lib	char	
var_basic	char	

basic_var_symbol translates a variable/symbol name from the library to the underlying basic/passive macro. This statement can be omitted if the variable name is the same in both the library and the macro.

See **material_lib** and **complex_var_symbol** for further information.

Examples

```
begin_library AlGaAs
import_basic name=algaas
import_complex name=cx-AlGaAs
complex_var_symbol  var_lib=x var_complex=xw
end_library
```

This set of commands defines the "AlGaAs" library as being composed of the "algaas" passive macro and the "cx-AlGaAs" active macro. The material parameter x used when invoking the library is translated into the xw parameter of the active macros and used "as-is" in the passive macro.

22.84 begin_bpmplot

begin_bpmplot is used as the starting point of a post-processing file displaying results from the Crosslight BPM model. The matching end command is **end_bpmplot**.

22.85 begin_cavity

parameter	data type	values [defaults]
cavity_num	intg	[1]

begin_cavity is used as a starting bracket for the statements belonging to a laser cavity in a multiple laser cavity application. The matching closing bracket is **end_cavity**.

This can be used in multi-junction devices where the optical modes peak in different active regions and there is poor overlap between the two sets of modes. It can also be used in quantum dot devices where there are multiple gain peaks at very different wavelength ranges.

Parameters

- **cavity_num** is the cavity number.

Examples

```
$$$$$$$$$$$$$$$$$$
$ Cavity #1 at 1.55 um
$$$$$$$$$$$$$$$$$$
begin_cavity cavity_number=1
init_wave init_wavel=1.55
multimode ...
end_cavity
$$$$$$$$$$$$$$$$$$
$ Cavity #2 at 1.3 um
$$$$$$$$$$$$$$$$$$
begin_cavity cavity_number=2
init_wave init_wavel=1.3
multimode ...
end_cavity
```

22.86 begin_complex

parameter	data type	values [defaults]
tag	char	void
cx_qw_side	char	[bottom], top
use_xy_range	char	[no] yes
y_start_label	char	[void]
qw_thick	real	(um)
y_start_complex	real	(um)
x_start_complex	real	(um)
y_from_bottom	real	(um)
layer_num	intg	[3]
column_num	intg	[1]

begin_complex is used as a starting bracket for a group of layers coupled to each other quantum mechanically, forming a roughly rectangular region. The matching closing bracket is **end_complex**.

See **complex_region** for examples and more details.

Parameters

- **tag** is a user-defined label to define identical complex MQW regions: this helps the solver save on computation time by re-using the solutions of the Schrödinger solver. It is usually defined through the **start_same_complex** command in the .layer file.
- **layer_num** is the number of layer within a complex.
- **column_num** is the number of columns within a complex.
- **use_xy_range** indicates whether the rectangle region is defined using an x-y range or by using the polygon names. The latter method is usually preferred.
- **y_start_label** and **y_end_label** are position labels used to define the y-range of the rectangle region.
- **y_start_complex** specifies the absolute y-coordinate of the starting point of the rectangle (um).
- **x_start_complex** specifies the absolute x-coordinate of the starting point of the rectangle (um).

- **y_from_bottom** specifies the y-position of the starting point of the rectangle (um) relative to the bottom of the mesh. It is primarily used in conjunction with **align_complex** when importing mesh from CSUPREM.

- **cx_qw_side** is used to define a special case of a complex MQW region with only two layers. For historical reasons (see **complex_region**) an odd number of layers is normally required in a complex region and the even-numbered layers are defined as quantum wells. With **cx_qw_side**, one of the two layers is internally split by the software to produce this barrier-well-barrier layout: this parameter specifies which of the two layers is to be split. The parameter **qw_thick** defines how much of this layer is used as a quantum well region.

  This parameter is most often used in GaN HEMT simulations where the piezoelectric potential defines the quantum confinement region. In this case, the two layers used to define the complex are a thick GaN region which serves as both barrier and well and an AlGaN cap layer which forms the top barrier.

22.87 begin_qwire_complex

parameter	data type	values [defaults]
x1	real	(μm)
x2	real	(μm)
y1	real	(μm)
y2	real	(μm)
x1_label	char	
x2_label	char	
y1_label	char	
y2_label	char	
auto_search	char	yes,[no]

begin_qwire_complex is used in the .mater or .sol file and is usually generated automatically by the .layer file statement **start_qwire_complex**. It serves the same role as **begin_complex** except that it defines the start of a 2D quantum-confined region (a quantum wire) rather than a complex MQW region with 1D confinement.

end_qwire_complex is the closing tag for this statement. Between these two commands, **qwire_complex_region** statements are used to define the materials in the quantum wire cross-section.

Parameters

- **x1**, **x2**, **y1** and **y2** define the xy range of the quantum wire cross-section. If position labels have been defined, **x1_label**, **x2_label**, **y1_label** and **y2_label** may be used instead.
- **auto_search** automatically sets the cross-section range based on material numbers. When importing mesh from CSUPREM, material labels must be set to correctly identify the quantum region.

22.88 begin_zdir_complex

parameter	data type	values [defaults]
num_segment	intg	[1]

begin_zdir_complex is used as a starting bracket for a group of layers coupled to each other quantum mechanically in the z direction (i.e. stacked mesh planes in a 3D simulation). The matching closing bracket is **end_zdir_complex**.

See also **zdir_cx** for more information.

Parameters

- **num_segment** is the z-segment number where the quantum-confined region begins.

Examples

```
begin_zdir_complex num_segment=   9
zdir_cx zseg=  16 type=barrier mater=   3
zdir_cx zseg=  17 type=barrier mater=   3
zdir_cx zseg=  18 type=well xy_ref=  0.1000E+01  0.5000E-01 mater=   4
zdir_cx zseg=  19 type=well xy_ref=  0.1000E+01  0.5000E-01 mater=   4
zdir_cx zseg=  20 type=well xy_ref=  0.1000E+01  0.5000E-01 mater=   4
zdir_cx zseg=  21 type=well xy_ref=  0.1000E+01  0.5000E-01 mater=   4
zdir_cx zseg=  22 type=well xy_ref=  0.1000E+01  0.5000E-01 mater=   4
zdir_cx zseg=  23 type=barrier mater=   3
zdir_cx zseg=  24 type=barrier mater=   3
end_zdir_complex
```

22.89 begin_zmater

parameter	data type	values [defaults]
zseg_num	intg	[1]

begin_zmater is used as a starting bracket for the material and doping statements belonging to a given z-segment. The matching closing bracket is **end_zmater**.

Parameters

- **zseg_num** is the z-segment number

Examples

```
{
begin_zmater zseg_num=1
include file=ref.mater
include file=ref.doping
end_zmater
}
```

22.90 bend_extern

parameter	data type	values [defaults]
variation	char	function,table,[constant]
var_symboli(1=1..5)	char	void
vari(i=1..5)	real	
value	real	
func_num	intg	[1]

bend_extern is used to define an external function used by **bend_xy_plane**.

Parameters

Except for **func_num**, all parameters are the same as in **bend_xy_plane**.

- **func_num** is a label to identify different external functions so that this statement can be used more than once.

Examples

See **bend_xy_plane**.

22.91 bend_xy_plane

parameter	data type	values [defaults]
variation	char	linear_x,linear_y,function,[constant]
var_symboli(1=1..5)	char	void
vari(i=1..5)	real	
value	real	
zseg_num	intg	[1]

To realize complex device shapes, the user may bend 2D mesh planes (x-y) in the z-direction. This bending is represented by a small local offset of the mesh point:

$$(x, y, z) \rightarrow (x, y, z + \Delta z(x, y)) \tag{22.5}$$

bend_xy_plane defines this bending function $\Delta z(x, y)$ for a single plane. It should return an offset value in microns. External functions may be defined to re-use certain elements and simplify the declaration by passing the z-position of the plane as a function argument: see **bend_extern**.

Parameters

- **variation** defines how the offset varies with position. There are two options for this parameter:
  - *constant* means that a fixed value is used everywhere.
  - *function* means a mathematical function of the form:

```
function(variable1, variable2, ...)
......
end_function
```

All lines in a function except the last one must end with a semicolon (;) to suppress the line output. The output of the last line is the value returned by the function. The rules for mathematical functions are further explained in Appendix B. Three reserved keywords can be used in addition to user-defined variables: x_point and y_point (the in-plane point coordinates) and z_plane (the position of the mesh plane before bending).

- **var_symboli** is an external function variable symbol to be used by the function mentioned above.
- **vari** is an external function variable value to be used by the function mentioned above.
- **value** is the constant value of shift in z-direction if **variation**=*constant*.
- **zseg_num** is the z-segment number affected by this statement.

Examples

```
$ bend function at z=350
$ bend = 0 @ y = 402, > 0 @ y =0 => points downwards
bend_extern func_num=1 variation=function
function(x_point,y_point)
y_0=0; z_0=-350; radius=sqrt(350**2+402**2);
z_0+sqrt(radius**2-(y_point-y_0)**2)
end_function

$ z-segment #1 is unbended
bend_xy_plane variation=function zseg_num=2
function(z_plane,x_point,y_point)
z0 = 100.0; z1 = 350.0;
zfac=(z_plane-z0)/(z1-z0);
bend_extern1*zfac
end_function
```

22.92 beta_mte

beta_mte is used in the modified transferred-electron mobility model for GaN. When this model is activated, the software actually implements the mixture of the MTE and Canali models proposed in Eqn.(11) of Ref. [7]. This model has a number of hard-coded parameters taken from the above reference but the value defined by **beta_mte** corresponds to β_T in Eqn.(3).

A number of other parameters related to the Canali model are also defined as usual. For example. **beta_n** corresponds to β_C in Eqn.(1).

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.93 beta_n

beta_n defines a parameter in the "beta" field-dependent mobility model for electrons (see Eq. 5.37).

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.94 beta_p

beta_p defines a parameter in the "beta" field-dependent mobility model for holes (see Eq. 5.38).

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.95 bias_output_at_maximum

parameter	data type	values [defaults]
variable	char	
xrange	realx2	(μm)
yrange	realx2	(μm)
zrange	realx2	(μm)

bias_output_at_maximum replaces the previous **more_bias_output** command; this command is specialized to output the spatial maximum of a variable. See also the following related commands which also replace **more_bias_output**:

- **bias_output_curr_flux**
- **bias_output_tunneling**
- **bias_output_longitudinal**

- **bias_output_near_point**
- **bias_output_spatial_integral**
- **bias_output_wave_average**

These various commands generate additional bias-dependent data for later plotting using the **plot_scan** command. Each statement generates a new plotting variable stored in *more_bias_vari,i=1..15*; the numbering corresponds to the order of the various bias_output commands in the .sol input file.

Parameters

- **variable** is the variable being exported:
  - *optical_loss*: the total optical loss
  - *optical_loss_mirror*: the optical mirror losses
  - *optical_loss_ncarr*: the optical loss due to electron free carrier absorption
  - *optical_loss_pcarr*: the optical loss due to hole free carrier absorption
  - *optical_loss_two_photon*: the optical loss due to the two-photon absorption process
  - *optical_loss_two_photon_carr*: the optical loss from free carriers generated during the two-photon absorption process
  - *potential*: the internal potential V
  - *field_mag*: the electric field magnitude
  - *electron*: the electron concentration
  - *hole_curr_mag*: the hole current magnitude
  - *recomb_aug*: the Auger recombination rate
  - *recomb_rad*: the total spontaneous emission rate
  - *recomb_rad_te*: the TE mode spontaneous emission rate
  - *recomb_rad_tm*: the TM mode spontaneous emission rate
  - *recomb_srh*: the SRH recombination rate
  - *recomb_st*: the stimulated recombination rate
  - *recomb_all*: the total recombination rate
  - *impact_alpha_n*: impact ionization coefficient for electrons
  - *impact_alpha_p*: impact ionization coefficient for holes
  - *impact_ionization*: total impact ionization rate

  - *impact_elec_rate*: electron impact ionization rate
  - *impact_hole_rate*: hole impact ionization rate

- **xrange**, **yrange** and **zrange** define the coordinates inside which the search for the spatial maximum takes place.

22.96 bias_output_curr_flux

parameter	data type	values [defaults]
variable	char	
at_y	real	(μm)

bias_output_curr_flux replaces the previous **more_bias_output** command; this command is specialized to output local current values at specific points of the device. See also the following related commands which also replace **more_bias_output**:

- **bias_output_tunneling**
- **bias_output_at_maximum**
- **bias_output_longitudinal**
- **bias_output_near_point**
- **bias_output_spatial_integral**
- **bias_output_wave_average**

These various commands generate additional bias-dependent data for later plotting using the **plot_scan** command. Each statement generates a new plotting variable stored in *more_bias_vari,i=1..15*; the numbering corresponds to the order of the various bias_output commands in the .sol input file.

Parameters

- **variable** is the variable being exported:
  - *elec_curr*: the electron current magnitude
  - *hole_curr*: the hole current magnitude

  - *nonloc_esc_cap_curr_n*: non-local escape current for electrons in **q_transport** model.
  - *nonloc_fly_over_curr_n*: non-local flyover current for electrons in **q_transport** model.

- **at_y** is the position of the y cutline used to export the data.

22.97 bias_output_ii_integral

parameter	data type	values [defaults]
variable	char	[max_alpha],
method	char	[via_max_field], manual
between_contacts	intgx2	[1 3]
pointi (i=1..9)	realx3	(μm)

bias_output_ii_integral is used to integrate the impact ionization profile and output the result as a variable that can be plotted vs. bias in the post-processing stage. Each statement generates a new plotting variable stored in *more_bias_vari,i=1..15*; the numbering corresponds to the order of the various bias_output commands in the .sol input file.

Parameters

- **variable** is the variable being integrated:
  - *impact_alpha_n*: the impact generation rate for electrons
  - *impact_alpha_p*: the impact generation rate for holes
  - *max_alpha*: the maximum of the two above values
- **method**
- **between_contacts**
- **pointi (i=1..9)**

22.98 bias_output_longitudinal

parameter	data type	values [defaults]
variable	char	
long_fraction	realx2	[0.5]
long_mode_index	intg	[1]

bias_output_longitudinal replaces the previous **more_bias_output** command; this command is specialized to output longitudinal data from PICS3D. See also the following related commands which also replace **more_bias_output**:

- **bias_output_curr_flux**
- **bias_output_tunneling**
- **bias_output_at_maximum**
- **bias_output_near_point**
- **bias_output_spatial_integral**
- **bias_output_wave_average**

These various commands generate additional bias-dependent data for later plotting using the **plot_scan** command. Each statement generates a new plotting variable stored in *more_bias_vari,i=1..15*; the numbering corresponds to the order of the various bias_output commands in the .sol input file.

Parameters

- **variable** is the variable being exported:
  - *kappa_l_real*: real part of the κL coupling coefficient
  - *kappa_l_imag*: imaginary part of the κL coupling coefficient
  - *phase_index*: effective index of the optical mode
  - *alpha_l_over_2*: $\frac{\alpha L}{2}$ from the coupled round-trip gain model
  - *delta_beta_l*: $\delta\beta L$ from the coupled round-trip gain model
  - *alpha_integral*: integrated value of $\frac{\alpha(z)}{2}$ in the round-trip gain model
  - *delta_beta_integral*: integrated value of $\delta\beta(z)$ in the round-trip gain model

  - *linewidth_spon*: linewidth contribution from spontaneous emission noise
  - *linewidth_carrier*: linewidth contribution from carrier density fluctuations
  - *linewidth_cross*: linewdith contribution due to cross-carrier modulation
  - *linewidth_sidemode*: linewidth contribution due to side modes
  - *linewidth_total*: total linewidth of main mode
  - *linewidth*power*: linewidth power product
  - *effective_alpha*: linewidth enhancement factor (α_H)
  - *density_mode_average*: carrier density averaged using the local mode profile as a weighting coefficient

- **long_fraction** sets the export position of the data as a fraction of the optical cavity length. It is not used for integrated variables.
- **long_mode_index** specifies which longitudinal mode is used in the export.

22.99 bias_output_mater_average

parameter	data type	values [defaults]
variable	char	
mater_label	char	
mater	intg	[1]

bias_output_mater_average is similar to other commands in the bias_output family. This version of the command is specialized to output the average of a variable over all the mesh points belonging to a specific material.

All commands in the bias_output family generate additional bias-dependent data for later plotting using the **plot_scan** command. Each statement generates a new plotting variable stored in *more_bias_vari,i=1..15*; the numbering corresponds to the order of the various bias_output commands in the .sol input file.

Parameters

- **variable** is the variable being exported.
- **mater_label** is the material label identifying the material for the variable average.
- **mater** is the material number for the variable average.

22.100 bias_output_near_point

parameter	data type	values [defaults]
variable	char	
near_xyz	realx3	(μm)

bias_output_near_point replaces the previous **more_bias_output** command; this command is specialized to output a spatial variable near a specified point. See also the following related commands which also replace **more_bias_output**:

- **bias_output_curr_flux**
- **bias_output_tunneling**
- **bias_output_longitudinal**
- **bias_output_at_maximum**
- **bias_output_spatial_integral**
- **bias_output_wave_average**

These various commands generate additional bias-dependent data for later plotting using the **plot_scan** command. Each statement generates a new plotting variable stored in *more_bias_vari,i=1..15*; the numbering corresponds to the order of the various bias_output commands in the .sol input file.

Parameters

- **variable** is the variable being exported:
  - *optical_loss*: the total optical loss
  - *optical_loss_mirror*: the optical mirror losses
  - *optical_loss_ncarr*: the optical loss due to electron free carrier absorption
  - *optical_loss_pcarr*: the optical loss due to hole free carrier absorption
  - *optical_loss_two_photon*: the optical loss due to the two-photon absorption process
  - *optical_loss_two_photon_carr*: the optical loss from free carriers generated during the two-photon absorption process

  - *potential*: the internal potential V
  - *field_mag*: the electric field magnitude
  - *electron*: the electron concentration
  - *hole_curr_mag*: the hole current magnitude
  - *recomb_aug*: the Auger recombination rate
  - *recomb_rad*: the total spontaneous emission rate
  - *recomb_rad_te*: the TE mode spontaneous emission rate
  - *recomb_rad_tm*: the TM mode spontaneous emission rate
  - *recomb_srh*: the SRH recombination rate
  - *recomb_st*: the stimulated recombination rate
  - *recomb_all*: the total recombination rate
  - *impact_alpha_n*: impact ionization coefficient for electrons
  - *impact_alpha_p*: impact ionization coefficient for holes
  - *impact_ionization*: total impact ionization rate
  - *impact_elec_rate*: electron impact ionization rate
  - *impact_hole_rate*: hole impact ionization rate

- **near_xyz** defines a set of coordinates; the mesh point nearest to these coordinates will be used for the export.

22.101 bias_output_peak_wavelength

parameter	data type	values [defaults]
variable	char	

bias_output_peak_wavelength searches for the peak wavelength of a spectral variable and outputs it for all bias steps so that it may be plotted in the post-processing stage. Each statement generates a new plotting variable stored in *more_bias_vari,i=1..15*; the numbering corresponds to the order of the various bias_output commands in the .sol input file.

Parameters

- **variable** is the spectral variable that is searched when exporting the peak. This variable corresponds to the full spectrum plot available in **gain_spectrum** or **plot_spectrum**.

22.102 bias_output_spatial_integral

parameter	data type	values [defaults]
variable	char	
active_reg_only	char	[no], yes
inactive_reg_only	char	[no], yes
xrange	realx2	(μm)
yrange	realx2	(μm)
zrange	realx2	(μm)

bias_output_spatial_integral replaces the previous **more_bias_output** command; this command is specialized to output the integral of a spatial variable over a specified range. See also the following related commands which also replace **more_bias_output**:

- **bias_output_curr_flux**
- **bias_output_tunneling**
- **bias_output_longitudinal**
- **bias_output_at_maximum**
- **bias_output_near_point**
- **bias_output_wave_average**

These various commands generate additional bias-dependent data for later plotting using the **plot_scan** command. Each statement generates a new plotting variable stored in *more_bias_vari,i=1..15*; the numbering corresponds to the order of the various bias_output commands in the .sol input file.

Parameters

- **variable** is the variable being exported:
  - *potential*: the internal potential V
  - *field_mag*: the electric field magnitude
  - *electron*: the electron concentration
  - *hole_curr_mag*: the hole current magnitude

  - *recomb_aug*: the Auger recombination rate
  - *recomb_rad*: the total spontaneous emission rate
  - *recomb_rad_te*: the TE mode spontaneous emission rate
  - *recomb_rad_tm*: the TM mode spontaneous emission rate
  - *recomb_srh*: the SRH recombination rate
  - *recomb_st*: the stimulated recombination rate
  - *recomb_all*: the total recombination rate
  - *impact_alpha_n*: impact ionization coefficient for electrons
  - *impact_alpha_p*: impact ionization coefficient for holes
  - *impact_ionization*: total impact ionization rate
  - *impact_elec_rate*: electron impact ionization rate
  - *impact_hole_rate*: hole impact ionization rate

- **active_reg_only** restricts the integration to optically active regions.
- **inactive_reg_only** restricts the integration to optically passive regions.
- **xrange**, **yrange** and **zrange** define the integration range.

22.103 bias_output_tunneling

parameter	data type	values [defaults]
variable	char	
tunnel_region	intg	[1]

---

bias_output_tunneling replaces the previous **more_bias_output** command; this command is specialized to output tunneling data for a specific tunneling region. See also the following related commands which also replace **more_bias_output**:

- **bias_output_curr_flux**
- **bias_output_at_maximum**
- **bias_output_longitudinal**
- **bias_output_near_point**
- **bias_output_spatial_integral**

- **bias_output_wave_average**

These various commands generate additional bias-dependent data for later plotting using the **plot_scan** command. Each statement generates a new plotting variable stored in *more_bias_vari,i=1..15*; the numbering corresponds to the order of the various bias_output commands in the .sol input file.

Parameters

- **variable** is the variable being exported:
  - *tunnel_average_factor*: the averaged (over the barrier) enhancement factor applied to the Drift-Diffusion current due to tunneling
  - *tunnel_enhance_factor*: the peak (at barrier maximum) enhancement factor applied to the Drift-Diffusion current due to tunneling
  - *tunnel_peak_transmittance*: the peak transmittance of the tunneling region
  - *tunnel_peak_energy*: the energy at which the tunneling transmittance peaks
  - *tunnel_above_bar_refl*: the reflection coefficient for carriers above the tunneling barrier
  - *tunnel_negf_current*: the tunneling current from the NEGF model, independent from terminal/electrode current.
- **tunnel_region** is the tunneling region used for the export, numbered according to the order of the **tunneling** statements in the .sol input file.

22.104 bias_output_wave_average

parameter	data type	values [defaults]
variable	char	
active_reg_only	char	[no], yes
inactive_reg_only	char	[no], yes
xrange	realx2	(μm)
yrange	realx2	(μm)
zrange	realx2	(μm)
wave_mode_index	intg	[1]
long_mode_index	intg	[1]

bias_output_wave_average replaces the previous **more_bias_output** command; this command is specialized to output the average of a variable, weighted by the optical mode profile. See also the following related commands which also replace **more_bias_output**:

- **bias_output_curr_flux**
- **bias_output_tunneling**
- **bias_output_longitudinal**
- **bias_output_at_maximum**
- **bias_output_near_point**
- **bias_output_spatial_integral**

These various commands generate additional bias-dependent data for later plotting using the **plot_scan** command. Each statement generates a new plotting variable stored in *more_bias_vari,i=1..15*; the numbering corresponds to the order of the various bias_output commands in the .sol input file.

Parameters

- **variable** is the variable being exported. This command supports two categories of variables:
  - Quantities averaged over the lateral wave profile:
    - *optical_loss*: the total optical loss
    - *optical_loss_ncarr*: the optical loss due to electron free carrier absorption
    - *optical_loss_pcarr*: the optical loss due to hole free carrier absorption
    - *optical_loss_two_photon*: the optical loss due to the two-photon absorption process
    - *optical_loss_two_photon_carr*: the optical loss from free carriers generated during the two-photon absorption process
    - *optical_loss_backg*: the optical loss due to background scattering
    - *optical_loss_bandtoband*: the optical loss from band-to-band transitions.

    - *optical_gain_bandtoband*: the optical gain from band-to-band transitions.
    - *optical_loss_mirror*: the optical mirror loss in Fabry-Perot lasers
    - *modal_index_change*: the modal effective index change vs. the equilibrium value
  - Quantities averaged over the longitudinal wave profile:
    - *alpha_l_over_2*: $\frac{\alpha L}{2}$ from the coupled round-trip gain model in PICS3D
    - *delta_beta_l*: $\delta\beta L$ from the coupled round-trip gain model in PICS3D
    - *density_mode_average*: electron carrier density averaged over lateral mode
    - *kappa_l_real*: real part of the κL coupling coefficient
    - *kappa_l_imag*: imaginary part of the κL coupling coefficient
    - *phase_index*: effective index of the optical mode

  Please note that most of the variables that are longitudinally averaged in this command are, *by definition*, already averaged over the lateral mode.

- **active_reg_only** restricts the integration to optically active regions.
- **inactive_reg_only** restricts the integration to optically passive regions.
- **xrange**, **yrange** and **zrange** define the integration range.
- **wave_mode_index** is the lateral mode index used to average data in the x,y plane.
- **long_mode_index** is the longitudinal mode index used for the average along z.

22.105 blank_area

parameter	data type	values [defaults]
point1	realx2	[0. 0.] (μm)
point2	realx2	[0. 0.] (μm)
point3	realx2	[0. 0.] (μm)
point4	realx2	[0. 0.] (μm)
num_points	intg	4

This statement can be used repeatedly to blank out areas in plotting statements in the .plt files. It is especially useful when areas with no material ("void" material) show non-physical current flows.

Each statement can only blank out one area of quadrilateral or triangle.

Parameters

- **pointk,k=1..4** is the x and y coordinates of point k in counter-clockwise order.
- **num_points** takes either 4 or 3, representing quadrilateral or triangle.

Examples

```
blank_area point1=(0. 0.) point2=(1. 0.)  &&
  point3=(1. 0.5) point4=(0. 0.5)
```

22.106 bottom_contact

left_contact is the same as **top_contact** except that the contact is placed at the bottom.

22.107 boundary_smooth

parameter	data type	values [defaults]
mesh_inf	char	
mesh_outf	char	
order	char	[yes],no

The mesh generation statement **boundary_smooth** is used to make the extension of added mesh lines (by using **double_mesh**) from one polygon to the adjacent one smooth. This prevents the accumulation of too many terminated mesh lines extending to the adjacent polygon.

Note that this statement is used only when the user wants to manually refine the mesh in a region of the device geometry. In most cases, this statement is not needed because the mesh may be automatically refined using **refine_mesh**.

Parameters

- **mesh_inf** is the mesh input file.

- **mesh_outf** is the mesh output file.

- **order** specifies the format of the mesh output. If the mesh output from this statement is directly to be used by the solver, choose *yes*. Choose *no* if further manipulations are required.

Examples

```
boundary_smooth mesh_inf=case1.msh1 &&
   mesh_outf=case1.msh2 order=yes
```

22.108 boundary_xpoint

parameter	data type	values [defaults]
xp_size	real	[0.0001] (um)
boundary	intg	[1]

The statement **boundary_xpoint** is used to control the internal extra point at a contact. The mesh spacing at a Schottky contact is important since the simulator treats the contact mesh point as a point at the metal while the next mesh point inside is regarded as the semiconductor. Therefore, the mesh spacing defines the sharpness of the interface and the Schottky barrier height.

Thus we need to allocate an extra point near every Schottky contact. If a contact is defined as Schottky contact in .layer file, this statement will be automatically generated in the .geo file. The distance at which the extra point is placed can be set in the contact definition in the .layer file.

Parameters

- **boundary** is the contact or electrode number.

- **xp_size** is the distance at which the extra point is placed. It is recommended that a value of around 1 Åbe used unless it causes convergence problems. In the limit of zero, the internal numerical Schottky barrier height is exactly as defined by the user in the **contact** statement.

22.109 bpm

parameter	data type	values [defaults]
go_backward	char	[yes],no
update	char	yes,[no]
paraxial_model	char	[yes],no
symmetrize	char	yes,[no]
fft	char	[yes],no
initialize_backward	char	[no],yes
coupled_mode	char	[no],yes
zstep_size	real	[0.1]
x_ratio	real	[1.0]
y_ratio	real	[1.0]
x_center	real	[1.0]
y_center	real	[1.0]
transm_xrange	realx2	(μm)
transm_yrange	realx2	(μm)
xsize_power2	intg	6
ysize_power2	intg	6
smooth_step	intg	0
update_per_scan	intg	
round_trip	intg	1
ipade	intg	[0]
semi_vect	intg	[0]
nx_mesh	intg	[40]
ny_mesh	intg	[40]
transm_2d_mode_zseg	intg	

bpm activates the Beam Propagation Method (BPM) to solve the wave equation in PICS3D. The BPM assumes a known wave distribution at an initial x-y plane and solves the wave distribution on the next plane in the z-direction. Repeating this procedure propagates the beam along the whole waveguide.

Please note that our BPM method maintains its own propagating power in both directions but only for the purpose of computing the lateral wave profile. Since the BPM method works on a relatively coarse mesh, the coupling with the local refractive index is not accurate enough to be used in actual lasing power computation.

Instead, the BPM method is used to obtain a lateral wave profile which is then mapped to our usual 2x2 transfer matrix plane wave model (coupled-wave theory). This transfer matrix is then solved self-consistently with the Drift-Diffusion equations

and takes into account the coupling of the gain, refractive index and DFB/DBR gratings.

This Crosslight-specific method can be viewed as a hybrid method that combines the benefits of BPM propagation with our complex round-trip gain (RTG) model.

Frequently Asked Questions

Q: Is Crosslight BPM bidirectional? A: Yes and no. The iteration method itself does not define a distributed reflection from local detailed grating but at the end of the wave guide, a reflective beam is launched with calculated intensity so that (**go_backward**=*yes*) both backward and forwards beams are present in the cavity and the lateral mode used by the solver is the sum of the two.

Q: Is BPM necessary for a tapered laser/amplifier ? A: Yes. The 2D eigenmode solver no longer gives an accurate solution when there is a longitudinal variation of the lateral waveguide dimensions. BPM works best when launched from a single mode/straight segment where the lateral profile is initialized by the usual 2D eigen mode solver.

Q: Can BPM used for tapers and DBR gratings ? A: Yes. The DBR grating is taken care of by the 2x2 transfer matrix approach (coupled-wave theory) while only the lateral wave profile from BPM is used. The hybrid approach ensures efficiency and self-consistency with local gain/index

Parameters

- **go_backward** directs the wave to be propagated backwards towards z=0 once it reaches the end of the device so it makes one or more complete round-trips (defined by **round_trip**). However, the current version of the Crosslight BPM code is not a fully bidirectional BPM method.
- **initialize_backward** tells the BPM solver to use the mode profile to initialize the backwards propagation rather than the forward propagated wave. In general, this setting should not be used unless the forward wave output is heavily distorted.
- **coupled_mode** enables use of the **bpm_coupled_mode** statement.
- **update** tells the software to update the results of the BPM method during the simulation. The BPM is recalculated automatically as needed when the bias increases but this can be explicitly controlled using **update_per_scan**.
- **paraxial_model** determines whether a paraxial approximation is used in the BPM code.

- **symmetrize** may be used to artificially force a symmetric BPM output.

- **zstep_size** is the BPM step size (in units of wavelength) within the semiconductor medium.

- **fft** switches between an older FFT-BPM algorithm and a newer FD-BPM method.

  For the FFT-BPM method, the mesh size must be a power of two: **xsize_power2** and **ysize_power2** set the exponent of that power of two in the x and y direction, respectively. Otherwise (FD-BPM), the number of mesh points in the x and y direction are defined using **nx_mesh** and **ny_mesh**. These points are distributed along center points **x_center** and **y_center** using mesh ratios **x_ratio** and **y_ratio**. This system is somewhat similar to the way mesh point ratios are allocated in the layer file and in **put_mesh**.

- **smooth_step** is used to transfer solution of regular mesh to BPM mesh smoothly. Usually the regular simulation mesh is non-uniform and much different from the uniform BPM mesh. Since the initial solution is provided on the regular mesh, transferring it BPM mesh does not give a smooth solution. Using this parameter (can be set to several hundred) will allow the wave to be propagated with the uniform mesh (without taper) until it becomes smooth and can be regarded as a real solution on the BPM mesh.

- **ipade** is the order of the Padé approximation used.

- **semi_vect** specifies the semi-vectorial settings for the BPM method:

  0 scalar BPM

  1 semi-vectorial BPM with TE mode

  2 semi-vectorial BPM with TM mode

- **transm_xrange** and **transm_yrange** are used to compute the transmission factor by integrating the BPM mode over a xy window. As an alternative, the mode profile in a particular z-segment specified by **transm_2d_mode_zseg** may be used to compute the transmission using a mode overlap criterion.

Examples

```
bpm zstep_size=0.2 xsize_power2=5 ysize_power2=5 update=yes
```

22.110 bpm_coupled_mode

parameter	data type	values [defaults]
z_from	real	(μm)
z_to	real	(μm)
mode_num	intg	[1]
zseg_num	intg	

bpm_coupled_mode works in conjunction with the **bpm** statement; it enables a filtering model that restricts the shape of the BPM mode at each propagation point z_{BPM}. Instead of the natural output of the BPM method, we use an approximate solution that is based on the shape of the eigenmodes present in the simulation; this has the advantage of eliminating spurious elements of the BPM solution which may accidentally propagate.

We describe this process by writing our filtered BPM mode shape (using bra-ket notation) as:

$$\left|w^f_{BPM}>=\sum_{i=1}^{N} a_i\left|w_{Ai}>+\sum_{j=1}^{N} b_j\left|w_{Bj}>\right.\right. \tag{22.6}$$

where $|w_{Ai}>$ are the sets of eigenmodes present at $z = z_A$ and $|w_{Bj}>$ is another set of modes present at $z = z_B$. The A and B reference planes are chosen to be the z-segments (c.f. Chap 6.3) surrounding the current BPM propagation point so that $z_A <= z_{BPM} <= z_B$. We note that the eigensolver is only called once per z-segment and not once for each electrical mesh plane.

To obtain the coefficients of the expansion, we left-multiply the above by $<w_{Ai}|$ and $<w_{Bj}|$ to obtain a set of linear equations; when solved, this yields a set of a_i, b_j coefficients which minimize the error between the actual BPM solution and the filtered mode shape.

The filtered BPM mode is then used as the initial value for the next propagation step; this process is repeated for all selected BPM steps. When the BPM propagates in reverse, the same method is used but an independent set of expansion coefficients is obtained.

Parameters

- **z_from** and **z_to** describe the range of BPM steps where the filtering occurs; outside this range, the standard BPM mode shape is used as-is. If multiple independent ranges need to be filtered, the **bpm_coupled_mode** statement may be issues multiple times in the .sol input file.

- **mode_num** describes the number of eigenmodes used in the filtering expansion. If **zseg_num** is omitted, this number is used for all z-segments; if not, this number is set only for the specified z-segment.

22.111 bpm_initial_import

parameter	data type	values [defaults]
intensity	char	[power_arb_unit.txt]
phase_in_pi	char	[void]

bpm_initial_import works in conjunction with the **bpm** statement. By default, the software uses the eigenmode profile at $z = 0$ as the initial wave profile for the BPM method; by using this statement, the user may import an external wave profile as this initial wave.

Parameters

- **intensity** is the name of a text file containing the BPM mode shape being imported. This data should be stored in a GnuPlot-compatible format with three columns: x,y and arbitrary power/intensity value.

- **phase_in_pi** is the name of a text file containing the local phase of the imported field; if empty, the phase factor is set to zero. This data should be stored in a GnuPlot-compatible format with three columns: x,y and the local phase value expressed as a multiple of π.

22.112 bpm_longmode_splot

parameter	data type	values [defaults]
data_file	char	
variable	char	
plane	char	[xy], xz, yz
cut_near	real	[0.0] (μm)
longmode	intg	

bpm_longmode_splot does a surface plot of the Beam propagation Method wave solution and other relevant model information. Unlike other BPM plotting commands, this statement ***does not*** fit inside a **begin_bpmplot** block: instead, it is meant to be used alongside regular .plt post-processing commands.

Parameters

- **data_file** may be used to save a copy of the plot data to a text file.
- **variable** is the variable being plotted:
  - *BPM_fwd_wave_intensity* is the forward wave intensity
  - *BPM_fwd_wave_phase* is the forward wave phase factor
  - *BPM_bck_wave_intensity* is the backward wave intensity
  - *BPM_bck_wave_phase* is the backward wave phase factor
  - *BPM_real_index* is the real part of the refractive index used for propagation
  - *BPM_imag_index* is the imaginary part of the refractive index used for propagation
  - *BPM_fwd_longmode_intensity* is the forward intensity of a given longitudinal mode.
  - *BPM_bck_longmode_intensity* is the backward intensity of a given longitudinal mode.
- **plane** controls which plane is used for the surface plot. The position of the cut plane along the remaining axis is determined by **cut_near**.
- **longmode** controls which longitudinal mode is used in the plot. If this parameter is omitted, the plot will show a sum over all longitudinal modes.

22.113 bpm_multimode

parameter	data type	values [defaults]
initial_amplitude	real	[0.2]
initial_phase	real	[0.0]
mode_index	intg	[2]

bpm_multimode is used to define an initial optical field profile consisting of more than one lateral mode for the beam propagation method.

Parameters

- **initial_amplitude** is the initial amplitude of mode profile relative to the fundamental mode.
- **initial_phase** is the initial phase (as a multiple of π) of the mode profile relative to the fundamental mode.
- **mode_index** is the mode index label (integer) of the lateral mode.

Examples

The following statements define a profile with 3 modes, with 2nd and 3rd order modes of amplitude being 50 percent and 10 percent of the 1st order mode, respectively.

```
bpm_multimode initial_amplitude=0.5 initial_phase=0 mode_index=2
bpm_multimode initial_amplitude=0.1 initial_phase=0 mode_index=3
```

22.114 bpmintegr_xy

parameter	data type	values [defaults]
outfile	char	[void]
backward	char	[no]
xrange	realx2	
yrange	realx2	
z	real	[0.0]

bpmintegr_xy may be used to integrate the Beam Propagation Method (BPM) wave intensity over an x-y plane. This statement must be used inside a **begin_bmplot** block.

Parameters

- **outfile** is the output file used to save the result of the numerical integration
- **backward**, if enabled, integrates the backward wave instead of the forward wave.
- **xrange** and **yrange** are the integration ranges in the x and y direction, respectively.

- **z** is the position of the x-y plane.

Examples

```
bpmintegr_xy z=10. outfile=tmp.data3
```

22.115 bpmplot_xy1d

parameter	data type	values [defaults]
direction	char	[y],x
plot_phase	char	[no]
backward	char	[no]
rel_cutpoint	real	[0.5]
xrange	realx2	
yrange	realx2	
z	real	[0.0]

bpmplot_xy1d is used to plot a 1D cut of the Beam Propagation Method wave solution on a specified x-y plane. This statement must be used inside a **begin_bmplot** block.

Parameters

- **direction** is used to indicate the direction of the cut-line.
- **contour** indicates if contour plot is required.
- **plot_phase** indicates whether the phase or intensity is plotted.
- **backward**, if enabled, plots the backward wave instead of the forward wave.
- **rel_cutpoint** is the relative cut point position.
- **xrange** and **yrange** are the plot ranges in the x and y directions, respectively.
- **z** is the position of the x-y plane.

Examples

```
bpmplot_xy1d direction=y rel_cutpoint=0.5 z=105 plot_phase=no backward=no
```

22.116 bpmplot_xyz1d

parameter	data type	values [defaults]
plot_phase	char	[no]
backward	char	[no]
rel_x	real	[0.5]
rel_y	real	[0.5]
xrange	realx2	
yrange	realx2	

bpmplot_xyz1d is used to plot the Beam Propagation Method wave solution on a line along the z-direction. This statement must be used inside a **begin_bmplot** block.

Parameters

- **plot_phase** indicates whether the phase or intensity is plotted.
- **backward**, if enabled, plots the backward wave instead of the forward wave.
- **rel_x** and **rel_y** are respectively the relative x and y positions of the cut line.
- **xrange** and **yrange** are the plot ranges in the x and y directions, respectively.

Examples

```
bpmplot_xyz1d plot_phase=no backward=no
```

22.117 bpmsplot_xy

parameter	data type	values [defaults]
contour	char	[no]
plot_phase	char	[no]
backward	char	[no]
view_xrot	real	[0.]
view_zrot	real	[0.]
xrange	realx2	
yrange	realx2	
zrange	realx2	
z	real	[0.0]

bpmsplot_xy is used to plot the Beam propagation Method wave solution on an specified x-y plane. This statement must be used inside a **begin_bmplot** block.

Parameters

- **contour** indicates if contour plot is required.
- **plot_phase** indicates whether the phase or intensity is plotted.
- **backward**, if enabled, plots the backward wave instead of the forward wave.
- **view_xrot** and **view_xrot** rotate the surface plot along the x and z axes, respectively.
- **xrange**, **yrange** and **zrange** are the plot ranges in the x, y and z directions, respectively.
- **z** is the position of the x-y plane.

Examples

```
bpmsplot_xy z=400 contour=no plot_phase=yes backward=no
```

22.118 bpmsplot_xz

bpmsplot_xz is used to plot the Beam propagation Method wave solution on an specified x-z plane. This statement must be used inside a **begin_bmplot** block.

Parameters

With the following exceptions, most parameters for this statement are identical to those of **bpmsplot_xy**:

- **rel_y** is the relative y position of the x-z plane.
- **abs_y** is the absolute y position of the x-z plane (in microns). This parameter will override **rel_y**.

22.119 bpmsplot_yz

bpmsplot_yz is used to plot the Beam propagation Method wave solution on an specified y-z plane. This statement must be used inside a **begin_bmplot** block.

Parameters

With the following exceptions, most parameters for this statement are identical to those of **bpmsplot_xy**:

- **rel_x** is the relative x position of the y-z plane.
- **abs_x** is the absolute x position of the y-z plane (in microns). This parameter will override **rel_x**.

22.120 bulk_dos_model

parameter	data type	values [defaults]
carrier	char	[electron],hole
model	char	[3d],2d
mater_label	char	
2d_thick	real	[0.01] (μm)
2d_barrier	real	[0.3] (eV)
scale	real	[1.]
mater	intg	[1]

This statement is used to modify the density of states used in a bulk layer. The most common application is when a thick MQW region is approximated by a single

average bulk material: the density of states in that case should be that of the original quantum wells and not that of the bulk material.

Parameters

- **carrier** is the carrier type whose density of states will be modified.
- **model** changes between the 2D (QW) and 3D (bulk) DOS models.
- **2d_thick** is the thickness of the QW to be used in the 2D DOS model.
- **2d_barrier** is the barrier height of the QW to be used in the 2D DOS model.
- **scale** is a scaling factor for the DOS.
- **mater** is the number identifying the material being modified. If a label alias has previously been defined for this material, **mater_label** may be used instead.

Examples

```
bulk_dos_model model=2d carrier=electron mater=2
```

22.121 bulk_treatment

parameter	data type	values [defaults]
type	char	[void]n,p

The statement **bulk_treatment** is used to turn on/off the quantum treatment for one type of carrier. By default, both electron and hole subbands of a quantum well are solved simultaneously. If one wish to achieve faster simulation or easier convergence, one may use this statement to skip the quantum treatment and treat the less important carrier as if it were in a bulk region.

Parameters

- **type** is the type of carrier to be treated as bulk carrier without quantum models.

Examples

```
bulk_treament type=p
```

The above statement will instruct the program to only compute the quantum states of the n-type carriers.

22.122 bulk_xfunc1

bulk_xfunck,k=1..9 are a set of external functions that can be used as part of a passive macro declaration. They can for example, define a bandgap formula that is reused in the declaration of different parameters.

These statements follow the same rules for functions defined for other material macro statements. See section 22.456 for examples and further details.

22.123 bulk_xfunc2

See **bulk_xfunc1**.

22.124 bulk_xfunc3

See **bulk_xfunc1**.

22.125 bulk_xfunc4

See **bulk_xfunc1**.

22.126 bulk_xfunc5

See **bulk_xfunc1**.

22.127 bulk_xfunc6

See **bulk_xfunc1**.

22.128 bulk_xfunc7

See **bulk_xfunc1**.

22.129 bulk_xfunc8

See **bulk_xfunc1**.

22.130 bulk_xfunc9

See **bulk_xfunc1**.

22.131 c11_bar

cij_bar are a set of statements defining the stiffness tensor (elastic constants) of the barrier material in the active macro of a quantum well. It is necessary to define strain effects on bandgap and other material properties.

See **c11_well** for a list of tensor elements used in zincblende and wurtzite materials.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.132 c12_bar

See **c11_bar**

22.133 c13_bar

See **c11_bar**

22.134 c33_bar

See **c11_bar**

22.135 c44_bar

See **c11_bar**

22.136 c11_bulk

cij_bulk are a set of statements defining the stiffness tensor (elastic constants) in a passive material macro. This is only used for wurtzite materials since the software does not support strained bulk zincblende.

See **c11_well** for a list of applicable tensor elements.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.137 c12_bulk

See **c11_bulk**

22.138 c13_bulk

See **c11_bulk**

22.139 c33_bulk

See **c11_bulk**

22.140 c44_bulk

See **c11_bulk**

22.141 c11_well

cij_well are a set of statements defining the stiffness tensor (elastic constants) in the active macro of a quantum well or bulk active region. It is necessary to define strain effects on bandgap and other material properties.

For zincblende, the following C_{ij} values are used[118]:

$$\begin{pmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & 4C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & 4C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & 4C_{44} \end{pmatrix} \tag{22.7}$$

For wurtzite, the following C_{ij} combinations are used:

$$\begin{pmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{13} & 0 & 0 & 0 \\ C_{13} & C_{13} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & 4C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & 4C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\left(C_{11} - C_{12}\right) \end{pmatrix} \tag{22.8}$$

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.142 c12_well

See **c11_well**

22.143 c13_well

See **c11_well**

22.144 c33_well

See **c11_well**

22.145 c44_well

See **c11_well**

22.146 column

parameter	data type	values [defaults]
indep_xmqw	char	[no] yes
add_top_mesh	char	[no] yes
w	real	
r	real	[1.]
column_num	intg	[1]
mesh_num	intg	[6]
shift_center	real	[0.] (fraction)

column is the statement used in the .layer file to define a column in the cross section of the device; it also defines the mesh allocation in the vertical direction within said column.

In that sense, this command is analogous to the **layer** statement which defines horizontal properties.

Parameters

- **indep_xmqw** may be used to force the active layers in different columns to be treated as different (or independent) active regions even if the material parameters are the same. Such treatment may be necessary if the potential distribution in the two columns are substantially different in a self-consistent simulation.
- **add_top_mesh** instructs the software to define a mesh line boundary at the top of the column. By default, this boundary is defined at the bottom of the column so if a void region is present in a column, this may cause issues for the mesh generator.
- **column_num** is a number assigned to the column for identification purposes.
- **w** defines the width of the column.
- **mesh_num** sets the number of vertical mesh lines for this column. This mesh line boundary is processed by the mesh generator.
- **r** is the ratio for mesh line distribution. See Fig. 22.22.
- **shift_center** is used when **r** is negative and a symmetric mesh distribution is requested. This parameter moves the position of the point around which the mesh is symmetric.

Examples

```
column column_num=1 w=1.0 mesh_num=6 r=0.8
```

22.147 column_position

parameter	data type	values [defaults]
label	char	
hline location	char	[right]
hline delta_x_from_left	real	[-9999.] (μm)
delta_x_from_right	real	[-9999.] (μm)

This statement is similar to **layer_position** and is used to mark a specific x coordinate for later use. This is useful to automatically track changes to a device structure without having to redefine coordinates in other commands.

All the parameters from this statement are analogous to those of **layer_position**. The main difference is that all positions are defined relative to the preceding **column** statement.

22.148 compact_junction_region

parameter	data type	values [defaults]
data_column	char	
data_folder	char	
data_file	char	[compact_junction_data.csv]
materi_label (i=1...9)	char	
yrange	realx2	
materi (i=1...9)	intg	

compact_junction_region works in conjunction with **compact_semiconductor_model** and essentially allows the MQW region of an optical device to be modeled like a "black box" using experimental data curves. It instructs the software to load a data file that contains optical emission information that would normally be computed to solve current continuity equations but which is unavailable in the compact semiconductor model.

Parameters

- **materi (i=1...9)**, with "i" as a placeholder number, define the different material numbers that are part of the semiconductor junction and which are affected by this command. Alternatively, pre-existing material labels can be defined using **materi_label (i=1...9)**, with "i" as a placeholder number again.
- **yrange** is the spatial extent of the region affected by this command.
- **data_file** defines the name of data file being loaded. The data must be stored in a comma-separated value (CSV) text format.
- **data_folder** defines the path to the data file being loaded.
- **data_column** specifies the column format for the CSV file. At present, the following settings are possible:
  - *v_j_iqe/temp* is a 3-column format with the first line containing column headers that will be ignored. The first column is the voltage, the second column is the current and the third column is the internal quantum efficiency (IQE) of a LED.

    Multiple curves at different temperatures can be included in the same file. The 3-column data blocks should be separated by a single-column line containing the new temperature; the first data block is assumed to be at 300K.
  - *j_gain/temp* is a 2-column format with the first line containing column headers that will be ignored. The first column is the current and the second column is the optical gain; this format is primarily used for lasers.

    Multiple curves at different temperatures can be included in the same file. The 2-column data blocks should be separated by a single-column line containing the new temperature; the first data block is assumed to be at 300K.

22.149 compact_semiconductor_model

parameter	data type	values [defaults]
set_min_elec_density	real	(m^{-3})
set_min_hole_density	real	(m^{-3})
set_max_resistivity	real	$(\Omega \cdot m)$

compact_semiconductor_model turns on a simplified electrical transport model for the simulation: instead of solving the coupled drift-diffusion equations of Chapter 5, only the Poisson equation is solved in the main Newton solver. The reduction in problem size that comes from solving only one equation greatly simplifies convergence and speeds up the computation speed in large 3D problems.

Since this approach does not solve the current continuity equations, only the drift current is available in this model: this current is obtained from the gradient of the potential, assuming an ohmic resistor model for all points. The local equivalent resistivity is computed from the carrier mobility defined in the material macros and the carrier density from the solution of the Poisson equatio. This conversion can be shown using the definition of the net drift-diffusion current in Eq. 5.6:

$$\begin{aligned} J_n &= n\mu_n \nabla E_{fn} \\ &\approx n\mu_n \nabla V \\ &\approx \frac{\nabla V}{\rho_n} \end{aligned}$$

from which we obtain $\rho_n = \frac{1}{n\mu_n}$.

As this model is an extreme simplification, using very small voltage steps in the **scan** command is ***strongly*** recommended to reduce the variation of carrier density between scan steps and obtain a smooth I-V curve.

Parameters

- **set_min_elec_density** sets the minimum value of the electron carrier density used to obtain the local resistivity: this is used to avoid dividing by zero during the conversion. **set_min_hole_density** is likewise the minimum value of the hole carrier density used in the conversion.

- **set_max_resistivity** artificially caps the value of the local resistivity. This cap thus takes into account small values in both the carrier density and carrier mobility.

22.150 complex_region

parameter	data type	values [defaults]
polygon_name	char	
mater_label	char	
thickness	real	[0.01] (μm)
x_size	real	(μm)
mater	intg	[1]

The statement **complex_region** is used to describe a rectangle region of material within a complex; that is a group of layers coupled to each other quantum mechanically. Note that there should be odd number of layer for each column due to historical reasons: the convention is that a complex describes a coupled MQW system in the sequence of barrier-well-barrier...well-barrier. As of the 2012 version, there is a special exception to this rule which can be used to define a complex region with only two layers: see **begin_complex** for details.

Also for historical reasons, the optical gain is only calculated in the even-numbered layers of a complex. This can be changed with **inner_bar_gain** but only for the "inner" barriers: the gain will not be calculated in the outer layers and only serve as boundary regions where the wavefunction decays. The user may split certain layers into smaller pieces to account for this.

Note that complex MQW region declarations are usually generated automatically by the layer.exe program. All that is needed is to use the appropriate complex MQW active macros (prefixed by "cx-") in the **layer_mater** statement.

Parameters

- **polygon_name** is the name of the polygon as defined in the .geo file.
- **thickness** is the thickness of the region in microns.
- **x_size** is the width of the region in microns.
- **mater** is the material number of the region. If a label has previous been defined for this material, **mater_label** may be used instead.

Examples

```
begin_complex layer_num=    5 column_num=    1
complex_region polygon_name=p002 mater=    2  thickness=  0.785000000000E-001
```

```
complex_region polygon_name=p003 mater=    3  thickness=  0.750000000000E-002
complex_region polygon_name=p004 mater=    2  thickness=  0.850000000000E-002
complex_region polygon_name=p005 mater=    3  thickness=  0.750000000000E-002
complex_region polygon_name=p006 mater=    2  thickness=  0.785000000000E-001
end_complex
```

22.151 complex_var_symbol

parameter	data type	values [defaults]
var_lib	char	
var_complex	char	

complex_var_symbol translates a variable/symbol name from the library to the underlying complex/active macro. This statement can be omitted if the variable name is the same in both the library and the macro.

See **material_lib** and **basic_var_symbol** for further information.

Examples

```
begin_library AlGaAs
import_basic name=algaas
import_complex name=cx-AlGaAs
complex_var_symbol  var_lib=x var_complex=xw
end_library
```

This set of commands defines the "AlGaAs" library as being composed of the "algaas" passive macro and the "cx-AlGaAs" active macro. The material parameter x used when invoking the library is translated into the xw parameter of the active macros and used "as-is" in the passive macro.

22.152 compute_inductance

parameter	data type	values [defaults]
ref_contact	intg	[1]
critical_curr	real	(A/m^2)
ref_zplane	real	[8.0] (μm)

compute_inductance is a post-processing statement used to compute the magnetic inductance ($L = \frac{d\Phi}{di}$) of the device. The software will compute the magnetic flux density ($B = \nabla \times A$) based on Ampere's law ($\nabla \times B = \mu_0 J$) and the DC current flow in the device.

Parameters

- **ref_contact** is the contact number used to inject current in this model.
- **critical_curr** is a current density used to determine an equivalent coil geometry based on the current distribution inside the device.
- **ref_zplane** is the position of z-plane used to compute the magnetic flux Φ.

22.153 cond_band2_edge

cond_bandj_edge, j=2,3 are a set of material statements used in active macros with the **layer_type** of general_cx_strain. They define the offset (in eV) of the 2nd or 3rd conduction band valley with respect to the lowest conduction band valley.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.154 cond_band3_edge

See **cond_band2_edge**.

22.155 cond_band1_valley

cond_bandj_valley,j=1..3 are a set of material statements used in active macros with the **layer_type** of general_cx_strain. When used, they override the number of band valleys defined for the conduction bands in that statement.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.156 cond_band2_valley

See **cond_band1_valley**.

22.157 cond_band3_valley

See **cond_band1_valley**.

22.158 cond_dos_mass_ratio_n

The statements **cond_dos_mass_ratio_n** and **cond_dos_mass_ratio_p** are used to define the ratio:

$$\text{ratio} = \frac{\text{conduction mass}}{\text{DOS mass}}$$

for electrons and holes, respectively. This allows the user to indirectly set the conduction mass of a material.

Theory

In the macros, the mass definitions are related to the density of states (DOS) and the curvature of the energy dispersion relation: i.e., the DOS mass is a measure of "how many" carriers there are in the material.

On the other hand, tunneling transparency (see Sec. 9.2) is derived from the Schrõdinger equation. This means that the mass used in the WKB approximation and other models originates from a kinetic energy term: it is a measure of "how fast" the carriers are. This is often called the conduction mass since it can be related to the carrier mobility.

We note however that since the software directly uses the mobility as a macro variable, the conduction mass is not used in the drift-diffusion model.

Parameters

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.159 cond_dos_mass_ratio_p

See Sec. 22.158

22.160 cond1_mass_para

condj_mass_para,j=1..2 are a set of active macro parameters defining the relative conduction band mass of valley j in a direction parallel to the quantum well[119].

The mass defined in this statement can also be modified by non-parabolic terms with the **condj_para_e_dep_mass1** and **condj_para_e_dep_mass2** statements:

$$\begin{aligned} m(E) &= a + bE + cE^2 \\ E(k) &= \frac{\hbar^2 k^2}{2m_0 m(E)} \end{aligned}$$

Note that the j placeholder value must be the same in all thee statements since they all refer to the same band valley.

Parameters

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

Examples

This statement is used for general complex strained macros; the most common applications are for strained silicon, SiGe and II-VI lead salt materials.

```
layer_type type=general_cx_strain valley_c1=4 valley_v1=4 &&
  optic_trans_valley_pair1=(1 1)

$ Define band gap as a bulk_xfunc1, to be referred to later by other functions
ext_func1 variation=function
function(x,temper)
0.17+0.057*x-0.095*x**2+sqrt(4.e-4+2.56e-7*temper**2)
end_function

cond1_mass_para value=0.05
```

```
cond1_para_e_dep_mass1 variation=function
function(x)
em_parab=0.05;
egt=ext_func1;
2.*em_parab/egt
end_function
```

This defines a mass equal to $m(E) = 0.05 + 0.1\frac{E}{E_g}$ in the notation above.

22.161 cond2__mass__para

Se **st:cond1__mass__para**.

22.162 cond1__mass__perp

condj__mass__perp,j=1..2 are a set of active macro parameters defining the relative conduction band mass of valley j in a direction perpendicular to the quantum well[119].

The mass defined in this statement can also be modified by non-parabolic terms with the **condj__perp__e__dep__mass1** and **condj__perp__e__dep__mass2** statements:

$$\begin{aligned} m(E) &= a + bE + cE^2 \\ E(k) &= \frac{\hbar^2 k^2}{2m_0 m(E)} \end{aligned}$$

Note that the j placeholder value must be the same in all thee statements since they all refer to the same band valley.

Parameters

The parameters for this statement are the same as for all other material statements. See **material__par** in section 22.456 for examples and further details.

Examples

This statement is used for general complex strained macros; the most common applications are for strained silicon, SiGe and II-VI lead salt materials.

```
layer_type type=general_cx_strain valley_c1=4 valley_v1=4 &&
  optic_trans_valley_pair1=(1 1)

$ Define band gap as a bulk_xfunc1, to be referred to later by other functions
ext_func1 variation=function
function(x,temper)
0.17+0.057*x-0.095*x**2+sqrt(4.e-4+2.56e-7*temper**2)
end_function

cond1_mass_perp value=0.05

cond1_perp_e_dep_mass1 variation=function
function(x)
em_parab=0.05;
egt=ext_func1;
2.*em_parab/egt
end_function
```

This defines a mass equal to $m(E) = 0.05 + 0.1\frac{E}{E_g}$ in the notation above.

22.163 cond2_mass_perp

Se **st:cond1_mass_perp**.

22.164 cond1_para_e_dep_mass1

See **cond1_mass_para**.

22.165 cond1_para_e_dep_mass2

See **cond1_mass_para**.

22.166 cond2_para_e_dep_mass1

See **cond1_mass_para**.

22.167 cond2_para_e_dep_mass2

See **cond1_mass_para**.

22.168 cond1_perp_e_dep_mass1

See **cond1_mass_perp**.

22.169 cond1_perp_e_dep_mass2

See **cond1_mass_perp**.

22.170 cond2_perp_e_dep_mass1

See **cond1_mass_perp**.

22.171 cond2_perp_e_dep_mass2

See **cond1_mass_perp**.

22.172 cond1_valley_prop1

condj_valley_propk are a set of active layer macro statements: j and k are placeholder values that indicate the conduction subband valley number (j=1..2) and the property number (k). For each valley, the subband concentrations are used to compute material properties.

This parameter is not directly used for simulation but it allows for printing of data in the post-processing stage. The statement **more_output** must be used to specifically turn on the printing of this data.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.173 cond2_valley_prop1

See **cond1_valley_prop1**.

22.174 contact

parameter	data type	values [defaults]
type	char	[ohmic], schottky
touch_macro	char	
touch_var_symbolk(k=1..0)	char	
mater_label	char	
auto_numbering	char	[no],yes
barrier	real	(eV)
junc_coefs	realx2	[1. 1.]
elec_temp	real	[1.5]
hole_temp	real	[1.5]
lattice_temp	real	[300.] (K)
heat_flow	real	[100.] (3D: W, 2D: W/m)
thermal_cond	real	[10.] (3D: W/K, 2D: (W/m)/K)
extern_temp	real	[300].
work_function	real	(eV)
touch_vark(k=1..9)	real	
num	intg	
thermal_type	intg	[1]
touch_mater	intg	

The statement **contact** defines the properties of an equipotential region belonging to an electrode or metal contact. It does not define any other properties associated with a metal contact such as optical absorption: this can be handled by using real metal layers in the device or other statements relevant to a particular model.

For ohmic contacts, the simulation program must determine a built-in voltage based on the flat band condition of semiconductor material in contact with the electrode. Therefore, the program will get confused if an ohmic contact is in contact with more than one semiconductor materials. In such a case, a solution is to use the parameters started with "touch_" to force the electrode to use the properties of a specific semiconductor.

In all thermal simulations, it is strongly recommended to override automatic contact definitions from the .layer file in order to guarantee that the correct thermal boundary

conditions are applied to the simulation. The default values may not make sense in all cases.

Parameters

- **type** defines the type of the contact.
- **touch_mater** is used to specify the material number used to compute the contact boundary conditions. For example, an ohmic contact touching two different materials cannot be at two different energy levels simultaneously so it only "touches" one of them. **mater_label** can also be used to identify the "touch" material if the material has previously been given an alias.

  The "touch" material can also be detected automatically by the software using the name of the material macro (**touch_macro**) and the macro parameters (**touch_var_symbolk(k=1..9)** and **touch_var_symbolk(k=1..9)**).
- **auto_numbering** may be used to automatically assign numbers to contact based on the local doping profile and "touch" material composition.
- **barrier** is the barrier height in eV which is used only for Schottky contacts. It is equal to semiconductor affinity minus the work function of the metal. The user should define either **barrier** or **work_function** but not both.
- **junc_coefs** are the Schottky junction coefficients of a contact used in the formulas of thermionic emission current model. They scale the emission velocities of electrons and holes, respectively.
- **num** is the contact number. It much match the boundary number information from the .geo file and the mesh.
- **elec_temp** and **hole_temp** are the electron and hole temperatures at the contact (in units of kT). This is used as the boundary condition for the hydrodynamic model and should not be confused with the lattice temperature which is a boundary of the thermal model.
- **lattice_temp** is the lattice temperature at the contact for thermal contact type 1.
- **heat_flow** is the heating power flow going out of the contact for thermal contact type 2. The unit of Watt/m for a 2D section of a cubic coordinate system. The unit is Watt for 3D device or a 2D section of a cylindrical system.
- **thermal_cond** is the thermal conductance connected to the contact for thermal contact type 3. The total heat flow (in Watt) to/from the device is given

by $\Sigma(T - T_{ext})$ where Σ is the thermal conductance and T_{ext} is the external thermal contact.

For a realistic 3D device, the unit is Watt/K. For a 2D simulation, the heat flow is in unit of Watt/m and the unit of Σ is Watt/m/K for a 2D section of a cubic coordinate system. That is, one must divide the total thermal conductance by the device depth (in z-direction).

For an ideal uniform device, the thermal conductance of a cubic system can be converted to that of cylindrical system by formula: $\Sigma_{cyl} = \frac{\pi R^2}{2D}\Sigma_{cub}$. R is the radius of the cylindrical device and D is the half-width of the cubic device.

If **external_cir** is used to specify the resistance of this thermal conductor, the self-heating effect will also be taken into account. A temperature difference will appear between the contact and the external heat sink.

Important Note: many users make the mistake of using the thermal conductivity (k) instead of the thermal conductance (Σ) in **thermal_cond**. These two quantities are related by:

$$\Sigma_{3D} = k\frac{w \times L}{t} \tag{22.9}$$

$$\Sigma_{2D} = k\frac{w}{t} \tag{22.10}$$

where w, L, t are the width,length and thickness of the contact, respectively. The 2D thermal conductance thus has, unfortunately, the same units of $Wm^{-1}K^{-1}$ as the thermal conductivity but the two values may differ by several orders of magnitude.

- **extern_temp** is the temperature of the heat sink connected to the thermal conductance for thermal contact type 3.

- See **barrier** above.

- **thermal_type** is the thermal boundary condition. All three types are defined in Sec. 11.5. Please also see **contact_heating** for self-heating due to contact resistance or other external circuit elements.

Examples

```
contact  num=1 type=ohmic
contact  num=2 type=schottky barrier=0.84
```

22.175 contact_heating

parameter	data type	values [defaults]
resistance	real	[0.1] (Ω)
contact_num	intg	[1]

contact_heating is used to supplement the **contact** statement in thermal simulations that use the **minispice** external circuit model. If the thermal boundary of the contact is Type 3 as defined in Sec. 11.5, this command will add a heat flow equal to the resistor's self-heating (RI^2) to the boundary condition.

Note that the resistance value defined in this command should be equal to the resistance of the path between the contact and the external heat sink, no matter how many resistors are actually connected to the contact node in the minispice circuit layout.

22.176 contact_metal_interface

parameter	data type	values [defaults]
use_fix_charge	char	[no], yes
interface_thickness	real	[1.e-3] (μm)

contact_metal_interface overrides some of the default behavior associated with electrical contact boundaries.

Parameters

- **use_fix_charge** controls whether or not fixed charges are applied on contact boundaries. This setting is relevant when dealing with ohmic contacts (which impose charge neutrality at the boundary) in wurtzite materials since a net polarization vector may be present at the outer surfaces of the device. The default behavior has the effect of automatically neutralizing the piezoelectric charge on contact boundary regions, leaving the usual ohmic behavior intact.
- **interface_thickness** is the effective thickness of the contact. Because each mesh point touching the contact may have its own effective thickness and node area, a common reference is needed so that the total charge can be collected/summed over the entire contact.

22.177 convention

parameter	data type	values [defaults]
positive_current_flow	char	[outward],inward

The command **convention** is used to define the positive current flow. By default, the simulator treats current flowing out of the device and in to a contact as being positive.

22.178 couple_input_power

parameter	data type	values [defaults]
fraction	real	[1.]
lateral_mode	intg	[1]

This statement is used in PICS3D when modeling semiconductor optical amplifiers (SOA) and modulators. It controls how much of the input light goes into each of the various lateral modes.

Parameters

- **fraction** is the fraction of power that is input into a given lateral mode.
- **lateral_mode** is the number of the lateral mode receiving a fraction of the input power.

Examples

```
$ Make sure these sum up to 1.0 ...
couple_input_power fraction=0.8 lateral_mode=1
couple_input_power fraction=0.2 lateral_mode=2
```

22.179 couple_next

parameter	data type	values [defaults]
power_loss	real	0. (fraction)
loss_modek (k=2..5)	real	0. (fraction)
sec_num	intg	[1]

The statement **couple_next** is used to define the coupling between sections of a device in PICS3D.

Parameters

- **power_loss** is the power loss fraction between sections.
- **loss_modek,k=2..5** is the power loss fraction between sections for higher order lateral mode number k.
- **sec_num** is the section number. The coupling defined by this statement is between sections **sec_num** and **sec_num**+1.

Examples

```
couple_next power_loss=0.05 sec_num=1
```

22.180 cplot_xy

parameter	data type	values [defaults]
variable	char	(see list)
data_file	char	[vttek]
mater_boundary	char	[no]yes
point_ll	realx2	(μm)
point_ur	realx2	(μm)
xrange	realx2	
yrange	realx2	
z	realx2	
level	intg	[10]
grid_sizes	intgx2	[20, 20]
mode_index	intg	[1]
trap_index	intg	[1]

cplot_xy is a post-processor statement used to plot structural data on a 2D plane. Unlike **plot_2d**, this command can only generate contour plots.

Other related commands exist and should be used depending on the 2D/3D nature of the original simulation results. The following rules apply:

- 2D simulations: use **plot_2d**
- 3D cylindrical simulations with one mesh plane: use **plot_2d**
- xy plane from a 3D simulation: use **cplot_xy** for contour plots of scalar variables, **splot_xy** for 3D surface plots of scalar variables or **vplot_xy** for vector variables.
- xyz plane from a 3D simulation: use **cplot_xyz** for contour plots of scalar variables, **splot_xyz** for 3D surface plots of scalar variables or **vplot_xyz** for vector variables.

Parameters

With the exception of the z-position and vector plot-related parameters that do not apply, all parameters are the same as in **plot_2d**.

- **z** is the position on the z-axis for the 2D plot. If necessary, the variable data will be interpolated from neighboring mesh planes.

Examples

```
cplot_xy variable=potential z=50.
```

22.181 cplot_xyz

parameter	data type	values [defaults]
variable	char	(see list)
data_file	char	
wave_option	char	[right],total,left
xy_from	realx2	
xy_to	realx2	
xrange	realx2	
yrange	realx2	
z_min	real	[-1.e9]
z_max	real	[1.e9]
mode_index	intg	[1]
grid_sizes	intgx2	[20, 20]
level	intg	[10]
trap_index	intg	[1]

cplot_xyz is a post-processor statement used to plot structural data on a 2D plane. Unlike **plot_2d**, this command can only generate contour plots.

Other related commands exist and should be used depending on the 2D/3D nature of the original simulation results. The following rules apply:

- 2D simulations: use **plot_2d**
- 3D cylindrical simulations with one mesh plane: use **plot_2d**
- xy plane from a 3D simulation: use **cplot_xy** for contour plots of scalar variables, **splot_xy** for 3D surface plots of scalar variables or **vplot_xy** for vector variables.
- xyz plane from a 3D simulation: use **cplot_xyz** for contour plots of scalar variables, **splot_xyz** for 3D surface plots of scalar variables or **vplot_xyz** for vector variables.

Parameters

This statement is similar to **cplot_xy** and varies only in the way it defines the plotting plane. As such, most of the parameters are also similar to those in **plot_2d**.

- **xy_from** and **xy_to** define the (x,y) corners of the plotting plane.

- **z_min** and **z_max** define the z coordinates of the corners of the plotting plane.

Examples

```
cplot_xyz variable=wave_intensity xy_from=(0.5, 0.0) xy_to=(0.5 3.0) &&
  grid_sizes=(60, 20)
```

22.182 csuprem_mask

parameter	data type	values [defaults]
tag	char	[void]

csuprem_mask is used in the .layer file pre-processing in conjunction with the **export_layers_to_suprem** command. It modifies the CSUPREM input files that are generated by that command and adds a comment line at the appropriate place. An optional used-defined label can also be added as part of the comment line to facilitate the identification of layers.

Parameters

tag is a user-defined label that can be added to the comment line in CSUPREM

22.183 current_conc

parameter	data type	values [defaults]
data_file	char	[void]
use_macro	char	[no]
fit_outfile	char	[void]
auto_pn_ratio	char	[no]
conc_log_scale	char	[no]
conc_range	realx2	[(1.e23 1.e24)] (m^{-3})
pn_ratio	real	[1.]
av_index	real	[3.3]
auger_n	real	[2.e-42] (m^6/s)
auger_p	real	[2.e-42] (m^6/s)
life_n	real	[1.e-6] (sec.)
life_p	real	[1.e-6] (sec.)
data_point	intg	[30]
well_num	intg	[1]
zseg_num	intg	[1]

The statement **current_conc** is a gain preview statement which allows the user to find the relation between carrier concentration and the recombination current in the quantum well active region. The recombination rate is expressed as

$$R = R_{spon} + R_{auger} + n/tau_n + p/tau_p$$

where the spontaneous emission spectrum computed from the quantum well model is integrated over all emission wavelengths and the Auger recombination is given by **auger_n** and **auger_p**, defined below. The last two terms are due to traps, surface recombination, thermal leakage, etc.

Parameters

- **data_file** is the file to which the graphic data is written in ASCII format.
- **use_macro** directs the program to use the material parameters such as Auger coefficients from the material macros instead from parameters in this statement. If this parameter is set to *yes*, parameters of **auger_n**, **auger_p**, **life_n** and **life_p** in this statement will not be used for the model fitting.
- **fit_outfile** if set to a valid file name, is used to direct the model fitting results to a file in the working directory.

- **auto_pn_ratio** is used to indicate if automatic setting of hole/electron density ratio is used in PICS3D simulation. If positive, averaged hole/electron density from the drift-diffusion solver is used. This parameter, if positive, will override **pn_ratio**.
- **conc_range** is the electron concentration range in the well.
- **pn_ratio** is the ratio of hole over electron concentrations. Note that this ratio can be set to an arbitrary number in the gain preview. In the main solver, however, this ratio is determined by the simulator automatically, according to the local Fermi levels.
- **conc_log_scale** would vary the concentration in log scale.
- **av_index** is the estimated average refractive index.
- **auger_n** is the Auger coefficient C_n .
- **auger_p** is the Auger coefficient C_p .
- **life_n** is the minority carrier life time for electrons.
- **life_p** is the minority carrier life time for holes .
- **data_point** is the number of data points to be used in the current versus concentration plot.
- **well_num** is the total number of active quantum wells involved.
- **zseg_num** is the z-segment number.

22.184 cylindrical

parameter	data type	values [defaults]
axis	char	x,[y]
cylindrical_origin	real	[0.] (μm)
min_core	real	[0] (μm)

The **cylindrical** statement is used to define a cylindrical coordinate system. It has been implemented so that 3D devices with cylindrical symmetry can be modeled with a single mesh plane.

With this setting, the 2D cut of the device shown in the .layer file corresponds to a radial cut. 3D results are obtained by integrating the 2D results of the simulation

from 0 to 2π. Note that in versions prior to 2012, the integration range was from 0 to π and a scaling factor had to be used during plotting. This scaling is no longer necessary.

Please refer to Sec. 6.2 for details.

Parameters

- **axis** is the cartesian axis which is used as the rotation or z-axis of the cylindrical coordinate system. The other direction is automatically equivalent to r.
- **cylindrical_origin** is used to offset r axis values during the conversion to the cylindrical coordinate system. A value larger than zero turns a rectangular 2D cut into a torus rather than a cylinder.
- **core_min** is a minimum r value used when converting the local mesh area into a volume for the FEM. This value is necessary to avoid division by zero errors since the gradient operator in cylindrical coordinates contains a term in $\frac{1}{r}\frac{\partial}{\partial\theta}$

22.185 dbr_truncate

parameter	data type	values [defaults]
dir	char	[left] right
kappa_l	real	[5.]
sec_num	intg	[1]

dbr_truncate is used to modify the DBR grating in edge type of lasers so that numerical stability may be achieved. When a DBR grating structure is used as a reflector for a laser diode, the propagating light can only penetrate a finite distance before fully reflected. Such a penetration length is usually in the order of several 1/kappa, where kappa is the coupling coefficient. The DBR gratings beyond this penetration length does not contribute to the laser diode operation but may cause numerical stability during Newton iterations. Thus, we artificially truncate the DBR portion beyond the penetration length by setting the kappa there to zero.

Parameters

- **dir**dir is the direction towards which the DBR grating is used as a reflector. For example, dir=left means the DBR is used to reflect light coming from the

left.

- **kappa_l** is a real number of around five. It is used to measure the penetration length beyond which the DBR is to be truncated. The penetration length is given by kappa_l/kappa, where kappa is the coupling coefficient.

- **sec_num** is the index of the section where the DBR truncation is applicable.

Examples

```
dbr_truncate sec_num=2 kappa_l=5
```

22.186 define_alias

parameter	data type	values [defaults]
alias	char	[void]
name	real	[void]

define_alias is used to define an alias which may be used instead of the original variable in all input files. It acts in many ways like the `#define` directive in C/C++.

To associate an alias to a numerical value, use **define_symbol** instead.

Parameters

- **alias** is the alias being used.

- **name** is the name the alias represents.

Examples

```
define_alias alias=Vg name=voltage_2
```

This defines Vg (gate voltage) as being equivalent to voltage_2.

22.187 define_cavity

parameter	data type	values [defaults]
propagation_dir	char	y, [z]
symmetric_x	char	[yes], no
import_xy_wave	char	
eim_core_xrange	realx2	(μm)
eim_core_zrange	realx2	(μm)
eim_core_index	real	[3.5]
eim_clad_xindex	real	[1.5]
eim_clad_zindex	real	[1.5]

The **define_cavity** command is exclusively used in PICS3D and is similar to vcsel_model. It is used to define the optical mode profile of a micro cavity/photonic crystal structure in the cartesian coordinate system. Unlike vcsel_model, **define_cavity** does not try to compute the standing wave overlap between the optical mode and the active region.

Parameters

- **propagation_dir** is the overall direction of the wave vector in the laser cavity.
- **symmetric_x** is a flag indicating the left border of the cavity is a symmetry axis.
- **import_xy_wave** is the name of text file containing a mode profile that should be imported into the simulation. The format should be defined in 3 columns (x y intensity) with x changing the fastest and a line break between each set of y values. A uniform grid is assumed.
- **eim_core_xrange** is the waveguide core x-range for effective index method (EIM) when computing the wave profile. If **propagation_dir**=y, **eim_core_zrange** serves the same purpose and the defines the depth of the waveguide core. If **propagation_dir**=z, it is assumed the y-range of the waveguide core is the same as the x-range.
- **eim_core_index** is the real part of the refractive index in the waveguide core. Similarly, **eim_clad_xindex** and **eim_clad_zindex** respectively define the real part of the refractive index in the cladding for the x and z directions. If **propagation_dir**=z, it is assumed that the cladding along y is the same as along x.

Examples

```
define_cavity propagation_dir=y &&
eim_core_xrange=(0 0.5) eim_core_zrange=(0. 1.035) &&
     eim_core_index=3.5 eim_clad_xindex=1.5 eim_clad_zindex=1.
```

22.188 define_material

parameter	data type	values [defaults]
mater	intg	[1]

This statement is used when a material number is re-used between z-segments. The first z-segment where this material occurrs must use **load_macro** to load the macro parameters. The other z-segments use **define_material** to indicate **load_macro** has already been used.

Parameters

- **mater** is the number of the material being reused.

Examples

```
define_material mater=8
```

22.189 define_symbol

parameter	data type	values [defaults]
symbol	char	[void]
value	real	

define_symbol is used to associate numerical values with symbols. It acts in many ways like the `#define` directive in C/C++.

This may be used to simplify input files or to represent repeated numerical values so that revisions are more convenient.

A few precautions should be taken when using this statement:

- 1) Avoid using it when the same symbol appears than once within a line;
- 2) The line containing the symbol must have enough line space so that the total length does not exceed 80 character after value substitution;
- 3) Please avoid using symbols that are already defined as parameters in crosslight.tab.

  For example, the following statement will cause problems:

  ```
  define_symbol symbol=doping  value=2.e24
  ```

  Since "doping" is commonly used statement, the program will replace all occurrence of "doping" by "2.e24" and the input file will loss its original meaning. To avoid this problem, try

  ```
  define_symbol symbol=john_smith_doping value=2.e24
  ```

To associate a symbol to a variable or a string input value, use **define_alias** instead.

Parameters

- **symbol** is the symbol to be used to represent numerical value in the program.
- **value** is the numerical value to be used.

Examples

```
define_symbol symbol=my_doping value=1.e24
```

22.190 define_vertical_position

parameter	data type	values [defaults]
label	char	[void]
reference_to	char	[void]
use_y_coord	real	(um)
above_previous	real	(um)
below_previous	real	(um)
cut_at_x	real	(um)
from_top	real	(um)
from_bottom	real	(um)
above_reference	real	(um)
below_reference	real	(um)
between_materials	intgx2	

define_vertical_position is used in .sol file to label a vertical position for later reference. This can be done in one of the following ways:

- 1) Use absolute coordinates
- 2) Defined as first occurrence of material interface between two materials when searching from bottom to top
- 3) In reference to a previously defined position, i.e., using relative distance
- 4) Use relative distance with respect to the top or bottom of the x-y mesh.

Parameters

- **label** is a position label at a specific y-coordinate.
- **reference_to** is used to define a reference point for approach number 3). It must be a predefined position label.
- **use_y_coord** is the y-coordinate for approach number 1).
- **above_previous** indicates the position is above a label in the preceeding statement for 3).
- **below_previous** indicates the position is below a label in the preceeding statement for 3).
- **cut_at_x** is used to define the x-position when using approach 2).
- **from_top** defines the distance from top of the mesh system in 4).
- **from_bottom** defines the distance from bottom of the mesh system in 4).
- **above_reference** is the distance above the reference point defined by parameter **reference_to**.
- **below_reference** is the distance below the reference point defined by parameter **reference_to**.
- **between_materials** defines the two materials used to identify the interface position in 2).

Examples

```
define_vertical_position label=sio2/si &&
      cut_at_x=0.  between_materials=(2 3)
```

This example labels a y-position between materials 2 and 3 and label it as “sio2/si".

22.191 delta_real_index_caxis

delta_real_index_caxis is a passive macro parameter describing the birefringence in wurtzite materials. This value describes the difference between the extraordinary index and the ordinary index with the latter value being defined in **real_index**.

To model the birefringence, **optical_axis** must also be used to configure the waveguide orientation with respect to the extraordinary axis.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.192 delta_so_bar

delta_so_bar is an active macro parameter defining the spin-orbit coupling (in eV) in a zincblende Hamiltonian. It applies to the barrier region of a quantum well.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.193 delta_so_well

delta_so_well is an active macro parameter defining the spin-orbit coupling (in eV) in a zincblende Hamiltonian. It applies to quantum wells and bulk active regions.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.194 delta1_bar

deltaj_bar,j=1..3 are a set of active macro parameters defining valence band edges (in eV) in the barrier of a wurtzite quantum well. They correspond to the Δ_j values

appearing in the Hamiltonian[62].

- Δ_1 corresponds to the crystal field splitting (Δ_{cr})
- Δ_2 and Δ_3 correspond to the spin-orbit splitting. It is common to set $\Delta_2 = \Delta_3 = \frac{\Delta_{so}}{3}$.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.195 delta2_bar

See **delta1_bar**.

22.196 delta3_bar

See **delta1_bar**.

22.197 delta1_bulk

deltaj_bulk,j=1..3 are a set of passive macro parameters defining valence band edges (in eV) in wurtzite materials. They correspond to the Δ_j values appearing in the Hamiltonian[62].

- Δ_1 corresponds to the crystal field splitting (Δ_{cr})
- Δ_2 and Δ_3 correspond to the spin-orbit splitting. It is common to set $\Delta_2 = \Delta_3 = \frac{\Delta_{so}}{3}$.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.198 delta2_bulk

See **delta1_bulk**.

22.199 delta3_bulk

See **delta1_bulk**.

22.200 delta1_well

deltaj_well,j=1..3 are a set of active macro parameters defining valence band edges (in eV) for a wurtzite quantum well or bulk active region. They correspond to the Δ_j values appearing in the Hamiltonian[62]:

- Δ_1 corresponds to the crystal field splitting (Δ_{cr})
- Δ_2 and Δ_3 correspond to the spin-orbit splitting. It is common to set $\Delta_2 = \Delta_3 = \frac{\Delta_{so}}{3}$.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.201 delta2_well

See **delta1_well**.

22.202 delta3_well

See **delta1_well**.

22.203 d1_bar

di_bar,i=1...6 are a set of parameters used to define barrier properties in wurtzite quantum well active macros. They define the shear deformation potential (D_i) as defined in Ref [62].

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.204 d2_bar

See Sec. 22.203.

22.205 d3_bar

See Sec. 22.203.

22.206 d4_bar

See Sec. 22.203.

22.207 d5_bar

See Sec. 22.203.

22.208 d6_bar

See Sec. 22.203.

22.209 d1_bulk

di_bulk,i=1...6 are a set of parameters used to define barrier properties in wurtzite passive macros. They define the shear deformation potential (D_i) as defined in Ref [62].

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.210 d2_bulk

See Sec. 22.209.

22.211 d3_bulk

See Sec. 22.209.

22.212 d4_bulk

See Sec. 22.209.

22.213 d5_bulk

See Sec. 22.209.

22.214 d6_bulk

See Sec. 22.209.

22.215 d1_well

di_well,i=1...6 are a set of parameters used in the active macros for wurtzite quantum wells and bulk active regions. They define the shear deformation potential (D_i) as defined in Ref [62].

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.216 d2_well

See Sec. 22.215.

22.217 d3_well

See Sec. 22.215.

22.218 d4_well

See Sec. 22.215.

22.219 d5_well

See Sec. 22.215.

22.220 d6_well

See Sec. 22.215.

22.221 diagonal_split

parameter	data type	values [defaults]
split_dir	char	[north-west], north-east
right_mater_from	char	[east], west, north, south, original
right_doping_from	char	[east], west, north, south, original

The **diagonal_split** command is used in the .layer file to split a rectangle into two: a line drawn between two corners splits the rectangle into two pieces which can have different material or doping properties from the original rectangle.

In this command, cardinal directions from a map are used instead of the usual up/down/left/right convention.

Parameters

- **split_dir** gives the direction of the diagonal line splitting the rectangle.
- **right_mater_from** is used to initialize the material number of the new triangle created from the split. The triangle left of the split line is left unchanged and uses the material number of the original rectangle. The triangle to the right of the split line may borrow the material number from one of the neighbors of the original rectangle or keep its original number.
- **right_doping_from** is the same as **right_mater_from** but deals with the doping value used in the new triangle to the right of the split line.

22.222 dielectric_constant

The material statement **dielectric_constant** defines the static dielectric constant which appears in the Poisson's equation. It does not affect the wave equation which is written for optical frequencies so this value is unrelated to the value of **real_index**.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.223 differential_gain

parameter	data type	values [defaults]
data_file	char	
include_data	char	
conc_log_scale	char	[no], yes
wavel_range	realx2	(μm)
conc_range	realx2	[1.e23 1.e24] (m^{-3})
pn_ratio	real	[1]
av_index	real	[3.3]
data_point	intg	

differential_gain plots the differential material gain (at the peak of the gain curve) versus carrier density in the active region. This statement can be used in the gain preview module.

Parameters

- **data_file** specifies a file name used to save a copy of the plot data.
- **include_data** includes data files from other gain calculations on the plot for comparison purposes.
- **wavel_range** is the wavelength range used to search for the gain peak.
- **conc_range** is the electron concentration range in the well used for the plot.
- **pn_ratio** is the ratio of hole over electron concentrations.

  Note that this ratio is set to an arbitrary number in the gain preview in order to generate a 1-dimensional plot or a family of spectral curves which depend on a single parameter. In the main solver, this ratio is normally unused and

the gain calculations automatically make use of the local Fermi levels for both electrons and holes.

- **av_index** is the estimated average refractive index.
- **data_point** is the number of data points in the curve.
- **conc_log_scale** determines if the carrier density points are spaced linearly or on a logarithmic scale.

Examples

```
differential_gain wavel_range=(1.0 1.4) &&
   conc_range=(5.e23 5.e24)  pn_ratio=1 data_point=20
```

22.224 direct_eigen

parameter	data type	values [defaults]
select_modes	char	[no] yes
max_index	char	[yes] no
max_element	char	[no] yes
sigma_factor	real	[2.]
select_index	real	
arnoldi_base	intg	[10]

The statement **direct_eigen** is used to activate the restarted Arnoldi algorithm for the optical mode solver.

This statement can be used in conjunction with **multimode** if multiple lateral modes are present.

Application Notes

direct_eigen is not recommended for use in structures without lateral variation. If the mode shape has no variation along the x direction, then the eigensolver should expect to find one degenerate mode for each vertical $x = x_i$ mesh line. In theory, picking any one of these degenerate modes is sufficient as they have the same shape.

In practice however, lack of numerical precision can result in nearly-degenerate modes which are identical in the y direction but differ along x. Picking a single mode out

of these nearly-degenerate modes can therefore result in an artificial lateral variation in the mode intensity. Considering ALL of the degenerate modes for the simulation works around this issue as the sum of all modes respects the expected uniform lateral behavior; however, this approach slows down the simulation.

direct_eigen is also not recommended for broad-area devices. The situation here is similar to the above case but even inclusion of all degenerate modes is not sufficient to capture the physics of interest. For broad-area devices, the lateral standing wave pattern is often of interest but the mesh density in the x direction is seldom dense enough to capture this effect.

In these two cases, using the **optical_field** statement instead of **direct_eigen** is recommended. Note that these two commands should not be used together.

Parameters

- **select_modes** , if positive, is used to select eigen modes based on an effective index value supplied through parameter **select_index**. The eigen modes found will have effective indices below **select_index**. If negative, the search of eigen value will start from effective index estimated from the highest local material index within the solution region.

- **max_index** turns on automatic mode search based on a maximum material index of the device structure.

- **max_element** , if positive, instructs the simulator to find eigen modes based on maximum value of matrix elements.

- **sigma_factor** is used in conjunction with **max_element** to obtain an estimate of the maximum eigen value. This factor is multiplied with the maximum matrix element to provide a starting point for eigen value search.

- **arnoldi_base** is used to control the number of Arnoldi eigen solutions we intend to work with. The total number of eigen vectors is this number plus the number of modes we specify in the **multimode**.

Examples

```
direct_eigen sigma_factor=2.
```

22.225 disconnect_zmesh

parameter	data type	values [defaults]
plane	intg	[-9999]
zseg_num	intg	[-9999]

This statement serves the same purpose as the **z_connect=no** option in the **3d_solution_method** statement: it forces all connecting elements between certain z planes to zero so that no current can flow between them. This statement offers a bit more control as it allows disconnection of individual planes and segments instead of turning off all connections at once.

Parameters

- **plane** is a plane number. The connection will be cut between plane and plane+1.
- **zseg_num** is a segment number. The connection will be cut between the last plane of zseg_num and the first plane of zseg_num+1, thereby disconnecting the two segments.

Examples

```
disconnect_zmesh zseg_num=2
```

22.226 do_raytrace_3d

parameter	data type	values [defaults]
3d_emission_model	char	[plane]
external_package	char	[void]
working_direction	char	[-y]
device_kind	char	[led]
incident_light_side	char	[+y]
surface_model	char	[void]
surface_model_side	char	[+y]
reject_edges	char	[no]
extract_to_file	char	[void]

extract_data	char	[wavelength_absorption]
append_data	char	[no]
dome_bottom_mirror	char	[yes]
precision	real	[0.01](percentage)
te_portion	real	[0.5]
tm_portion	real	[0.5]
plate_thickness	real	[100.] (um)
plate_width	real	[1000.] (um)
dome_cyl_height	real	[500.] (um)
dome_cyl_radius	real	[400.] (um)
dome_sph_radius	real	[400.] (um)
dome_base_thickness	real	[10.] (um)
scale_refl	real	[1.]
scale_tran	real	[1.]
thickness1	real	[0.023]
thickness2	real	[0.034]
package_refr_index	realx2	[1.7,1.e-7]
effective_exit_index	realx2	[1.0,0.0]
refr_index1	realx2	[1.3,0.0]
refr_index2	realx2	[1.7,0.0]
datafile_range	intgx2	[1 1000]
initial_rays	intg	[5000]
max_secondary	intg	[100000]
3d_angle_store	intg	[50]
emit_points_number	intg	[10]
max_primary	intg	[5000]
n_layer	intg	[1]
surface_power_grid	intg2	[100,100]
TE_TM_mode	intg	[0]

do_raytrace_3d is used to set parameters for and to activate the raytracing module for 3D simulation of light emitting diodes (LED).

It can also be used for 2D simulations. In this case, the x-y modeling plane is extruded in the z direction with a thickness of 1 μm to create a 3D volume compatible with the 3D raytracing program. However, emitted rays are restricted to the original 2D plane.

Parameters

- **3d_emission_model** is the emission source model of an LED active region. It can take one of the following values:
  - *one_point*: the spontaneous emission power is averaged to find the center of emission. All the rays emanate from this one point.
  - *xy_plane*: the spontaneous emission power is averaged into a few "hot spots" of emission. Each of these spots is allowed to emit rays which carry the total integrated power of that spot. This is the recommended model when the quantum well plane normal is along the z-axis.
  - *xz_plane*: same as *xy_plane* except for the plane of averaging for the "hot spots". This is the recommended model when the z-axis is in the plane of the quantum well.
  - *mesh_points* and *active_mesh*: every mesh point assigned to an active region is allowed to emit rays. This model is very time-consuming but is the most accurate. The number of angular points should be reduced when using this model to get a reasonable computation time.
  - *box_emitting*: every polygon of the .geo file becomes a 3D box for the raytracing program. The spontaneous emission source power from each box is averaged into a hot "spot" of emission and used to emit rays.
  - *random_points*: Rays are emitted from a randomly-selected points in the device.
- **external_package** indicates the existence of an external package that encapsulate the LED. The value of this variable can be chosen as: *void, medium, plate* or *dome*. A value of *void* means the LED is not encapsulated.
- **working_direction** defines the working direction of all external packages. It can be chosen as: *-x, +x, -y, +y, -z* or *+z*. Figure 22.3 shows how this parameter is defined.
- **device_kind** determines what kind of device (led or detector) is being modeled.
- **incident_light_side** determines which side of a detector is illuminated.
- **surface_model** determines which boundary conditions are applied to certain surfaces; this can be used to represent optical coatings. Note that in certain structures, the coating must be modeled inside the device with the **set_3dray_internal_interface** statement.
- **surface_model_side** determines which side of the device is affected by surface_model.

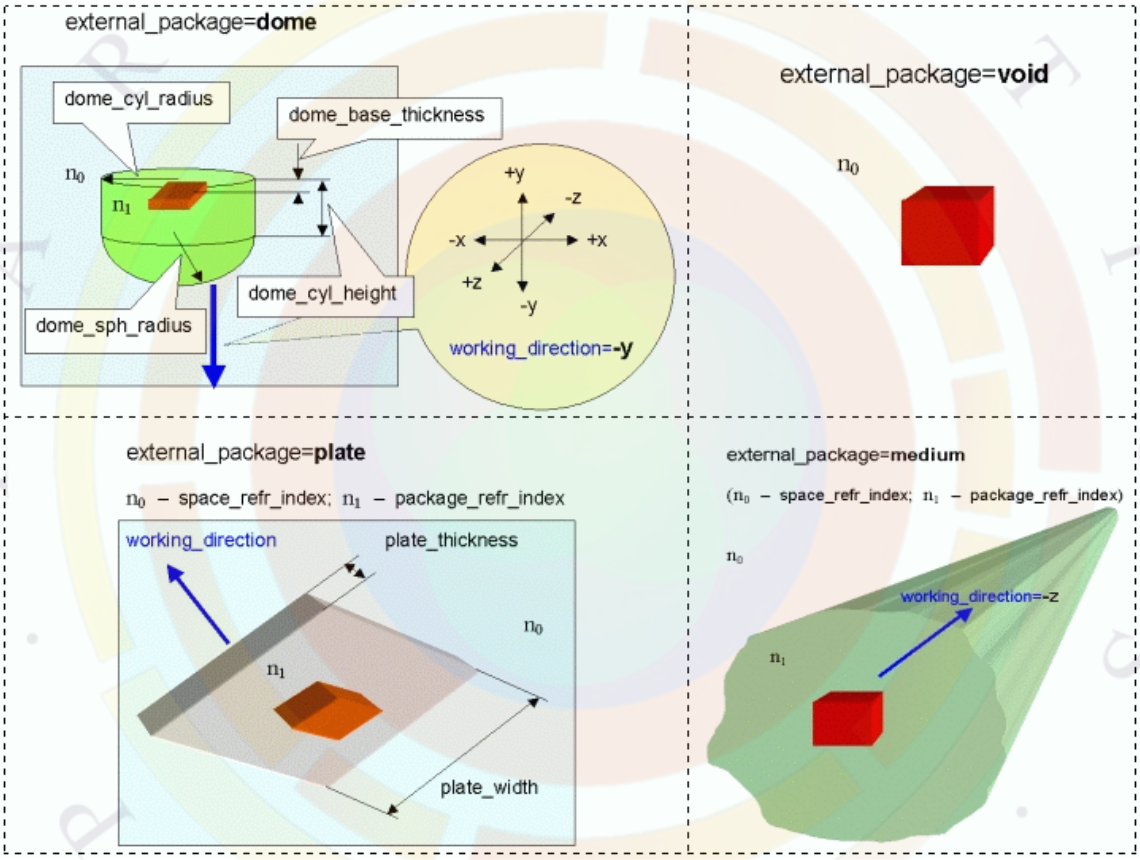


Figure 22.3: An explanation of the packages parameters

- **reject_edges** determines how the mesh points are used internally by the ray tracing program to average material properties. Choosing yes will cause the program to ignore mesh points too close to box edges which can lead to more a accurate refractive index near large discontinuities. However, it will cause an error if there are not enough mesh points.

- **extract_to_file** specifies which file the ray tracing code will export data to; this will be disabled when set to void. The extracted data will be appended to an existing file or will overwrite it, depending on the value of **append_data**.

- **append_data** would instruct the program to append data to an existing data file instead of creating a new one.

- **dome_bottom_mirror** would set a perfect mirror on the bottom of a dome external package.

- **extract_data** specifies which ray tracing result will be extracted to the file specified by **extract_to_file**.

- **precision** is the ray-tracing precision of the ray-tracing calculation. The ray-tracing procedure is terminated if the relative power of a ray decays to this value.

- **te_portion, tm_portion** are the fractions of TE and TM polarizations of propagating light. For non-polarized light **te_portion** and **tm_portion** are equal.

- **plate_thickness, plate_width, dome_cyl_height, dome_cyl_radius,dome_sph_radi dome_base_thickness** are geometrical dimensional parameters of the packages. Figure 22.3 illustrates the meaning of all these parameters. They are in units of μm. A given package is built around (or besides) the device. If one or more sizes of a package are set less than device sizes, the program corrects the sizes automatically to their possible minimum values.

- **scale_refl, scale_tran** are scaling factors to the Fresnel coefficients which are applied when **surface_model**=*scale_fresnel_coef*.

- **thickness1, thickness2, refr_index1,refr_index2, n_layer** are the parameters of the *multi_layer* option of **surface_model**.

- **package_refr_index** defines the complex optical refractive index for all packages.

- **effective_exit_index** is the effective index of the coating when **surface_model** is set to exit_index.

- **datafile_range** the range of data sets exported from APSYS/Optowizard used to do the raytracing simulation. Note that this may be different than the data sets used for normal plotting purposes.
- **initial_rays** is the number of initial rays launched from one emission source point. Equal angular distribution is forced upon these initial rays.
- **max_secondary** is an integer value parameter, that defines maximum secondary rays generated during multiple reflections and refractions. The raytracing procedure will terminate for a certain initial ray if the number of secondary rays exceeds this number. A ray is regarded as secondary if it is generated by transmission from its initial material box to one of its neighbors.
- **3d_angle_store** is the resolution which is used to store the angular power distribution. It is equal to the number of angular divisions in theta angle and is equal to half of the number of divisions in phi angle.
- **emit_points_number** is a total given number of emitting points ("hot spots") for the plane emission models.
- **max_primary** is the maximum number of primary rays allowed. If the primary rays exceeds this number, the program will terminate the tracing of the primary rays. A ray is regarded as primary if it is directly produced from one of the emission points and not from transmission at a material box interface.
- **surface_power_grid** is the size of the grid used to record the location and power of rays exiting the device. Six such grids are located around the device to form a box. The resulting surface emission map can be plotted with 3drayplot_surfpower
- **TE_TM_mode** defines the behavior of the TE/TM emission model for LEDs. If equal to 0, the mix of TE/TM emission from APSYS is used for the raytracing. For 1, the TE emission spectrum is used and for 2, the TM emission is used.

Examples

```
rt3d_contact_reflector transp_contact=yes &&
contact1_compindex=(2.
      0.5) contact1_thick=0.05 && contact2_compindex=(2. 0.5)
      contact2_thick=0.05

$ No dome do_raytrace_3d precision=0.01 datafile_range=(2,6)
initial_rays=8000
```

```
$ With dome external package $do_raytrace_3d precision=0.01
datafile_range=(2,6) initial_rays=8000  && $  external_package=dome
working_direction=+y && $  dome_cyl_height=110
dome_base_thickness=10 && $  dome_cyl_radius=500
dome_sph_radius=500 && $  package_refr_index=(1.7,1.0e-6)
```

22.227 dopant_ionization_model

parameter	data type	values [defaults]
mott_trans_correction	char	[yes],no
mater_label	char	
scale_factor	real	[1.]
max_activation_energy	real	[0.1] (eV)
mater	intg	[1]

This statement controls the heavy doping effects model for a given material. By default, the Mott transition model is always active for shallow dopants so this command does not need to be issued. See Sec. 5.1.5 for details.

Parameters

- **mott_trans_correction** turns on or off the Mott transition model.
- **scale_factor** is a scaling factor for the transition.
- **max_activation_energy** defines the maximum ionization energy where the Mott transition is allowed. Deep dopants with ionization energies beyond this value will not be affected by the Mott transition.
- **mater** identifies the material affected by this statement. If a label has previously been defined for this material, **mater_label** may be used instead.

Examples

```
$ To specifically turn off the Mott transition for a material
dopant_ionization_model mott_trans_correction=no mater=1

$ To specifically turn on the Mott transition for a material
```

```
$ with deep dopants
dopant_ionization_model max_activation_energy=1e99 mater=2
```

22.228 doping

parameter	data type	values [defaults]
impurity	char	[shal_dopant]
charge_type	char	[donor],acceptor
shape	char	[gaussian],polygon
pf_model	char	[no],yes
x_datafile	char	[void]
y_datafile	char	[void]
direction	char	[positive],negative
log_scale	char	[no] yes
xy_datafile	char	[void]
xy_linedata	char	[yes]
traplevel_model	char	[void], gaussian, expo_tail, uniform
traplevel_tail_side	char	[conduction]
datafile_unit	char	[1/m^3],1/cm^3
sheet_density	char	[no] yes
sheet_location	char	[void],bottom,top
polygon_file	char	[void]
all_segment	char	[no]
polygon_z_datafile	char	[void]
polygon_z_datafile_unit	char	[1/m^3],1/cm^3
polygon_z_datafile_log	char	[yes]
decay_type	char	[gaussian],erfc
use_position_label	char	[no], yes
x1_label	char	[void]
x2_label	char	[void]
y1_label	char	[void]
y2_label	char	[void]
max_conc	real	1.e10 (m^{-3})
level	real	[0.01] (eV)
x_prof	realx4	x1,x2,dx1,dx2 (μm)
y_prof	realx4	y1,y2,dy1,dy2 (μm)
edge1_prof	realx3	x2,y2,dx2
edge2_prof	realx3	x2,y2,dx2
edge3_prof	realx3	x3,y3,dx3
edge4_prof	realx3	x4,y4,dx4

pf_cont_shift	real	[0.] (eV)
arc_height1	real	[0.](μm)
arc_height2	real	[0.](μm)
arc_height3	real	[0.](μm)
arc_height4	real	[0.](μm)
origin	real	[0.] (μm)
grading_from	real	(fraction)
grading_angle	real	[90.] (degrees)
xy_data_factor	real	[1.]
xy_x_transform	realx2	[1. 0.]
xy_y_transform	realx2	[1. 0.]
traplevel_stddev	real	[0.1](eV)
traplevel_tail	real	[0.05](eV)
label_tail	real	[0.01] (μm)
polygon_file_stddev	real	[0.01] (μm)
z_stddev	real	[0.] (μm)
z_stddev2	real	(μm)
pnt_num	intg	[4]
newdoping_index	intg	
xy_skipline	intg	[1]
xy_data_column	intg	[3]

The statement **doping** is a frequently used statement which describes the doping profile of impurities or impurity concentration distribution. This statement is often generated automatically when processing the .layer file and should only be used by advanced users.

Multiple doping statements covering overlapping areas can be used to define co-doping or create advanced shapes. Experimental or doping profiles simulated in an another program can also be defined using this statement.

For an even more accurate model, the user is encouraged to try our CSUPREM software which can model the entire growth process of semiconductor devices including etching, diffusion and implantation.

Parameters

- The parameter **impurity** defines the type of impurity being modeled:

- *shal_dopant* or *shal_dopanti,i=1..9* for shallow dopants; multiple species may be defined
- *trap_i,i=1..9* for deep level traps of type i
- *fix_charge* for fixed space charges

When defining traps, the user should also define the statements **trap_ncap_i** and **trap_pcap_i** in the .sol file to specify the capture cross sections of the trap.

- **charge_type** is the charge state of the impurity. For shallow dopants and traps, it must be either *donor* or *acceptor*. For fixed space charges, it must be *positive* or *negative*.
- **shape** controls the shape of the doping region created by this statement:
  - For *gaussian* or *rectangle*, the profile is uniform within a rectangle (see Fig. 22.4) and falls off from the sides using Gaussian tails. The corner points and standard deviations for the Gaussian tails are given by using **x_prof** and **y_prof**.
    The 4 values in **x_prof** correspond to (x1,x2,dx1,dx2) (in μm). **y_prof** similarly defines (y1,y2,dy1,dy2). Note that if x1=x2 and y1=y2, only the tails remain and the shape collapses into a standard Gaussian.
    Alternatively, **use_position_label** allows the use of the position labels in **x1_label**, **x2_label**, **y1_label** and **y2_label** to define the corner points. In such a case, the tail value is set using **label_tail**.
  - *polygon* extends the above behavior with uniform doping inside a more generic polygon with corner points defined through **edgei_prof,i=1..4**. These commands each define 3 values(x,y,dx); (x,y) are the corner coordinates for point i and dx is the standard deviation of the Gaussian tail extending perpendicularly from the polygon edge connecting points i and i+1.
    The number of corner points actually used to define the polygon is controlled with **pnt_num** (see Fig. 22.5).
  - *use_polygon_file* further extends the above behavior with a uniform doping with a polygon whose corners are defined inside a separate text file (**polygon_file**). As before, each edge of the polygon has a Gaussian tail extending perpendicular to it; the standard deviation of this tail is defined in **polygon_file_stddev**.

  Note that when importing doping profiles (e.g. SIMS) from a data file, the **shape** parameter should be left to its default values.
- **decay_type** can be used to replace the standard Gaussian tails mentioned throughout this section with the complimentary error function (erfc).

- **pf_model** turns on/off the Poole-Frenkel model for field-induced thermionic emission of partially ionized dopants. Such a model is important when the ionization energy is high or when the operating temperature is low.
- **x_datafile** or **y_datafile** allows the user to import numerical doping data for the x or y direction. The file data must be stored in 2-column format:

```
relative_distance_in_microns  doping_density
(no blank lines anywhere)
```

  The doping density stored inside the file can be in linear or log10 scale depending on the choice of **log_scale**. The units of the data can also be set by **datafile_unit** ($1/m^3$ or $1/cm^3$).

  The exact position of the doping when importing a profile is further controlled by two other parameters:

  - **direction** is used to flip the sign of the coordinates for imported data files. If negative, the actual doping profile is in a direction opposite to the relative-distance given in the data file.
  - **origin** is the origin used to calculate the absolute position for the data points of the imported doping profile.

  The actual coordinates used in device simulation are therefore:

```
(origin) + (sign-of-direction)*(relative-distance)
```

- **sheet_density** instructs the program that the present command is used to describe a sheet density profile instead of the bulk default. The **max_conc** would be in unit of $1/m^2$ to define a sheet charge over the x-y plane.
- **sheet_location** is used in conjunction with **sheet_density**=*yes* for a 3D structure. It defines the location of the sheet charge either on the top or bottom of the present z-segment.
- **max_conc** defines the maximum concentration of the doping profile.
- **level** defines the relative energy level of the impurity. For shallow donors and deep level traps, the level is calculated from the conduction band. For shallow acceptors the level is measured from the valence band. For gaussian type of trap model, this parameter also refers to the maximum trap level position.

  Note that this parameter is used to describe the energy level of the shallow or deep impurities over the whole device. If the properties of the impurity in each different material need to be defined, the user must override this value with one of the following statements:

  - **shal_dnr_level**
  - **shal_dnr_level_i**
  - **shal_acpt_level**
  - **shal_acpt_level_i**
  - **trap_level_i**

- **pf_cont_shift** is the ionization energy level shift (due to the Poole-Frenkel effect) at the contact. Once it is specified by this parameter, the ionization energy is fixed during the simulation while the shift in the bulk material is varied as a function of the local electric field.

- **arc_heighti,i=1..4** are parameters are used to bend the edge of the doping profile given in **edgei_prof**. The values of **arc_height** can take positive or negative numbers. The sign of the height is understood as follows:

  Let the counter-clockwise direction of the vector defined by two adjacent points of the polygon be vector e and the vector pointing from the starting point of e to a point on the arc be vector a. Then a positive height means that the cross product a cross e must point to positive z direction. On the other hand, if the height is negative, the cross product yields a vector in the negative z direction.

  In other words, a positive height makes the edge curve outwards while a negative height makes the edge curve inwards. These parameters can be used together with the **edge_curve** statement.

- **grading_from**, if used, is the relative starting value of a linearly graded doping profile; its value is relative to the final doping value of **max_conc**. This linear grading is in the direction specified with **grading_angle** where the angle definition follows the usual convention (0 degrees = +x direction).

- **newdoping_index**, if set to positive integer, is used to label a doping profile so that the doping concentration of this profile may be varied during a simulation. The variable key word in the **scan** statement to control this profile is *new_doping*.

- **xy_datafile**, **xy_linedata**, **xy_data_factor**, **xy_x_transform**, **xy_y_transform**, **xy_skipline**, **xy_data_column**.

  All these parameters with the **xy_** prefix are used to import 2D doping profiles in ASCII format:

  - **xy_datafile** is doping data file in a column-wise format:

    ```
    x y doping-concentration
    ```

where x and y are in micons and doping-concentration is in $1/m^3$. To convert to a different unit, the parameter **xy_data_factor** may be used.

- **xy_linedata** indicates whether the doping profile is line-ordered data or randomly distributed. If *yes*, the data is ordered in multiple straight x-lines, from left to right. The x-lines are ordered from bottom to top. Using random data generally yields a less accurate profile after interpolation.
- **xy_data_factor** is a multiplication factor applied to the doping concentration. It may also be used as a way of converting units.
- **xy_x_transform** consists two parameters (say a and b) which can be used to perform a linear transform of the x-coordinate in the doping profile so that the actual coordinate used by the simulator is a*x+b.
- **xy_y_transform** is the same as **xy_x_transform** except it is for y-coordinate.
- **xy_skipline** is the number of header lines in the doping file that should be skipped.
- **xy_data_column** is the data column number where the doping-concentration can be found. In some data files, n- and p-doping are stored in different columns and this parameter can be used to specify which type of doping is to be used from the same file.

The following code is used internally to read the doping profile and may be helpful in understanding the required syntax:

```
!  program skips lines of xy_skipline first. then, the following code
! is used
      do jj=1,npnt
        read(iu,*) (adata(kk),kk=1,jcolumn)
! this include unit change, direction flip, etc., and shift ref. point
! let me assume unit is um
        xdata(jj)=adata(1)*xy_x_transform1+xy_x_transform2
        ydata(jj)=adata(2)*xy_y_transform1+xy_y_transform2
! unit here is 1/m**3
        if(log_flag.eq.0) then
        ddata(jj)=adata(jcolumn)*xy_data_factor
        else
        ddata(jj)=10.d0**(adata(jcolumn))*xy_data_factor
        endif
      enddo
```

- **traplevel_model** may be used to define a continuous distribution of trap states. If this statement is omitted, then a single energy level is used.

- **traplevel_stddev** is the trap level standard deviation if the trap level model is gaussian.
- **traplevel_tail** is the characteristic decay constant (L) of the exponential tail model $e^{-E/L}$. The energy E is measured from either the conduction of valence band depending on the value of **traplevel_tail_side**.
- **traplevel_width** is the energy width for a uniform distribution of trap states.
- **all_segment** instructs the software to apply the 2D doping profile defined by this command to all mesh planes of all z-segments.
- **polygon_z_datafile** is used to extend the 2D doping profile across multiple mesh planes. The file specified in this command must contain 2 columns: the z-position and the maximum 2D doping concentration at that position. Units and log-scaling for this file are controlled by **polygon_z_datafile_unit** and **polygon_z_datafile_log**; these parameters follow the same rules as **datafile_unit** and **log_scale**, respectively.
- **z_stddev** extends the 2D doping profile in the z-direction so that it straddles multiple mesh planes. It defines a Gaussian tail which is applied in both directions. For an asymmetric Gaussian profile in z, **z_stddev2** may also be defined.

Examples

```
doping impurity=shal_dopant charge_type=donor max_conc=1.e24  &&
  x_prof=(0.0e0, 10.0e0,  0.01e0, 0.01e0 )  &&
  y_prof= 0.e0, 1.4e0,  0.02e0, 0.02e0

doping impurity=trap_1 charge_type=acceptor max_conc=1.e22  &&
  x_prof=(0.0e0, 10.0e0,  0.01e0, 0.01e0 )  &&
  y_prof=(0.e0, 10.e0,  0.02e0, 0.02e0 )

doping impurity=shal_dopant charge_type=donor max_conc=1.e24  &&
   shape=polygon &&
   edge1_prof= (0. 1.1 0.01)   &&
   edge2_prof= (0.429 1.1 0.01)   &&
   edge3_prof= (0.146 1.5 0.01)   &&
   edge4_prof= (0. 1.5 0.01)


doping impurity=shal_dopant charge_type=acceptor max_conc=1.5e23  &&
  level=0.115 pf_model=yes pf_cont_shift=0.115 &&
```

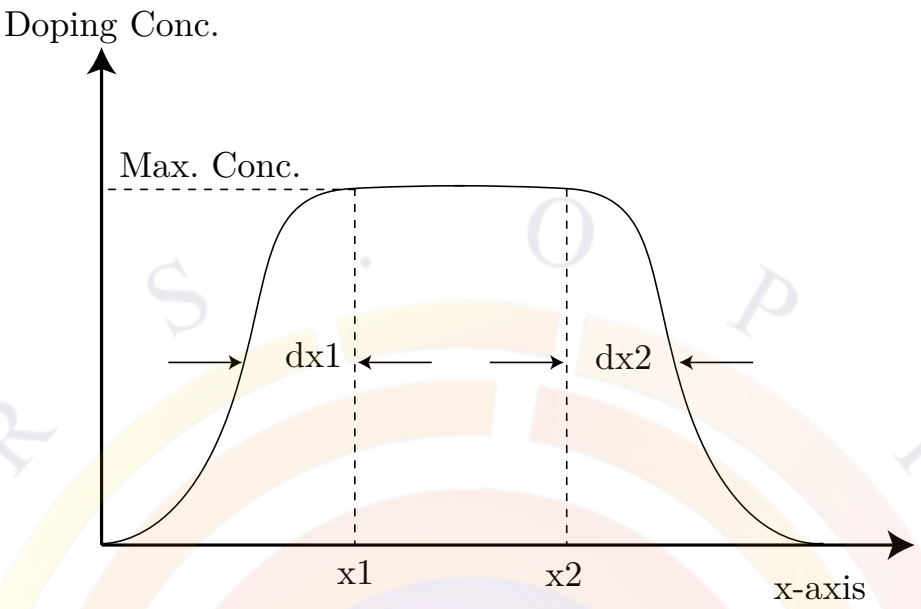


Figure 22.4: Schematics for the Gaussian/Rectangle doping profile in the x-direction.

```
x_prof=(0.0e0, 10.0e0,  0.01e0, 0.01e0 )  &&
y_prof=(0.e0, 1.e0,  0.02e0, 0.02e0 )
```

22.229 double_mesh

parameter	data type	values [defaults]
mesh_inf	char	
mesh_outf	char	
polygon	char	
order	char	[yes],no
direction	intg	1,2
range_1	realx2	(um)
range_2	realx2	(um)

The mesh generation statement **double_mesh** provides the basic mechanism to refine a rough mesh manually. It can be used to double the existing mesh density in a given region in the device. This statement is rarely needed because mesh can be automatically refined using the **refine_mesh** statement.

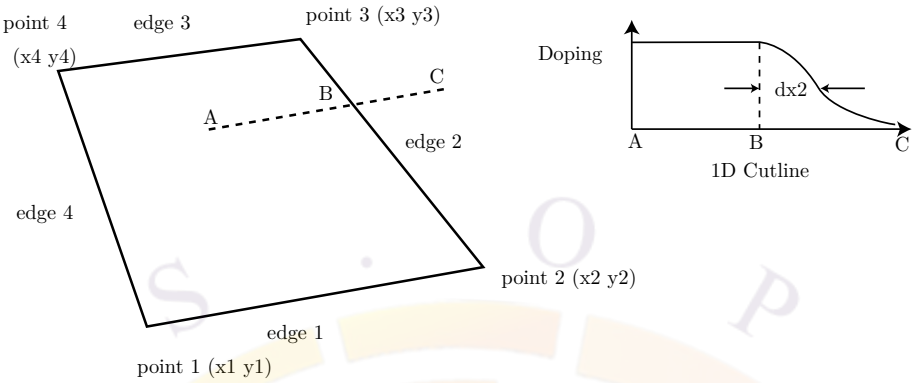


Figure 22.5: Schematics for the Polygon doping profile. Note the labeling of points and corresponding edges.

Parameters

- **mesh_inf** is the mesh input file.
- **mesh_outf** is the mesh output file.
- **polygon** is the name of the polygon affected by this statement.
- **order** specifies the format of the mesh output. If the mesh output is to be used by the solver, *yes* should be used, i.e., ordering of the mesh is required to interface with the solver. If the mesh is to be further manipulated after this statement, *no* should be used.
- **direction** is the direction of the mesh lines to be doubled. Direction 1 corresponds to lines parallel to the first edge of the polygon; direction 2 refers to lines parallel to the second edge. See Fig. 22.6 for details.
- **range_1** defines the range along direction 1 in which the mesh doubling will occur. This distance is measured on the first edge of the polygon.
- **range_2** is the same as **range_1** excepts that it works along direction 2.

Examples

```
double_mesh polygon=p1 direction=1 &&
  range_1 =(0.1 0.5) range_2=(0.6 0.9) &&
  mesh_inf=cas1.msh1  mesh_outf=cas1.msh2  order=yes
```

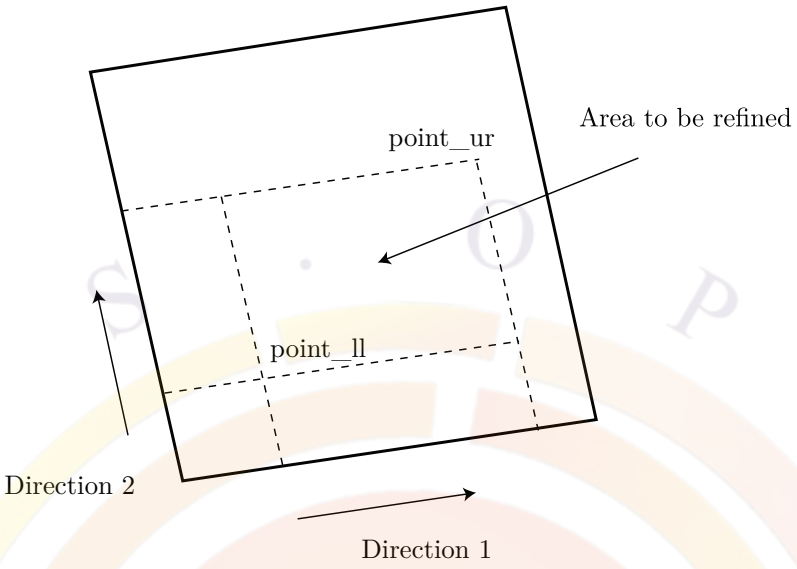


Figure 22.6: Schematics for the region to be manually refined.

22.230 dox_efield0_pf_elec

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox_efield0_pf_elec** is a constant $E0_{pf}$ (in V/m) used to define the Poole-Frenkel-like E-field dependent electron mobility: $\mu_n = \mu_0 exp[\sqrt{(E/E0_{pf})}]$ This parameter is used to describe a doped organic (thus the prefix dox) material. For more details, please see the following reference:

B. Ryhstaller and S.A. Carter, S. Barth, H. Riel, and W. Riess, "Transient and steady-state behavior of space charges in multilayer organic light-emitting diodes," J. Appl. Phys., 15 April, 2001, Vol. 89, No. 8, pp. 4575-4586.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

```
Example(s)
dox_efield0_pf_elec value=1.959e8 mater=1
```

22.231 dox_efield0_pf_hole

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox_efield0_pf_hole** is a constant $E0_{pf}$ (in V/m) used to define the Poole-Frenkel-like E-field dependent hole mobility: $\mu_p = \mu_0 exp[\sqrt{(E/E0_{pf})}]$ This parameter is used to describe a doped organic (thus the prefix dox) material. For more details, please see the following reference:

B. Ryhstaller and S.A. Carter, S. Barth, H. Riel, and W. Riess, "Transient and steady-state behavior of space charges in multilayer organic light-emitting diodes," J. Appl. Phys., 15 April, 2001, Vol. 89, No. 8, pp. 4575-4586.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

```
Example(s)
dox_efield0_pf_hole value=1.959e8 mater=1
```

22.232 dox_el_weight

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox_el_weight** is used to define the electroluminescent spectrum weight for the dopant material. This is useful when there is dopant in the system so that one has to define the relative contribution from host and the dopant.

For EL spectrum model when there is dopant, two approaches are used, depending on whether exciton diffusion equations are solved. If exciton transport is solved by **organic_exciton_diff**, dopant and host spectra at each mesh point are simply mixed by local exciton density, divided by their respective life times, and weighted by this factor. If exciton transport is not considered, their respective bimolecular recombination rates are used along with this factor to determine the relative contributions.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
dox_el_weight value=0.01 mater=1
```

22.233 dox_exciton_eg

This statement is the same as **ox_exciton_eg** except it is for the dopant material.

22.234 dox_extern_spectrum

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox_extern_spectrum** is used to define the electroluminescent (EL) spectrum of the dopant material imported from an external source. This may be based on experimental data or on a different type of model. If table format is used, the program expects a 2-column uniformly spaced data with first column being the wavelength in micron meters. The unit of the EL is arbitrary and the program will normalize it.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
dox_extern_spectrum variation=table
table(wavelength)
  0.488524E+00  0.309522E-01
  0.499601E+00  0.421672E-01
  ...
end_table
```

22.235 dox__gaussian__divj

This statement is the same as **ox__exciton__eg** except it is for the dopant material.

22.236 dox__gaussian__sdj

This statement is the same as **dox__gaussian__sdj** except it is for the dopant material.

22.237 dox__hopping__energy

This statement is the same as **ox__hopping__energy** except it is for the dopant material.

22.238 dox__peak__abs

This statement is the same as **dox__peak__abs** except it is for the dopant material.

22.239 dox__vib__quanta

This statement is the same as **ox__vib__quanta** except it is for the dopant material.

22.240 dox__xp__coupling

This statement is the same as **dox__xp__coupling** except it is for the dopant material.

22.241 dox2__el__weight

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox2_el_weight** is used to define the electroluminescent spectrum weight for the 2nd dopant in multiply doped materials. This is useful to define the relative contribution from host and the dopant.

For EL spectrum model when there is dopant, two approaches are used, depending on whether exciton diffusion equations are solved. If exciton transport is solved by **organic_exciton_diff**, dopant and host spectra at each mesh point are simply mixed by local exciton density, divided by their respective life times, and weighted by this factor. If exciton transport is not considered, their respective bimolecular recombination rates are used along with this factor to determine the relative contributions.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
dox2_el_weight value=0.01 mater=1
```

22.242 dox2_extern_spectrum

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox2_extern_spectrum** is used to define the electroluminescent (EL) spectrum of the 2nd dopant material imported from an external source. This may be based on experimental data or on a different type of model. If table format is used, the program expects a 2-column uniformly spaced data with first column being the wavelength in micron meters. The unit of the EL is arbitrary and the program will normalize it.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
dox2_extern_spectrum variation=table
table(wavelength)
  0.488524E+00  0.309522E-01
```

```
  0.499601E+00  0.421672E-01
  ...
end_table
```

22.243 dox3_el_weight

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox3_el_weight** is used to define the electroluminescent spectrum weight for the 3rd dopant in multiply doped materials. This is useful to define the relative contribution from the host and the dopant.

For EL spectrum model when there is dopant, two approaches are used, depending on whether exciton diffusion equations are solved. If exciton transport is solved by **organic_exciton_diff**, dopant and host spectra at each mesh point are simply mixed by local exciton density, divided by their respective life times, and weighted by this factor. If exciton transport is not considered, their respective bimolecular recombination rates are used along with this factor to determine the relative contributions.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
dox3_el_weight value=0.01 mater=1
```

22.244 dox3_extern_spectrum

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox3_extern_spectrum** is used to define the electroluminescent (EL) spectrum of the 3rd dopant material imported from an external source.

This may be based on experimental data or on a different type of model. If table format is used, the program expects a 2-column uniformly spaced data with first column being the wavelength in micron meters. The unit of the EL is arbitrary and the program will normalize it.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
dox3_extern_spectrum variation=table
table(wavelength)
  0.488524E+00  0.309522E-01
  0.499601E+00  0.421672E-01
  ...
end_table
```

22.245 dox4_el_weight

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox4_el_weight** is used to define the electroluminescent spectrum weight for the 4th dopant in multiply doped materials. This is useful to define the relative contribution from host and the dopant.

For EL spectrum model when there is dopant, two approaches are used, depending on whether exciton diffusion equations are solved. If exciton transport is solved by **organic_exciton_diff**, dopant and host spectra at each mesh point are simply mixed by local exciton density, divided by their respective life times, and weighted by this factor. If exciton transport is not considered, their respective bimolecular recombination rates are used along with this factor to determine the relative contributions.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
dox4_el_weight value=0.01 mater=1
```

22.246 dox4_extern_spectrum

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox4_extern_spectrum** is used to define the electroluminescent (EL) spectrum of the 4th dopant material imported from an external source. This may be based on experimental data or on a different type of model. If table format is used, the program expects a 2-column uniformly spaced data with first column being the wavelength in micron meters. The unit of the EL is arbitrary and the program will normalize it.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

```
Example(s)

dox4_extern_spectrum variation=table
table(wavelength)
  0.488524E+00  0.309522E-01
  0.499601E+00  0.421672E-01
  ...
end_table
```

22.247 dox5_el_weight

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox5_el_weight** is used to define the electroluminescent spectrum weight for the 5th dopant in multiply doped materials. This is useful to define the relative contribution from host and the dopant.

For EL spectrum model when there is dopant, two approaches are used, depending on whether exciton diffusion equations are solved. If exciton transport is solved by **organic_exciton_diff**, dopant and host spectra at each mesh point are simply mixed by local exciton density, divided by their respective life times, and weighted by this factor. If exciton transport is not considered, their respective bimolecular recombination rates are used along with this factor to determine the relative contributions.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
dox5_el_weight value=0.01 mater=1
```

22.248 dox5_extern_spectrum

parameter	data type	values [defaults]
(see) material_par		

The material statement **dox5_extern_spectrum** is used to define the electroluminescent (EL) spectrum of the 5th dopant material imported from an external source. This may be based on experimental data or on a different type of model. If table format is used, the program expects a 2-column uniformly spaced data with first column being the wavelength in micron meters. The unit of the EL is arbitrary and the program will normalize it.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
dox5_extern_spectrum variation=table
table(wavelength)
  0.488524E+00  0.309522E-01
  0.499601E+00  0.421672E-01
  ...
end_table
```

22.249 edge_curve

parameter	data type	values [defaults]
edge		
shape		[arc]
height		[μm]

The mesh generation statement **edge_curve** bends an edge of a polygon.

- **edge** contains the two points of the edge to be bent.
- **shape** is the shape into which the edge is to be bent.
- **height** is the height of the arc. The sign of the height is understood as follows: Let the counter-clockwise direction of the vector defined by two adjacent points of the polygon be vector e and the vector pointing from the starting point of e to a point on the arc be vector a. Then a positive height means that the cross product a cross e must point to positive z direction. On the other hand, if the height is negative, the cross product yields a vector in the negative z direction. In another word, positive height makes the edge curve outwards while a negative height makes the edge curve inwards.

Example(s)

```
edge_curve edge=(a b) shape=arc height=-0.3
```

22.250 eeim_optic

parameter	data type	values [defaults]
gain_correction	char	[yes],no
sort_imag	char	[yes],no
pure_index_loss	char	[yes],no
export_eeimmsg	char	[no],yes
modal_base	intg	[3]
xsearch_real	intg	[2000]
xsearch_imag	intg	[40]
muller_maxit	intg	[80]
wi (i=1,2,3...19)	real	(um)
xsearch_range	real	[5.]
muller_vartol	real	[1.e-8]
muller_functol	real	[1.e-5]

The **eeim_optic** statement is used to invoke the enhanced effective index method (EEIM) for the optical mode computation. It can be used to treat device structure with leaky (radiative) modes.

The rectangle solution area is defined by **point_ll** and **point_ur** in the **init_wave** statement. It assumes that the modes in the y-direction is confined. The boundary on the right is such that it can support traveling wave to the right (which may be decaying as in the case of confined modes).

The boundary on the left can be confined as even or odd, or as radiative. The left boundary may be modified using the first number (we shall refer to as lx1) in **boundary_type** of the **init_wave** statement. If lx1 is 1, the left boundary is to set the wave to zero (odd). If lx1 is 2, the left boundary is to set the wave derivative to zero (even). If lx1 is 3, the left boundary is to have traveling wave going out of the device (to the left).

To study device with radiation loss to the substrate, we suggest that the device be rotated by 90 degrees so that the substrate is towards the +x direction. The rotated device may be done within the .geo file or by using the GeoEditor.

- **gain_correction** may be used to enable the optical gain in the rate equation to be corrected for radiation loss. With this setting to "no", The simulator will calculate the modal gain by averaging the local material gain with weight of the wave intensity.
- **sort_imag** is used to decide if EEIM modes are sorted according to the imaginary parts of the optical dielectric constant. If positive, the modes will be

sorted according to modal gain with the highest first. If negative, the real part will be sorted with the highest real index first.

- **pure_index_loss**. There are two ways to estimate the modal radiative loss. The first is the pure index method. This is by setting all imaginary material indices to zero throughout the device. The imaginary part of the eigen value of the EEIM is the modal loss. The other method is to estimate the bias dependent radiative loss by comparing the EEIM solution with averaged material gain. The former is a convenient and definite way of characterizing the device. The latter is more realistic but less stable numerically. Please note that if **pure_index_loss=yes**, only the modal radiative loss for pure index structure is printed as run-time message. If **pure_index_loss=no**, both the radiative loss for pure index structure and the bias-dependent radiative loss at different bias will be printed
- **export_eeimmsg** is used to enable the export of EEIM column index data in .sol.msg files.
- **modal_base** is the number of y-modes within each column. In principle, there are infinite number of y-modes is the mode searching range is large enough. In practice, several of these will be sufficient.
- **xsearch_real** is the number of search points for the x-modes in the real part of $\bar{\varepsilon}$, the eigen solution of the x-modes.
- **xsearch_imag** is the number of search points for the x-modes in the imaginary part of $\bar{\varepsilon}$, the eigen solution of the x-modes.
- **muller_maxit** is the maximum number of the Muller solver that is used to find the complex root as the eigen solution of the lateral modes.
- **wi (i=1,2,3,...,19)** is the ith column width.
- **xsearch_range** is the search range for the real part of $\bar{\varepsilon}$, the eigen solution of the x-modes.
- **muller_vartol** is the variable relative tolerance at convergence of the Muller solver.
- **muller_functol** is the function relative tolerance at convergence of the Muller solver.

```
eeim_optic modal_base=3  &&
  w1=2 w2=0.75 w3=1.25 &&
  xsearch_real=2000 xsearch_imag=5
```

The above statement uses three y-modes, sets the with for each column. The number search points are also specified.

22.251 effective_medium

parameter	data type	values [defaults]
materi(i=1..9)	intg	
xrange	realx2	(um)
yrange	realx2	(um)
zrange	realx2	(um)

effective_medium defines an effective "average" material out of multiple bulk materials. This is useful in regions like DBR layers where multiple thin layers are needed for optical modeling but the electrical modeling can be approximated by a single bulk materials. The software will compute the affinity, bandgap, DOS, etc... based on a weighed average of the materials present in the specified region.

For a similar model used in QWs, please see **effective_miniband_model**.

Parameters

- **materi(i=1..9)** are the material numbers that are considered when constructing the effective medium.
- **xrange**, **yrange** and **zrange** are the extent of the effective medium.

Examples

```
effective_medium mater1=1 mater2=2 yrange=(2.5 2.75)
```

22.252 effective_miniband_model

parameter	data type	values [defaults]
materi(i=1..9)	intg	
xrange	realx2	(um)
yrange	realx2	(um)
zrange	realx2	(um)
set_cond_level	real	(ev)
set_val_level	real	(ev)

effective_miniband_model defines an effective miniband model for coupled quantum wells/dots. For the specified materials, the solver creates an effective band structure with band edges defined by the miniband levels and density of states from the quantum mechanical modeling.

For bulk layers, please see **effective_medium**.

Parameters

- **materi(i=1..9)** are the material numbers assigned to the miniband.
- **xrange**, **yrange** and **zrange** are the extent of the miniband.
- **set_cond_level** and **set_val_level** can be used to manually override the position of the miniband energy levels. The values are measured from the QW bottom.

Examples

```
effective_miniband_model mater1=1 mater2=2 yrange=(2.5 2.75)
```

22.253 efield0_pf_elec

parameter	data type	values [defaults]
(see) material_par		

The material statement **efield0_pf_elec** is a constant $E0_{pf}$ (in V/m) used to define the Poole-Frenkel-like E-field dependent electron mobility: $\mu_n = \mu_0 exp[\sqrt{(E/E0_{pf})}]$ For more details, please see the following reference:

B. Ryhstaller and S.A. Carter, S. Barth, H. Riel, and W. Riess, "Transient and steady-state behavior of space charges in multilayer organic light-emitting diodes," J. Appl. Phys., 15 April, 2001, Vol. 89, No. 8, pp. 4575-4586.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

```
Example(s)
efield0_pf_elec value=1.959e8 mater=1
```

22.254 efield0_pf_hole

parameter	data type	values [defaults]
(see) material_par		

The material statement **efield0_pf_hole** is a constant $E0_{pf}$ (in V/m) used to define the Poole-Frenkel-like E-field dependent hole mobility: $\mu_p = \mu_0 exp[\sqrt{(E/E0_{pf})}]$ For more details, please see the following reference:

B. Ryhstaller and S.A. Carter, S. Barth, H. Riel, and W. Riess, "Transient and steady-state behavior of space charges in multilayer organic light-emitting diodes," J. Appl. Phys., 15 April, 2001, Vol. 89, No. 8, pp. 4575-4586.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

```
Example(s)
efield0_pf_hole value=1.959e8 mater=1
```

22.255 eg0_bar

eg0_bar is an active layer macro statement defining the unstrained bandgap (in eV) of a quantum barrier. This overrides bandgap values defined in the passive macro.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.256 eg0_bulk

eg0_bulk is a passive macro material statement that defines the unstrained bandgap for a bulk wurtzite material. It is used instead of **band_gap** due to the more complex strain effects in this material system.

For quantum well barriers and active regions, this value will be overridden with the values of **eg0_bar** or **eg0_well** from the active macro.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.257 eg0_well

eg0_well is an active layer macro statement defining the unstrained bandgap (in eV) of a quantum well or bulk active region. This overrides bandgap values defined in the passive macro.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.258 e15_bulk

e15_bulk and related commands are used in a passive macro (wurtzite only) to define piezoelectric tensor components[120]. This is used in conjunction with the **spont_charge** macro parameter and the **polarization_charge_model** command to define the total polarization vector and the interface charges that occur in GaN and ZnO-based materials.

See also **piezo_d11** for related commands used in SAWAVE.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.259 e31_bulk

See **e15_bulk**

22.260 e33_bulk

See **e15_bulk**

22.261 elec_carr_loss

elec_carr_loss is a passive macro material statement defining the dependence of the optical loss coefficient on the electron density for a given material. It introduces a loss term equal to α_n = elec_carr_loss $\times (n - n_0)$ so that **elec_carr_loss** has units of m^2.

Note that this term is only used for passive regions as active regions have their own mechanism for carrier-dependent losses; see **passive_carr_loss** for more information.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

Examples

```
elec_carr_loss value=2.e-21 mater=1
```

22.262 elec_dos_energy

parameter	data type	values [defaults]
data_file	char	
energy_range	realx2	
data_point	intg	[50]
subband_valley	intg	[1]

elec_dos_energy plots the electron density of states (DOS) versus energy. This statement is only used in the gain preview module.

Parameters

- **data_file** is a user-specified text file containing a copy of the plot data.
- **energy_range** is the range of energy in eV.
- **data_point** is number of data points in the plot.
- **subband_valley** is the index labeling the subband valley.

Parameters

```
elec_dos_energy  energy_range=(1 1.7)
```

22.263 electron_mass

parameter	data type	values [defaults]
(see) material_par		

The material statement **electron_mass** defines the electron effective mass relative to the free electron mass.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.264 electron_mobility

The statement **electron_mobility** is usually used as part of a passive material macro and provides a way for the user to define the low field mobility for electrons. The user may input a specific value or use a custom function to model the doping or trap dependence.

An alternative way to define the low field mobility is to use the following statements to implement Eq. 5.42 exactly:

- **max_electron_mob** defines μ_{2n}
- **min_electron_mob** defines μ_{1n}
- **electron_ref_dens** defines N_{rn}
- **alpha_n** defines α_n

Note that using **electron_mobility** explicitly overrides this alternative method.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.265 electron_ref_dens

See **electron_mobility**

22.266 electron_sat_vel

The material statement **electron_sat_vel** is used to define the saturation electron velocity (in m/s). It is used in Eq. 5.37 to define the field-dependent mobility function.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.267 eliminate_mesh

parameter	data type	values [defaults]
line_dir	char	x,[y]
xrange	realx2	(μm)
yrange	realx2	(μm)
column_num	intg	1
ntimes	intg	1

eliminate_mesh is used after the **layer_mater** statement in the .layer file to change the mesh inside the "box" defined by the intersecting layer and column. It removes mesh lines so that in this particular region, the mesh is sparser than what is specified by the **layer** and **column** statements.

See also the related low-level commands **double_mesh** and **half_mesh**.

Parameters

- **line_dir** removes either horizontal or vertical mesh lines.
- **xrange** and **yrange**, when used, limit the effect of this command to a specific region. Coordinates are relative to the lower left corner of the "box" affected by this command.
- **column_num** is the column number; it matches the declaration in **layer_mater** and is used for the same reason.
- **ntimes** is the number of times is the mesh lines are eliminated; each elimination removes half of the mesh lines so that the total reduction is 2^{-n}.

22.268 embedded_material

parameter	data type	values [defaults]
mater_label	char	
position_from	real	(μm)
position_to	real	(μm)
grade_from	real	
grade_to	real	
mater	intg	
grade_var	intg	[0]
grade_num	intg	[5]

embedded_material is used to define a material occupying the same space as another material; the second material is "embedded" into the first. This command works in conjunction with **get_active_layer** to load material parameters related to the embedded material.

Parameters

- **position_from** and **position_to** are the absolute starting and ending coordinates of the embedded region.
- **grade_from** and **grade_to** are the starting and ending compositions.
- **mater** is the material number inside which the new material is embedded. If a label alias has previously been defined, **mater_label** may be used instead.
- **grade_var** is the material macro variable number in which there is a composition grading (if > 0).
- **grade_num** is the number of composition evaluation points if there is a composition grading.

Examples

```
embedded_material position_from=0.5 position_to=0.501 mater=6
```

22.269 end_bpmplot

See **begin_bpmplot**.

22.270 end_cavity

See **begin_cavity**.

22.271 end_complex

See **begin_complex**.

22.272 end_loop

end_loop terminates a loop initialized by **start_loop**.

22.273 end_qwire_complex

end_qwire_complex is the closing tag for **start_qwire_complex** (in the .layer file) or **begin_qwire_complex** (in the .sol or .mater file).

22.274 end_same_complex

See **start_same_complex**.

22.275 end_zdir_complex

end_zdir_complex is the the closing bracket for **begin_zdir_complex**. This statement has no parameters.

See also **zdir_cx** for more information.

22.276 end_zmater

See **begin_zmater**.

22.277 endloopif

Terminates a conditional execution block started by **loopif**.

22.278 equilibrium

parameter	data type	values [defaults]
print_output	char	[yes]
all_newdoping	char	[no]
outfile_label	char	[void]
scan_label	char	[void]
impose_bg_field	char	[no]
bandgap_reduction	real	[0.]
mobility_order	real	[0.]
surf_charge_expo	real	[0.]
newdoping_order	real	[0.]
impact_fermi_factor	real	[0.]
scale_impact_cr_field	real	[1.]
scale_polar_charge	real	[1.]
bg_field_x	real	[0.] (V/m)
bg_field_y	real	[0.] (V/m)
bg_field_z	real	[0.] (V/m)
bg_ref_potential	real	[0.] (V)
scale_negative_stim	real	[1.]
nstep	intg	[1]
index_newdoping	intg	
num_newdoping	intg	[1]
n_index_newdoping	intgxn	

---

equilibrium solves the thermal equilibrium state of the device. All electrode voltages and currents are set to zero, as are all other forms of bias (e.g. optical pumping). The quasi-Fermi levels are flat ($E_{fn} = E_{fp} = 0$) and the net current for electrons and holes has a trivial solution ($\vec{J} = 0$). Only the Poisson equation needs to be solved to find the position of the band edges.

The results of **equilibrium** are often used as the initial guess for a subsequent **scan** statement.

Parameters

- **print_output** is used to decide whether or not to print detailed xy-data sets.
- **all_newdoping** decides if all doping profiles labeled as "newdoping" will be scaled by this statement.
- **outfile_label** labels the output file (.out) so that future **scan** statements can refer to it.
- **scan_label** labels the output file so it can be easily referred to by plotting tools such as CrosslightView.
- **impose_bg_field** can be used to impose a uniform background field to the Poisson equation. Note that this clearly violates the assumption of zero net current during thermal equilibrium so it should never be used on a real device.

  It can however be useful when modeling a stand-alone GaN-based quantum dot where the external piezoelectric field must be considered for self-consistent quantum calculations. The results of such a microscopic-scale calculation are then imported into the full macroscopic device simulation.

  The background field is defined through the parameters **bg_field_x**, **bg_field_y** and **bg_field_z**.
- **bandgap_reduction** is used to set the overall initial bandgap reduction factor at equilibrium. The bandgap is multiplied by (1-bandgap_reduction).
- **mobility_order** is used to increase or reduce the overall mobility at equilibrium. A factor of $10^{m.o.}$ is multiplied to the mobility where m.o. stands for **mobility_order**.
- **surf_charge_expo** is used to increase or reduce the equilibrium surface charge density as specified by the **interface** statement. A factor of $10^{s.c.e}$ is multiplied to the surface charge density where s.c.e stands for **surf_charge_expo**.
- **scale_polar_charge** is a linear factor used to scale the fixed charges created as a result of the **polarization_charge** and **polarization_charge_model** commands.
- **newdoping_order** is used to specify an initial doping concentration reduction factor given by $10^{newdoping\_order}$ which is applied to doping profiles with a new-doping label. The labels where the reduction is applied must be specified using **all_newdoping**, **index_newdoping** or **n_index_newdoping**. If **newdoping_order** is zero (default), then no reduction of doping concentration is initially set.

  The purpose of using this factor is to reduce the doping concentration at equilibrium so that numerical convergence is more easily achieved.

- **nstep** is used only when equilibrium is difficult to converge. In such case, several steps may be used.
- **index_newdoping** refers to a single doping profile that will be varied during the simulation. This index must corresponds to the new-doping labels attached to a doping statement.
- **num_newdoping** is the number of indices to be specified by **n_index_newdoping**.
- **n_index_newdoping** is the same as **index_newdoping** except it is used to specify multiple new-doping labels that will be simultaneously varied during the simulation.
- **impact_fermi_factor** is a coefficient used to scale between a completely field-driven impact ionization model (0.0) and a model driven by the gradient of the Fermi level (1.0). The latter model must be activated in the **impact_model** statement for it to be available here.
- **scale_impact_cr_field** can be used to artificially scale the critical field for the impact ionization model.
- **scale_negative_stim** can be used to artificially scale the stimulated recombination rate in areas where it is below zero.

22.279 evaluate_parameter

parameter	data type	values [defaults]
para_name	char	
printout	char	[parametervalueout.txt]
var_symboli (i=1..9)	char	
var_values_from_file	char	
mater_label	char	
vari (i=1..9)	real	
mater	intg	[1]

evaluate_parameter can be used to manually evaluate a macro function; it is mainly used with the MacPlot GUI to automate the input.

Parameters

- **para_name** is the macro function or parameter name to be evaluated.

- **printout** is a user-specified text file where the results are saved.
- **var_symboli** and **vari** define the input parameters to the macro. The syntax rules are the same as in **layer_mater**.
- **var_values_from_file** is a user-specified file that can be used to input a series of values that will be used to evaluate the macro parameter. Each line of the file corresponds to one call to the function.
- **mater** is the material number of the function being evaluated. It should match that of the previous **load_macro** statement.
- **mater_label** may be used instead of **mater** if a label has previously been defined as an alias.

22.280 exclude_from_electrical

parameter	data type	values [defaults]
mater_label	char	
x_on_left	real	-1.e89 (um)
x_on_right	real	1.e89 (um)
y_below	real	-1.e89 (um)
y_above	real	1.e89 (um)
mater	intg	

exclude_from_electrical is used to exclude a specific region of the mesh from the electrical solver.

Parameters

- **x_on_left** excludes all mesh points to the left of this value.
- **x_on_right** excludes all mesh points to the right of this value.
- **y_below** excludes all mesh points below this value.
- **y_above** excludes all mesh points above this value.
- **mater** excludes all mesh points with the specified material number. If a label has been defined for this material, **mater_label** may be used instead.

22.281 export_3dgeo

export_3dgeo is reversed for use in CSUPREM and PROCOM and is outside the scope of this manual.

22.282 export_fdtd_inputdata

export_fdtd_inputdata forces the device simulator to stop just after generating the input data file (fdtd.aux) for the FDTD solver. It can be used in cases where the electrical calculations are not needed. It should not be needed in OptoWizard which does not include the electrical solver at all.

This statement has no parameters.

22.283 export_gain_data

parameter	data type	values [defaults]
file	char	[void]
pn_ratio_range	realx2	[1. 1.]
temper_range	realx2	[300. 300.] (K)
pn_ratio_points	intg	[1]
temper_points	intg	[1]

export_gain_data exports the gain/index/PL data into an ASCII file for later use. It is a preview simulation statement (used in a .gain file).

- **file** the data file into which the gain data is to be exported. If not specified, a default file name gain_datafile.txt is to be used.
- **pn_ratio_range** is the range of hole/electron ratio changes for the data.
- **temper_range** is the temperature range in the data.
- **pn_ratio_points** is the number of data points for hole/electron ratio changes.
- **temper_points** the number of temperature points.

Example(s)

```
export_gain_data pn_ratio_range=[0.5 1.5] pn_ratio_points=5 &&
  temper_range=[250 350] temper_points=5
```

22.284 export_kp_data

parameter	data type	values [defaults]
file	char	[void]

export_kp_data exports the k.p subbands and dipole moment data an ASCII file for later use. It is a preview simulation statement (used in a .gain file).

- **file** the data file into which the k.p data is to be exported. If not specified, a default file name kp_datafile.txt is to be used.

Example(s)

```
export_kp_data
```

which will save the k.p data in kp_datafile.txt.

22.285 export_kp_para

parameter	data type	values [defaults]
file	char	[void]

export_kp_data exports the k.p parameters, such as Luttinger numbers as an ASCII file for later use. It is a preview simulation statement (used in a .gain file).

- **file** the data file into which the k.p data is to be exported. If not specified, a default file name kp_parafile.txt is to be used.

Example(s)

```
export_kp_para
```

which will save the k.p parameters in kp_parafile.txt.

22.286 export_layers_to_suprem

parameter	data type	values [defaults]
column_num	intg	[1]
plane_num	intg	[2]
z_size	real	[1.0] (μm)

export_layers_to_suprem is a .layer pre-processing statement which emulates the functions of the MaskEditor GUI. When used, the layer.exe output is modified so that 3D CSUPREM input files are generated.

As in the normal .layer output, the plane of .layer file corresponds to the xy plane of the simulation. Multiple copies of this mesh plane are stacked in the z direction to form the 3D device. This convention is the same as that used in the **z_structure** statement.

Note that this behavior is different from that of **layers_for_semicrafter** which uses the SemiCrafer convention for the orientation of the mesh planes.

Parameters

- **column_num** is the column number which is used in the CUSPREM output. The current version of the software supports the export of only one column to CSUPREM.
- **plane_num** is the number of mesh planes used in the z-direction.
- z_size is the length of the device in the z-direction, in microns.

22.287 export_raytrace

parameter	data type	values [defaults]
every_bias	char	[no]
ray3d_convert	char	[no]

export_raytrace is used to export control parameters for the ray-tracing model in the post-processing stage.

- **every_bias** indicates whether ray-tracing data are exported at every bias point. If "no", the export action will occur only at structure data printing point corresponding to .out and .std files.
- **ray3d_convert** should be used to convert 2D electrical simulations to 3D ray tracing structures: this is done by setting a fake z-length of 1 μm. Emitted rays will also be restricted to the 2D plane since the z-direction has no real meaning: this must be taken into account during plotting the emission pattern.

  As of the 2009 version of APSYS, 2D ray tracing is no longer supported and the 3D ray tracing program must be used. This options is mandatory to do ray tracing in 2D APSYS simulations.

22.288 export_to_iccap_mdm_file

parameter	data type	values [defaults]
iccap_file_name	char	[iccap.mdm]
input_vari(i=1..5)	char	voltage_1,current_1
input_namei(i=1..5)	char	v1,vs,vg
output_vari(i=1..5)	char	voltage_1,current_1
output_namei(i=1..5)	char	v1,vs,vg
point_num	intg	[51]

export_to_iccap_mdm_file is a post-processing statement used to export I-V data to the .mdm file format for the parameter extraction program IC-CAP. This command acts like the **plot_scan** command except that only voltage and current are supported.

Parameters

- **iccap_file_name** is the output file name.
- **input_vari(i=1..5)** is the input variable, similar to scan_var in **plot_scan**. A label may be assigned to this variable with **input_namei(i=1..5)**.

Output variables are likewise defined using **output_vari(i=1..5)** and **output_namei(i=1..5)**.

- **point_num** is the number of data points used to sample the simulated IV curve and export the .mdm file.

Examples

The following scan commands are used to generate (.sol) and plot (.plt) the IdVg curve:

```
scan var = voltage_3 value_to = 15 max_step = 2
scan var=voltage_2 value_to=3  max_step= 0.5
scan var=voltage_2 value_to=-6 max_step= 0.5

...

get_data  main_input=hemt.sol sol_inf=hemt.out &&
  xy_data=(4 4) scan_data=(4 4)
export_to_iccap_mdm_file input_var1=voltage_2 input_var2=voltage_1 &&
      input_var3=voltage_3 input_name1=vg input_name2=vs &&
      input_name3=vd output_var1=current_3 output_var2=current_2 &&
      output_name1=id output_name2=ig
```

The above command is similar to the usual Id-Vg plot:

```
plot_scan scan_var = voltage_2 variable = current_3  &&
    user_xlabel = vg  user_ylabel = id
```

The output iccap.mdm looks like the following:

```
BEGIN_HEADER
ICCAP_INPUTS
vg V 2 GROUND SMU1 0.1 LIN -1 2.99000 -6.00000 51 -0.179800
vs V 1 GROUND GND 0.1 CON 0.00000
vd V 3 GROUND GND 0.1 CON 15.0000
ICCAP_OUTPUTS
id I 3 GROUND SMU1 B
ig I 2 GROUND SMU1 B
END_HEADER
BEGIN_DB
ICCAP_VAR vs   0.00000
ICCAP_VAR vd   15.0000
```

```
# vg id ig
  2.99000       711.714       0.00000
  2.81020       703.636       0.00000
  2.63040       695.340       0.00000
  2.45060       686.417       0.00000
  2.27080       677.368       0.00000
  2.09100       668.234       0.00000
  1.91120       657.506       0.00000
  1.73140       646.778       0.00000
  1.55160       635.566       0.00000
  1.37180       623.040       0.00000
  1.19200       610.514       0.00000
  1.01220       597.816       0.00000
 0.832400       584.936       0.00000
 0.652600       572.057       0.00000
 0.472800       558.599       0.00000
 0.293000       544.903       0.00000
 0.113200       531.206       0.00000
-0.666000E-01   513.744       0.00000
-0.246400       495.984       0.00000
-0.426200       477.230       0.00000
-0.606000       452.650       0.00000
-0.785800       428.070       0.00000
-0.965600       402.283       0.00000
 -1.14540       374.394       0.00000
 -1.32520       346.504       0.00000
 -1.50500       318.346       0.00000
 -1.68480       289.997       0.00000
 -1.86460       261.648       0.00000
 -2.04440       232.944       0.00000
 -2.22420       204.152       0.00000
 -2.40400       175.349       0.00000
 -2.58380       146.058       0.00000
 -2.76360       116.767       0.00000
 -2.94340       88.8666       0.00000
 -3.12320       65.3389       0.00000
 -3.30300       41.8113       0.00000
 -3.48280       24.3041       0.00000
 -3.66260       13.8499       0.00000
 -3.84240       3.39561       0.00000
 -4.02220     0.138864E-02    0.00000
 -4.20200     0.463291E-05    0.00000
 -4.38180     0.259545E-05    0.00000
```

```
 -4.56160      0.557988E-06  0.00000
 -4.74140      0.375334E-09  0.00000
 -4.92120      0.674823E-09  0.00000
 -5.10100      0.974312E-09  0.00000
 -5.28080      0.134528E-08  0.00000
 -5.46060      0.143912E-08  0.00000
 -5.64040      0.798202E-09  0.00000
 -5.82020      0.157282E-09  0.00000
 -6.00000     -0.671439E-12  0.00000
END_DB
```

For detailed explanation of the mdm file format, the user is referred to the IC-CAP manual.

22.289 export_wave

parameter	data type	values [defaults]
backward	char	[no]
data_file	char	[void]
print_point	intg	

export_wave is used to export the wave solution of Beam Propagation Method to a designated output file.

Parameters

- **backward** specifies which wave is being exported (forward or backward).
- **data_file** is the file name where the wave data will be exported
- **print_point** is the print point at which the data to be to exported.

Example(s)

```
export_wave backward=no data_file=tmp.wave print_point=12
```

22.290 ext_funck (k=1..9)

parameter	data type	values [defaults]
(see) material_par		

The material statement **ext_funci** (i=1,2,..,9) is an active layer macro statement used to define an external function to be used by other statements in the same macro.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.291 external_cir

external_cir is outdated. It has been replaced with a full-fledged mixed-mode simulator using the **minispice** command.

22.292 external_heat_source

parameter	data type	values [defaults]
mater_label	char	
power_density	real	[0.0] (W/m^3)
power	real	(W) for 3D, (W/m) for 2D
mater	intg	1

external_heat_source is used to define an artificial heat source inside the device.

Parameters

- **power_density** is the heat source density inside the specified material. Alternatively, **power** may be used to define the heat source using the same units as the overall simulation: see the (W) for 3D, (W/m) for 2D. If used, **power** overrides the **power_density** setting.
- **mater** is the material number where the heat source is defined; alternatively, a pre-existing material label may be set in **mater_label**.

22.293 external_spice_cir

external_spice_cir is outdated. It has been replaced with the **minispice** command.

22.294 external_stress_band_model

parameter	data type	values [defaults]
device_lattice_const	real	[3.189] (A)
device_lattice_c_const	real	[3.189] (A)
device_bulk_mater	intg	[]
ref_elasticity_mater	intg	[]

external_stress_band_model works in conjunction with **set_stress** to define externally applied stress (σ) on a device. This should be understood as the stress applied on the whole device rather than the internal, as-grown, strain (ϵ). A good example of this is GaN-based devices grown on silicon substrates.

external_stress_band_model is used to define the reference lattice parameters and internal strain of the device so the software can correctly calculate the total displacement/strain caused by the external stress.

At the moment, this statement primarily applies to wurtzite materials.

Parameters

- **device_bulk_mater** is used to indicate the buffer layer which controls the initial growth and internal strain in the device. When the external stress is applied, it is assumed that this layer is massive enough that its mechanical properties dominate the overall bending in the device.

- **device_lattice_const** and **device_lattice_c_const** are the reference a and c lattice constants of the device during the growth. These values are only used when the bulk reference material is not provided.

- **ref_elasticity_mater** is the reference material which provides the stiffness coefficients c_{ij}. This value is only used when the bulk reference material is not provided.

Examples

```
$ no for 50:50 EBL offset
wurtzite_offset_model bulk_strain_exist = no
$ material numbers
$ 1 -   n-GaN = ingan(0.001)
$ 2 -   MQW barrier = ingan(0)
$ 3-7   quantum well = ingan(0.24)
$ 8 -   EBL = gaalinn(0.20)
$ 9 -  p-GaN = gaalinn(0)
set_stress xx=10.  yy=10.
external_stress_band_model device_bulk_mater=1
```

22.295 extract_contour

parameter	data type	values [defaults]
variable	char	(see list)
data_file	char	
flip_y	char	yes,[no]
at_value	real	
at_fraction	real	
extend_y_fraction	real	0.
set_yrange	realx2	(μm)
mode_index	intg	1
trap_index	intg	1

extract_contour is a post-processor statement used to plot structural data on a 2D plane. It is similar to **plot_2d** in that it plots contour lines but this command only plots a single line at a specified value or fraction.

Parameters

- **variable** is the variable to be plotted. See App. G for a full list of available variables.
- **data_file** is the name of a text file in which a copy of the plot data will be saved.
- **flip_y** flips the y axis of the display.

- **at_value** is the absolute value of the contour cut line. It is mutually exclusive with **at_fraction**.
- **at_fraction** sets the value of the contour cut line to a fraction of the maximum. It is mutually exclusive with **at_value**.
- **extend_y_fraction** is used to zoom in on the y axis: a scaling factor of $1 + value$ is applied
- **set_yrange** limits the contour plot to the data points found within this range.
- **mode_index** is the index of the lateral mode in a multi-lateral mode simulation. If only the fundamental mode is concerned, the default value of one should be used.
- The parameter **trap_index** has been added for SRH recombination, so that the user can specify which trap center is being plotted.

Examples

```
extract_contour variable=wave_intensity at_fraction=0.367879
```

This plots the contour of the wave intensity where the wave is $1/e$ of its maximum.

22.296 farfield

parameter	data type	values [defaults]
option	char	[contour], segment_x, segment_y, surface
data_file	char	[void]
symmetric	char	[yes],no
add_all_modes	char	[yes],no
type	char	[intensity], real, imaginary
theta_x	real	[10] (degrees)
theta_y	real	[40] (degrees)
2ndmode_phase	real	[0.]
points	intg	[30]
mode_index	intg	[1]
level	intg	[9]
sum_2ndmode	intg	[0]

farfield is a post-processing statement used to compute the far-field distribution of an optical mode. The near-field distribution is obtained by simply plotting the *wave_intensity* variable in statements such as **plot_2d**.

Parameters

- **option** specifies the format of the far-field output:
  - *contour* shows a 2D contour plot
  - *segment_x* shows a 1D plot along the x direction
  - *segment_y* shows a 1D plot along the y direction
  - *surface* shows a 3D surface plot
- **type** instructs the software to plot either the intensity, the real part or the imaginary part of the far-field pattern.
- **data_file** can be used to save the data to a text file.
- **symmetric** specifies whether or not the device being simulated is symmetric. If set to *yes*, it is assumed that only the right half is simulated and the far-field is computed with the missing near-field on the left half. Otherwise, the far-field is computed as is.
- **add_all_modes** instructs the program to sum the intensities of all the far-field modes without considering any phase-interference effects. The sum is weighed by the power in each mode.
- **theta_x** and **theta_y** are the range of the far-field angle along the x and y directions, respectively.
- **2ndmode_phase** is the phase factor of the higher order mode with respect to the main mode when **sum_2ndmode** is used. This factor corresponds to the phase difference between the two modes over a single cavity round-trip and is added as part of the summation for the far-field pattern.
- **points** is the number of points used to evaluate the far-field in each direction.
- **mode_index** is the index of the lateral mode in a multi-lateral mode simulation. If only the fundamental mode is concerned, the default value of 1 should be used.
- **level** is the number of contour lines when **option**=*contour*.

- **sum_2ndmode** is the index of the second mode added to the main mode for the far-field calculations; unlike **add_all_modes** phase effects are used for this sum, using the value of **2ndmode_phase**. This parameter is ignored when using the default value of 0.

Examples

```
farfield option=contour points=30 &&
  theta_x=15 theta_y=70
```

22.297 farfield_couple

parameter	data type	values [defaults]
symmetric	char	[yes],no
theta	real	[2] (degrees)
2ndmode_phase	real	[0.]
scale_lit	real	[1.]
scale_curr	real	[2.]
points	intg	[50]
mode_index	intg	[1]
sum_2ndmode	intg	[0]
electrode	intg	[1]

farfield_couple used to integrate a certain farfield angle and modify the L-I curve to simulate a detector.

- **symmetric** specifies whether or not the device being simulated is symmetric. If the choice is “yes”, it is assumed that only the right half is simulated and the far-field is computed with the missing near field on the left half. If the choice is “no”, then the far field is computed using only the near field as it is.
- **theta** is the farfield angle of power detection.
- **2ndmode_phase** is the phase factor of the 2nd order mode to be added to the farfield of the main mode.
- **scale_lit** is a scale factor to be multiplied to light power.
- **scale_curr** is a scale factor to be multiplied to current.

- **points** is the number of points in each direction for farfield integration.
- **mode_index** is the index of the lateral mode in a multi-lateral mode simulation. If only the fundamental mode is concerned, the default value of one should be used.
- **sum_2ndmode** is the mode number of the higher order mode to be added to the main mode for farfield computation. (see also 2ndmode_phase parameter above).
- **electrode** is the electrode upon which the current of the L-I curve is based.

```
Example(s)
farfield_couple theta=5 sum_2ndmode=1
```

22.298 fdtd_background_mater

parameter	data type	values [defaults]
mater	intg	

The command **fdtd_background_mater** defines a material number that is to be used as a background material for the FDTD model. This material contains the FDTD current source and the FDTD model will always start by modeling light propagation in this isotropic material in order to set normalization constants (e.g. for power).

Usually, the FDTD current source is placed in an air or vacuum region. However, in special cases, the user may want to put the FDTD current source inside of the device which is where this command becomes useful.

Parameters

- **mater** is a material number to be dealt as background material.

Examples

```
fdtd_background_mater mater=1
```

22.299 fdtd_CLFDTD_control

parameter	data type	values [defaults]
operation	char	[append_xml]
append_from	char	
append_to	char	

fdtd_CLFDTD_control may be used to directly issue commands to the Crosslight FDTD solvers using XML input files; the syntax of these files is beyond the scope of this document.

Only append operations between two XML files are currently supported by this command.

Examples

This example appends contents of QE_monitor.xml to fdtdinput_device.xml:

```
fdtd_CLFDTD_control operation=append_xml &&
  append_from=QE_monitor.xml &&
  append_to=fdtdinput_devicespace.xml
```

22.300 fdtd_data_analysis

parameter	data type	values [defaults]
tag	char	
operation	char	relative_energy_intensity, transmittance, reflectance, poynting, QE, OE
operandA_tag	char	
operandB_tag	char	
data_file	char	
scale	real	1.0

The command **fdtd_data_analysis** is used to define data analysis post-processing for FDTD simulations. This command only works for CLFDTD.

Parameters

- **tag** is a user-defined name assigned to the results of this analysis and which may be referenced by other post-processing commands. The assigned **tag** must be unique and not reused by other **fdtd_data_analysis** commands.
- **operandA_tag** and **operandB_tag** define the quantities being analyzed. These user-defined labels are inherited from **fdtd_field_monitor**.
- **operation** is the kind of analysis performed by this command; the analysis operates on **operandA_tag** and **operandB_tag** as outlined in the examples below.
- **data_file** is the name of a data file where the data analysis results are recorded.
- **scale** is an artificial scaling factor that may be used to multiply the analysis results.

Examples

operation=*relative_energy_intensity* calculates the spatial distribution of $n\frac{|\mathbf{operandA\_tag}|^2}{|\mathbf{operandB\_tag}|^2}$ where n is the refractive index:

```
fdtd_data_analysis &&
   operation=relative_energy_intensity &&
   operandA_tag = device_fields &&
   operandB_tag = empty_fields &&
   data_file=rei.txt
```

operation=*transmittance* calculates $\frac{\int \mathbf{operandA\_tag}\cdot\vec{n}d\vec{r}}{\int \mathbf{operandB\_tag}\cdot\vec{n}d\vec{r}}$:

```
fdtd_data_analysis &&
   operation=transmittance &&
   operandA_tag = trans_fields &&
   operandB_tag = injection_fields &&
   data_file=trans.txt
```

operation=*reflectance* calculates $\frac{\int(\mathbf{operandA\_tag}-\mathbf{operandB\_tag})\cdot\vec{n}d\vec{r}}{\int \mathbf{operandB\_tag}\cdot\vec{n}d\vec{r}}$:

```
fdtd_data_analysis &&
   operation=reflectance &&
   operandA_tag = refl_fields &&
   operandB_tag = injection_fields_refl &&
   data_file=refl.txt
```

operation=*poynting* calculates the Poynting vector distribution of **operandA_tag**:

```
fdtd_data_analysis &&
   operation=poynting &&
   operandA_tag = trans_fields &&
   data_file=poynting.txt
```

operation=*QE* calculates the quantum efficiency:

```
fdtd_data_analysis &&
   operation=QE &&
   operandA_tag = QE_monitor_devicespace &&
   operandB_tag = QE_monitor_emptyspace &&
   scale=1e-6 &&
   data_file=QE.txt
```

operation=*OE* calculates the optical efficiency.

```
fdtd_data_analysis &&
   operation=OE &&
   operandA_tag = flux_monitor_group &&
   operandB_tag = injection_monitor &&
   data_file=OE.txt
```

22.301 fdtd_define_region

parameter	data type	values [defaults]
tag	char	
shape	char	line, plane, box
flip_normalj (j=1..6)	char	-x,+x,-y,+y,-z,+z
point_ll	realx3	(um)
point_ur	realx3	(um)
normal	realx3	

fdtd_define_region defines a geometric region in a FDTD simulation domain. This command only works for CLFDTD.

Parameters

- **tag** is a user-defined name for this region which allows it to be referenced by other commands such as **fdtd_field_monitor**. The **tag** must be unique and not reused by other **fdtd_define_region** commands.

- **shape** specifies shape and dimensionality of the region.

- **flip_normalj** is used in the context of box regions; each facet/side of the box has its own surface normal vector. By default, surface normals point outwards from the box center; flipping the normal means that the normal now points towards the center of the box.

  Multiple flips (j=1..6) may be issued in the same command. For example, **flip_normal1**=-*y* will flip the surface normal vector on the -*y* side of the box region as the first operation.

  Issuing this command may be required depending on whether the user is interested the fields that are incoming or exiting from a specific region.

- **point_ll** is used to specify the lower-left corner of the region.

- **point_ur** is used to specify the upper-right corner of the region.

- **normal** is used to specify the normal vector of the region when **shape**=*plane*: it is necessary to compute the flux through a specific region. In the case of **shape**=*box*, the normal vector for each facet is automatically assigned by the simulator as described in **flip_normalj**.

Examples

```
fdtd_define_region &&
    shape=plane &&
    point_ll=(0.0 -1.2 0.0) &&
    point_ur=(1.0 11.5 0.0) &&
    normal=(0.0 0.0 1.0) &&
    tag=planeXY
```

22.302 fdtd_dispersion

parameter	data type	values [defaults]
type	char	[Lorentz],Drude,Drude-Lorentz,Direct
import	char	[no],yes
file	char	void
freq_convention	char	[omega],frequency
input_unit	char	[MEEP],SI
autofit_singlepole	char	[no],yes
mater	intg	1
order	intg	0
order_drude	intg	0
epsinf	real	1.0
omegak(k=1..9)	real	
gammak(k=1..9)	real	
delta_epsk(k=1..9)	real	
omega_drudek(k=1..9)	real	
gamma_drudek(k=1..9)	real	
a_directk(k=1..9)	real	
b_directk(k=1..9)	real	
c_directk(k=1..9)	real	
d_directk(k=1..9)	real	
scale_absorption	real	

The command **fdtd_dispersion** defines dispersion parameters for a particular material. These parameters are used in the FDTD model to account for dispersion and loss. Different models are available depending on the FDTD solver being used (**interface** in **fdtd_model**).

Theory

Since the FDTD model works in the time domain, it is not possible to directly use spectral properties. Instead, spectral properties must be fitted to a set of simple basis functions which can be represented more easily in the time domain. This is analogous to the concept of "digital filters" used in modern digital signal processing algorithms.

The Lorentz dispersion model is supported by both MEEP and Acceleware FDTD (in $\exp(-i\omega t)$ convention):

$$\epsilon(\omega, \mathbf{x}) \;=\; \epsilon_\infty(\mathbf{x}) \;+\; \sum_n \frac{\omega_n^2 \Delta\epsilon_n}{\omega_n^2 - \omega^2 - i\omega\gamma_n}, \tag{22.11}$$

The Drude (in $\exp(+i\omega t)$ convention):

$$\epsilon(\omega, \mathbf{x}) \;=\; \epsilon_\infty(\mathbf{x}) \;-\; \sum_n \frac{\omega_n^2}{\omega(\omega + i\gamma_n)}, \tag{22.12}$$

and Direct (in $\exp(+i\omega t)$ convention):

$$\epsilon(\omega, \mathbf{x}) = \sum_n \frac{A_n}{B_n\omega^2 + C_n i\omega + D_n}, \tag{22.13}$$

dispersion models are only supported by Acceleware FDTD.

Important note on units and fitting of experimental data

Most FDTD solvers use dimensionless and scale-invariant units internally to speed up computations. Because users of Crosslight deal mostly with photonic applications, the scale of the simulation is always set to be 1 μm and the speed of light is set to $c = 1$: this is consistent with the convention used by the MEEP FDTD solver.

Using this convention, the following relations hold in vacuum:

$$\omega \;=\; 2\pi f \tag{22.14}$$

$$f \;=\; \frac{1}{\lambda} \tag{22.15}$$

where λ is equal the wavelength value in μm but is otherwise without units.

However, not all FDTD solvers use the same convention when defining the dispersion relations. The Acceleware solver use the formulas shown above for all available dispersion models. On the other hand, MEEP's *polarizability* class expresses the Lorentz dispersion model using an alternate but equivalent form (in terms of f rather than ω):

$$\epsilon(f, \mathbf{x}) \;=\; \epsilon_\infty(\mathbf{x}) \;+\; \sum_n \frac{f_n^2 \Delta\epsilon_n}{f_n^2 - f^2 - if\frac{\gamma_n}{2\pi}}, \tag{22.16}$$

The parameters **freq_convention** and **input_unit** can be used to convert to the appropriate between frequency conventions and to specify the choice of units in the dispersion relations.

Parameters

- **mater** is a material number in which dispersion parameters are to be assigned.
- **type** specifies type of dispersion function.
- **import** is a switch for the importation of dispersion function parameters. When **import**=*yes*, all the dispersion function parameters will be read from an external file.
- **file** is the name of a file containing dispersion function parameters.
- **freq_convention** tells simulator what kind of frequency was used in the fitting procedure of dispertion function. Augular frequency ω is assumed by default. Since MEEP accepts ω, whereas Acceleware FDTD accepts $frequency$, adequate conversion of frequency type will take place in the Crosslight/FDTD interface according to the combination of FDTD program and **freq_convention**.
- **input_unit** tells simulator what kind of unit is used for the fitting parameters of dielectric function. Since MEEP accepts MEEP unit, whereas Acceleware FDTD accepts SI unit, adequate unit conversion will take place according to the combination of FDTD program and **input_unit**.
- **autofit_singlepole** is a switch for fitting real and imaginary part of complex index at single wavelength automatically. By activating this switch, user do not have to fit index spectrum to analytical functions like Lorentzian. This switch only works for single wavelength simulation.
- **order** is the number of dispersion terms (n) used in the Lorentz and Direct models.
- **order_drude** is the number of dispersion terms (n) in the Drude model.
- **epsinf** is the instantaneous dielectric function: the real part of permittivity at infinite frequency. It corresponds to ϵ_∞ appearing in the equations above.
- **omega#** is the resonant frequency of the absorption peak in the Lorentz model. For the Acceleware solver, it corresponds to ω_n in Eqn. (22.11). For the MEEP solver, it corresponds to $f_n = \frac{\omega_n}{2\pi}$ in Eqn. (22.16).
- **gamma#** sets the width of the absorption peak in the Lorentz model. For the Acceleware solver, it corresponds to γ_n in Eqn. (22.11). For the MEEP solver, it corresponds to $\frac{\gamma_n}{2\pi}$ in Eqn. (22.16).
- **delta_eps#** sets the amplitude of the resonance peak in the Lorentz model. It corresponds to $\Delta\epsilon_n$ for both the Acceleware and MEEP solvers.
- **omega_drude#** corresponds to ω_n appearing in Eq. (22.12).

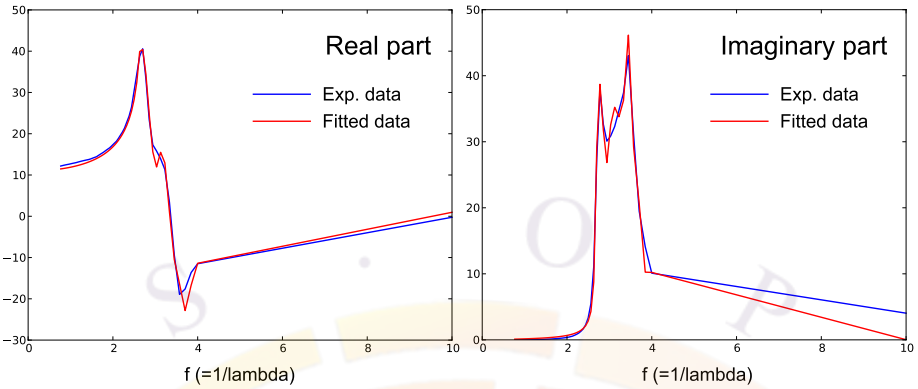


Figure 22.7: Fitted dielectric function for Silicon

- **gamma_drude#** corresponds to γ_n appearing in Eq. (22.12).
- **a_direct#** corresponds to A_n appearing in Eq. (22.13).
- **b_direct#** corresponds to B_n appearing in Eq. (22.13).
- **c_direct#** corresponds to C_n appearing in Eq. (22.13).
- **d_direct#** corresponds to D_n appearing in Eq. (22.13).
- **scale_absorption** scales strength of absorption of dispersion function. This scaling only works for single wavelength case.

Examples

Figure 22.7 shows an example of how the dielectric function of crystalline silicon is fitted with a sum of harmonic oscillators (Lorentzian model); nine oscillator terms were used in this fitting.

The horizontal axis corresponds to frequency $f = 1/\lambda$ using the MEEP convention which means that λ is the wavelength value in μm. The absorption peaks between $f = 2$ and $f = 4$ on Figure 22.7 correspond to a wavelength range of 250-500 nm.

The fitting was done by using non-linear least square fitting procedure. In this fitting, different weight was used at lower and higher frequency region. Especially, large weight value was assigned to lower frequency region($f \leq 2.5$) of the imaginary part, since fitting errors in this region cause large errors in the resulting optical absorption. Note that no automated fitting procedure is included in APSYS: it is the responsibility of the user to provide the correct fit.

The corresponding fdtd_dispersion command reads as follows:

```
fdtd_dispersion mater=1 order=9 &&
  type=Lorentz &&
  freq_convention=frequency &&
  input_unit=MEEP &&
  epsinf=2.05727379912 &&
  omega1=2.70630963121 gamma1=0.0792498919692 delta_eps1=0.622486271034 &&
  omega2=2.78980009035 gamma2=0.0698148765121 delta_eps2=0.735692163881 &&
  omega3=2.88533425743 gamma3=0.0877692173939 delta_eps3=0.764330138351 &&
  omega4=3.00784243943 gamma4=0.1224552227370 delta_eps4=0.914575597926 &&
  omega5=3.14120289007 gamma5=0.1277355532670 delta_eps5=0.911707698778 &&
  omega6=3.43582761159 gamma6=0.1694383912360 delta_eps6=1.719650817200 &&
  omega7=3.27736546304 gamma7=0.1298284642990 delta_eps7=1.043230189780 &&
  omega8=3.63115274113 gamma8=0.2545018091260 delta_eps8=1.386792385230 &&
  omega9=4.09954249395 gamma9=0.3634196648580 delta_eps9=0.778171989294
```

Here, freq_convention=frequency and input_unit=MEEP mean that horizontal axis used in our parameter fitting is frequency, and units follow the MEEP covention.

The interface program will convert fitting parameters automatically depending on the combination of freq_convention, input_unit and the FDTD solver chosent. The user is only responsible for fitting the experimental data using consistent freq_convention and input_unit settings.

If the experimental data was using a horizontal axis of angular frequency ω and SI units the equivalent fdtd_dispersion command would read as follows:

```
fdtd_dispersion mater=1 order=9 &&
  type=Lorentz &&
  freq_convention=omega &&
  input_unit=SI &&
  epsinf=2.05727379912 &&
  omega1=5.09774437845e+015 gamma1=1.49279183217e+014 delta_eps1=0.622486271034 &&
  omega2=5.25501131267e+015 gamma2=1.31506901563e+014 delta_eps2=0.735692163881 &&
  omega3=5.43496439623e+015 gamma3=1.65326623905e+014 delta_eps3=0.764330138351 &&
  omega4=5.66572712525e+015 gamma4=2.30662972233e+014 delta_eps4=0.914575597926 &&
  omega5=5.91693174711e+015 gamma5=2.40609275112e+014 delta_eps5=0.911707698778 &&
  omega6=6.47190206558e+015 gamma6=3.19162891213e+014 delta_eps6=1.719650817200 &&
  omega7=6.17341459110e+015 gamma7=2.44551590258e+014 delta_eps7=1.043230189780 &&
  omega8=6.83982655196e+015 gamma8=4.79392731643e+014 delta_eps8=1.386792385230 &&
  omega9=7.72210964398e+015 gamma9=6.84556021301e+014 delta_eps9=0.778171989294
```

Note the use of freq_convention=omega and input_unit=SI in this case.

22.303 fdtd_far_field

parameter	data type	values [defaults]
do_ffp_only	char	[no] yes
plane	char	[void] -y +y -x +x -z +z
data_file	char	[void] far_field.dat
subtract_source	char	[no] yes
boxlabel_normalize	char	[void] Box1
box_point_ll	realx2	
box_point_ur	realx2	
distance	real	[100.0]
range_theta	realx2	[0 180] (degrees)
range_phi	realx2	[0 360] (degrees)
plane_offset	char	(um)
ngrid_surfint	intgx3	
ngrid_theta_phi	intgx2	[10 20]

fdtd_far_field is used to calculate the far-field distribution of light scattered from a device in a FDTD simulation.

Parameters

- **do_ffp_only** directs the FDTD program to only calculate the far-field pattern (FFP). With this setting, the FDTD program will just read temporary data generated by a previous FDTD run and calculate the FFP.
- **plane** is used to specify the direction of the FFP plane axis. For example, **plane**=+y means that FFP will be calculated on a plane perpendicular to y-axis and the sign specifies the direction of the plane normal. The position of the plane should be specified by the **plane_offset** parameter. Note that **plane** cannot be specified at the same time as **box_point_ll** and **box_point_ur**.
- **data_file** is the name of a file where FFP information will be stored.
- **subtract_source** is used to instruct the simulator to substract the contribution of the incident wave from the FFP result. This directive is useful when calculating reflectance spectrum from FFP.
- **boxlabel_normalize** is a user-defined label specifying a monitoring box. This box is used to record the radiation power or the current source(s) in ordert to normalize the FFP result. The label must match the one used in the box declaration in fdtd_monitor_box.

- **box_point_ll** and **box_point_ur** are the lower-left and upper-right corner points (in APSYS coordinates) of an integration box used in the calculation of the FFP. The integration box should be large enough to contain the whole device but must exclude the absorbing boundary.

- **distance** is the distance between far-field point and the device. In principle, FFP does not depend on the **distance** parameter, so users do not have to care about this parameter.

- **range_theta** specifies the range of θ, which is the angle measured from Z-axis.

- **range_phi** specifies the range of ϕ, which is the angle measured from X-axis.

- **plane_offset** is an offset value of FFP plane from origin of cartesian coordinate. For example, **plane**=+y and **plane_offset**=5.0 means that XZ plane with +y normal direction will be located at y=5.0(um). In case of 2-D, Z-dimension is zero.

- **ngrid_surfint** is the grid size used to discretize the integration box. Since simulator automatically assigns this parameter by counting FDTD cells over FFP box/plane, users do not have to set this parameter.

- **ngrid_theta_phi** is the number of divisions for the θ and ϕ angles.

Examples

```
fdtd_far_field &&
  box_point_ll=(0.0 -0.5 0.0) &&
  box_point_ur=(5.0 5.5 0.0) &&
  range_phi = (0.0 180.0) &&
  ngrid_theta_phi = (1 51)
```

22.304 fdtd_field_monitor

parameter	data type	values [defaults]
tag	char	
target_region_tag	char	
simulation_space	char	emptyspace, devicespace
monitor_comp	char	Ex, Ey, Ez, Efields, Hx, Hy, Hz, Hfields, all
surface_only	char	yes,[no]
FFT	char	[yes], no
interval	intg	10
step_start	intg	0
step_end	intg	-9999
resolution	realx3	(0.5,0.5,0.5)

The command **fdtd_field_monitor** instructs the FDTD solver to monitor and record certain field components for later use. This command only works for CLFDTD.

Parameters

- **tag** is user-defined name which allows the recorded fields to be referenced in other commands. This name must be unique and not reused by other **fdtd_field_monitor** commands.
- **target_region_tag** is a region tag from **fdtd_define_region** which defines the simulation region being monitored.
- **simulation_space** specifies which simulation is being monitored: all FDTD simulations model field propagation in an empty region and in a region where the device exists.
- **monitor_comp** specifies which field components are being monitored.
- **surface_only** directs the program to monitor only the surface of the specified region.
- **FFT** directs the program to apply a FFT at each of the monitor points.
- **interval** is the interval time step used for monitoring.
- **step_start** controls the start time of the monitoring: $t_0 = \textbf{step\_start} * dt$.
- **step_end** controls the stop time of the monitoring: $t_f = \textbf{step\_end} * dt$.

- **resolution** specifies the resolution of the monitoring on each axis. This defines a subset of the FDTD grid points used in the simulation so this parameter scales between 0.0 and 1.0: for example, **resolution**=(1.0 1.0 1.0) means that all the FDTD grid points are being monitored.

Examples

```
fdtd_fieldmonitor &&
   target_region_tag=planeXY &&
   simulation_space=devicespace &&
   monitor_comp=Ez &&
   FFT=no &&
   step_start=0 &&
   step_end=2000 &&
   interval=100 &&
   resolution=(0.5 0.5 0.5) &&
   tag=field
```

22.305 fdtd_fourier

parameter	data type	values [defaults]
input_file_empty_space	char	
input_file_device_space	char	
output_file	char	fdtd_fourier.dat
windowing	char	[no],yes
field_comp	char	[Ex],Ey,Ez
time_start	real	
time_end	real	
window_center	real	
window_FWHM	real	
read_interval	intg	[0],1,2,3
num_freq	intg	[100],200,300

The command **fdtd_fourier** is used to set parameters for the Fourier transform of E-fields. Amongst other things, this can be used to estimate the resonance frequency of a high-Q cavity.

All time positions/durations should be specified in the same unit as the MEEP simulation if it was used to generate the original data. Otherwise, a multiple of the simulation time step is used in the Acceleware and Crosslight FDTD solvers.

Parameters

- **input_file_empty_space** is the name of the file which contains the time evolution of E-fields for the "empty space" FDTD simulation.
- **input_file_device_space** is the name of the file which contains the time evolution of E-fields for the "device space" FDTD simulation.
- **output_file** is the name of file where the Fourier transformed results will be stored.
- **windowing** tells the simulator to apply a Gaussian window to the transform. This may be useful in limiting aliasing.
- **time_start** is the start time of the Fourier analysis; this may be set to avoid the initial transient response.
- **time_end** is the end time of the Fourier analysis.
- **window_center** is the center time of Gaussian windowing function.
- **window_FWHM** is the FWHM of the Gaussian windowing function.
- **read_interval** is used to specify read interval. This parameter is useful when user wants to avoid oversampling of the data.
- **num_freq** is the number of frequency points in the spectrum.

Examples

```
fdtd_fourier input_file_empty_space=efields_empty.dat &&
             input_file_device_space=efields_device.dat &&
             num_freq=300
```

22.306 fdtd_glass_coating

parameter	data type	values [defaults]
spectrum_file	char	
fdtd_refl_file	char	
position	char	[top]bottom
do_precalc	char	[no]yes
thickness	real	0 (μm)
real_index	real	3.2
imag_index	real	0.0

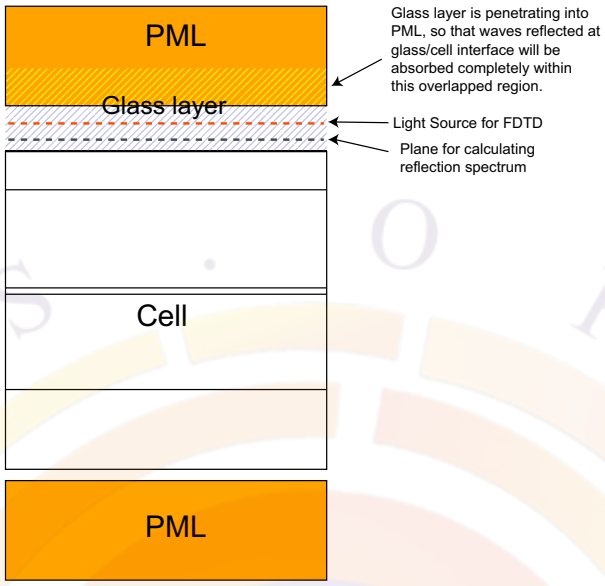


Figure 22.8: First step of simulation with **fdtd_glass_coating**

---

fdtd_glass_coating sets parameters for the FDTD simulation with a thick glass coating. Currently, this command only works for MEEP.

Under normal circumstances, a thick layer would require too much memory and computational layer for a FDTD simulation since it uses such a small grid size. We thus use a hybrid approach combining plane wave transfer matrix formalism with FDTD and split the simulation into two steps.

In the first step, the glass layer is embedded in the FDTD's PML layer to isolate the reflection coefficient of the cell (Fig. 22.8). In the second step, this is used to compute the effective plane wave transfer matrix of the glass layer and create a transfer function. The glass layer is then removed from the FDTD simulation and a filtered light spectrum is used to represent its effects (Fig. 22.9).

- **spectrum_file** is the name of a file which contains complex refractive index spectrum of the glass. If not specified, constant refractive index given by **real_index** and **imag_index** will be used.

- **fdtd_refl_file** is the name of the file where the reflection spectrum is recorded in the precalculation (first step). During the second step, the reflection spectrum is read from this file and used to calculate the filtered light spectrum.

- **position** is the location of the glass layer.

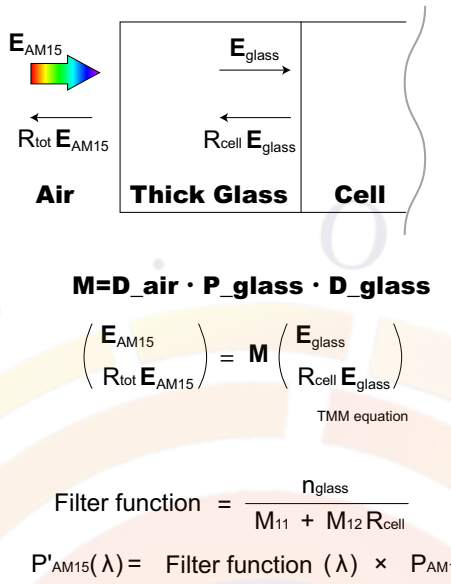


Figure 22.9: Second step of simulation with **fdtd_glass_coating**

- **do_precalc** determines if this simulation will be the first or second step as outlined above.

- **thickness** is thickness of the glass layer in μm.

Examples

The example below shows the pre-calculation of reflection spectrum: this is the first step discussed above. The reflection spectrum is defined by using the **fdtd_plane_refl** to define a monitoring plane.

```
fdtd_glass_coating &&
    do_precalc = yes &&
    fdtd_refl_file=reflection_precalc.dat &&
    real_index=1.57  imag_index=0.0e0 &&
    position=bottom thickness=0.5

fdtd_plane_refl center=(0.05 0.05 -0.1) size=(0.1 0.1 0.0)
```

22.307 fdtd_group_monitors

parameter	data type	values [defaults]
tag	char	
monitor_tagj (j=1..9)	char	

The command **fdtd_group_monitors** is used to group together FDTD field monitors. This command only works for CLFDTD.

Parameters

- **tag** is a user-defined name for this group of FDTD field monitors and which allows it to be referenced by post-processing command such as **fdtd_data_analysis**. The **tag** must be unique and not be reused by either the **fdtd_group_monitors** or **fdtd_field_monitor** commands.
- **monitor_tagj(j=1..9)** is the label of the jth field monitor that is being grouped together under the same label. This value is inherited from **fdtd_field_monitor**.

Examples

```
fdtd_group_monitors &&
  monitor_tag1=influx_monitor &&
  monitor_tag2=outflux_monitor &&
  tag=flux_monitor_group
```

22.308 fdtd_model

parameter	data type	values [defaults]
export_var	char	[density],flux_x,flux_y,flux_z,total_flux
export_grid	char	[FEM],ALL,NOBG
export_data	char	[meep_density.dat]
auto_finish	char	[no],yes
hdf5out_field	char	[no],yes
hdf5out_comp	char	[Ez], Ex, Ey, Hx, Hy, Hz, EnergyDensity
hdf5tovtk	char	[no],yes

parallel	char	[no],yes
interface	char	[MEEP], Ax, CLFDTD
subpixel_averaging	char	[no],yes
use_gpu	char	[no],yes
wavel_range	realx2	(um)
PML_thickness	real	(um)
fixed_time	real	[20.0] (unit depends on FDTD interface)
buffer_x	realx2	[0.0 0.0] (um)
buffer_y	realx2	[0.0 0.0] (um)
buffer_z	realx2	[0.0 0.0] (um)
auto_dt	real	[10.0]
auto_dt2	real	
auto_until_ratio	real	[0.01]
watch_pointk (k=1..9)	realx3	(um)
hdf5out_dt	real	[2.0] (unit depends on FDTD interface)
cell_size	realx3	(um)
fdtd_meshdensity	real	(um)
adjust_xdim	real	(um)
adjust_ydim	real	(um)
adjust_zdim	real	(um)
courant_factor	real	
nb_wavel	intg	[100]
nb_mesh	intgx3	[20 20 0]
boundary_type	intgx3	
num_zero_optgen_mater	intg	[0]
n_zero_optgen_mater	intgxn	
npe_para	intg	[1]
iauto_dt_step	intg	
iauto_dt2_step	intg	

fdtd_model sets all the parameters for a FDTD (Finite-Difference Time Domain) simulation using APSYS device data. The FDTD algorithm uses a regular grid (Yee lattice) so there is a mapping of the device structure and the irregular finite element data to/from APSYS as shown in Fig. 22.10.

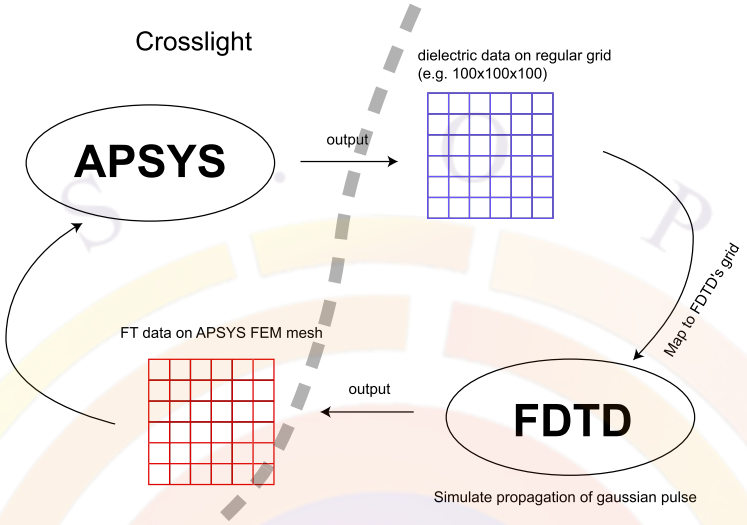
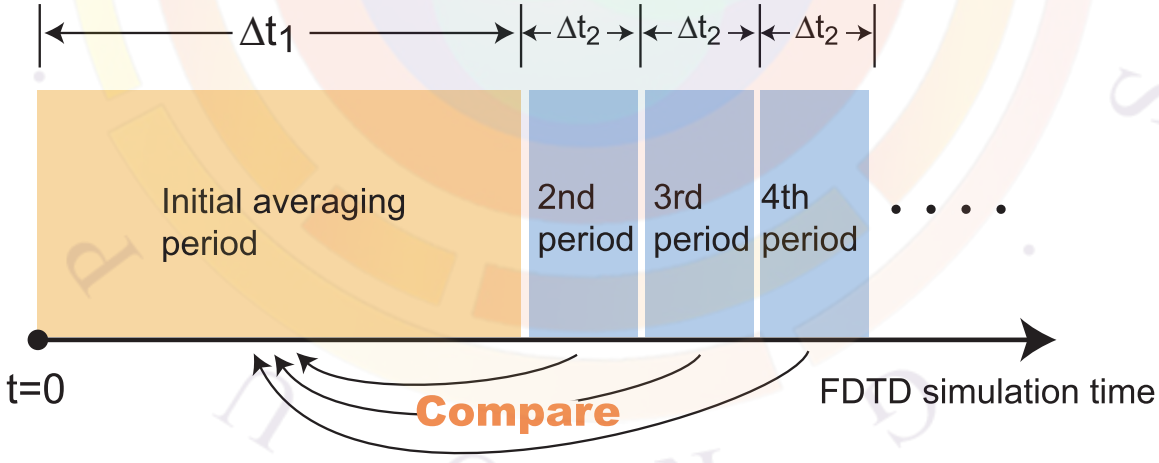


Figure 22.10: Schematic of the interface between the FDTD program and APSYS



Δt1 should be long enough so as to catch propagating wave at every watch points.

Figure 22.11: Schematic of auto-finish criterion used in FDTD.

Parameters

- **export_var** is the variable to export from FDTD calculation to be used by device simulator. For standard device simulation, only **export_var**=*density* is acceptable since the photon density S must be used in the optical generation and stimulated recombination terms. The other settings define flux values which are only useful for debugging purposes.

- **export_grid** defines the type of grid data used to export FDTD simulation results. Since the goal of FDTD simulations is usually to import back the photon density into APSYS, only the *FEM* should be used; the other setings are used for debugging purposes.

- **export_data** is the file name that the FDTD program outputs for use in the device simulator. This file name may be used in subsequent APSYS simulations in the **import_fdtd_data** statement.

- **auto_finish** directs the FDTD program to automatically judge if the FDTD simulation has run long enough to reach a steady state. This situation is detected using the parameters **auto_dt**, **auto_dt2**, **auto_until_ratio** and **watch_pointk (k=1..9)**.

  To determine if the simulation should end, the FDTD simulation monitors the magnitude of the electric field at each watch point. The first monitoring period begins from $t = 0$ and ends at $t =$ **auto_dt**. The maximum value of the field magnitude in the first monitoring period is defined as E^1_{max}.

  After the first period, a second monitoring will take place for $\Delta t =$ **auto_dt2**, and we obtain E^2_{max}. Successive monitoring periods with the same length continue until E^n_{max} becomes smaller than $E^1_{max} \times$ **auto_until_ratio**. A schematic of this criterion is shown in Fig. 22.11.

  Note that **auto_finish** can only be used in simulations where the fields can decay properly so if the FDTD simulation uses a single wavelength, then the time-domain source is a sine wave which goes on forever and **auto_finish** may not be used. PML boundaries are also recommended when using **auto_finish** as it prevents outgoing waves from traveling back into the simulation domain.

- **hdf5out_field** is a switch to turn on/off field output in the MEEP solver and is ignored in the Ax and CLFDTD solvers. The component of the field that is output is specified by **hdf5out_comp** parameter. The output is done at specific time intervals controlled by **hdf5out_dt**.

- **hdf5tovtk** is used to convert HDF5 files generated by the MEEP solver to the VTK format; file conversion will take place at the end of FDTD simulation. VTK files can be opened by several free programs such as MayaVi, ParaView

and VisIt. This parameter is ignored when using the Ax and CLFDTD solver since they natively use the VTK format and do not require conversion.

- **parallel** is used to enable parallel execution of the FDTD simulation; the number of parallel jobs is defined by the **npe_para** parameter. This parameter is ignored when **interface**=*Ax*.

  Note that unlike the multi-threaded sparse solver used in the main APSYS simulation, parallel FDTD relies on the Message Passing Interface (MPI). The Intel® MPI run-time libraries should be installed on the end-user's computer to add the necessary DLL support and the mpiexec launcher: only the free run-time is needed for end-users, not the paid developer version.

  As mpiexec also requires user credentials to launch the parallel FDTD jobs, default user credentials should also be added to the system prior to running the .sol file; otherwise, the mpiexec launcher will ask for user credentials before starting the FDTD simulation. Defining default credentials allows the software to run unattended from batch files and also avoids incompatibilities with the display-only view of the output that is used in the SimuAPSYS GUI.

  The default user credentials may be encrypted into the Windows registry using the following syntax, using either a command shell or from "Run" input box in the Windows start menu:

  ```
  mpiexec -register
  ```

  The above step only needs to be run a single time for each computer and should only be re-run as needed to replace these user credentials; the `-remove` option may be also used to delete existing credentials from the registry. We also wish to warn users that the wmpiconfig program from the start menu ***WILL NOT*** store the user credentials needed for mpiexec: it instead stores the credentials for the other MPI launcher (wmpiexec) which is not used by APSYS.

  Note that MPI-based acceleration is currently incompatible with GPU acceleration.

- **subpixel_averaging** is used to turn on/off the sub-pixel averaging feature of MEEP. This parameter is ignored in the Ax and CLFDTD solvers.

- **wavel_range** is the wavelength range the FDTD simulation should cover. It is related to the FDTD time step by the Nyquist sampling criterion. This range should be big enough to cover the entire spectrum defined in the **light_power** statement.

- **PML_thickness** is the thickness of the PML layer inside the air buffer. This should be thick enough to absorb electromagnetic waves completely and so

should be at least equal to the maximum value of **wavel_range**. The axes on which the PML boundary are defined by the **boundary_type** parameter.

Note that this parameter is ignored when **interface**=*Ax*, since the interface program for the Acceleware FDTD solver automatically applies a five layer CPML model.

- **fixed_time** should be specified in case of **auto_finish**=*no*, in which case the FDTD simulation runs for a predetermined number of time steps.

  The units of **fixed_time** depend on the solver being used. In MEEP, internal scaling of variables generates a simulation time reference unit of 3.3333333 fs; this is based on a reference unit length of 1 μm and is a distinct value from the time step Δt. As a result, the total simulation time for the FDTD simulation is **fixed_time** $\times$ 3.3333333 $\times$ 10^{-15} seconds.

  In the Ax and CLFDTD solvers, **fixed_time** acts a multiplier to the Δt time step. This value is determined automatically by the cell size and refractive index and is recorded in the simulation log file.

- **buffer_x**, **buffer_y**, **buffer_z** are used to denote empty space added to negative (first value) and positive (second value) sides of the simulated device in the x,y and z directions, respectively (e.g. left/right, bottom/top, etc...). This sets the effective size of the FDTD simulation so the user must be careful is choosing a value that is not too big to save on computation time. However, the PML layer is also contained in this empty space so the buffer must be large enough to encompass both the PML layer and sufficient empty space to position monitor points and other reference positions where the outside field is needed.

- **auto_dt** and **auto_dt2** are the lengths of the averaging periods used in the auto_finish criterion in MEEP and use the same time reference as the **fixed_time** setting. They are shown in Fig. 22.11 as Δt_1 and Δt_2, respectively.

  For the Acceleware and Crosslight FDTD interfaces, please use **iauto_dt_step** and **iauto_dt2_step** instead. These values are integer multipliers of the Δt time step which are used to define Δt_1 and Δt_2.

- **auto_until_ratio** is a ratio used to judge if the electric field has decayed to a sufficiently small value that the FDTD simulation can be terminated. For example, when **auto_until_ratio**=0.01, the simulation will finish if the magnitude of electric field is 100 times smaller than that of first averaging period.

- **watch_point1-9** are the coordinates of "watch points" that are used to monitor the magnitude of electric field for auto finish feature. The coordinates of these watch points use the same reference as the original FEM mesh so that

watch points set in the air buffer outside the device may have values which go out of bounds.

- **cell_size** is the size of the FDTD simulation cell. The FDTD space(=APSYS device + empty buffer space) is discretized by a large number of rectangular cuboid cells (Yee lattice); this array defines the edge length of each cell along the x, y and z directions. Once the cell size is set, the number of grid points is determined based on the total size of the FDTD simulation space: if this space cannot be filled by an integer multiple of cells, the simulator will adjust the buffer size to fit.

  Note that instead of setting the cell size directly **fdtd_meshdensity** (in $1/\mu m$) may be used to sample the original APSYS data and set the cell size; this setting does not use the total size of the FDTD space to set the cell size. The total number of grid points on each axis can also be set manually using **nb_mesh**, which has the effect of setting the cell size (=total size/nb_mesh); users are reminded that some of these points will be used for the air buffer around the device.

  All three methods of defining the FDTD grid size are mutually incompatible: users should only define one of the above parameters in the .sol file.

- **interface** is used to switch between the different FDTD solvers available. In many cases, Crosslight only provides an interface to an existing FDTD solver library which may be licensed under different terms than APSYS. The options currently available are:

  – MEEP: A free FDTD library that supports MPI acceleration and which is licensed under the GPL. The source code of the interface program between APSYS and the MEEP FDTD solver library available on request to comply with GPL requirements.
  http://ab-initio.mit.edu/wiki/index.php/Meep

  – Ax: A proprietary FDTD solver library from Acceleware®. The library must be purchased separately from the vendor, with Crosslight providing an interface program; supports GPU hardware acceleration.
  http://www.acceleware.com/

  – CLFDTD: Crosslight's own FDTD implementation, new to the 2012 version of the software. Supports both MPI and GPU acceleration.

  Please contact your local Crosslight sales agent to learn more about the difference between the various solvers and the optional modules required to access the various FDTD interfaces available through this statement.

- **adjust_xdim**,**adjust_ydim** and **adjust_zdim** are used to adjust the dimension of X,Y or Z coordinates of the device structure. Such adjustment is

useful when the APSYS mesh is imported from another source (e.g. CSUPREM) and there are tiny errors in the mesh.

- **courant_factor** specifies the Courant factor of the FDTD simulation. The default value is 0.5.
- **nb_wavel** is the number of wavelength points for data export. In FDTD simulation, the wavelength range **wavel_range** is converted to frequency range, then divided by **nb_wavel**. Therefore, data points are distributed uniformly in frequency space.
- **boundary_type** represents boundary type along each axis. The only supported boundaries for FDTD are 0 (PML) and 1 (periodic).

  PML boundaries are used to represent an outgoing wave propagating towards infinity. This Perfectly Matched Layer absorbs the outgoing wave so that it reaches zero on the edges of the simulation domain but unlike a traditional absorbing region, there is no refractive index step which would cause a reflection of the wave back into the simulation domain.

  Periodic boundaries are used to sample infinite periodic variations of the dielectric constant; the outgoing wave on one side becomes an incoming wave on the opposite side.
- **num_zero_optgen_mater** is a number of the materials of whose optical carrier generation rate is forced to zero. The optical energy density in those materials is also forced to zero. The actual list of material numbers where this setting is active is given by **n_zero_optgen_mater** parameter.

  This option may prove necessary when defining dispersion curves with **fdtd_dispersion**. A poor quality fit may introduce an imaginary component to the refractive index which would be unphysical. Note that by default, FDTD simulations use only a single dielectric constant/refractive index so this option is not usually required.
- **use_gpu** turns on GPU acceleration of FDTD solvers that support this feature. This normally requires Nvidia® cards as the solvers are written using the CUDA™ library. This setting is incompatible with MPI-based acceleration.

Application Notes

FDTD is an algorithm which parallelizes very naturally so using MPI or GPU accelerations is strongly recommended. However, the choice of the acceleration method depends on the situation.

GPUs are known for their large number of cores compared to CPUs, which more than make up for the slower speed of the individual cores. The downside is that GPU cards

often have a limited amount of on-board memory compared to the system memory; this introduces performance penalties (swapping between system RAM and GPU RAM) if the FDTD grid cannot be stored entirely within GPU memory. Multiple GPU cards may be used to partially offset this limitation but most workstations cases and motherboards do not have sufficient space to expand and meet the needs of larger 3D problems.

MPI parallelism tends to be slower the GPU for smaller problems due to the limited number of processing cores than can be brought to bear on the problem. However, regular CPUs can access the entire RAM on the system which makes it more suitable for larger problems which cannot fit into the GPU memory. MPI technology is also a base building block for modern supercomputer clusters which allows the FDTD workload to be split between several computers, thereby expanding the available memory and the number of processing cores.

Users are therefore advised to estimate the amount of memory required for their FDTD problem and pick the best parallel solution. Since each grid point stores 6 variables (E and H field components) in IEEE754 double precision, the memory usage can be approximated based on the total grid size:

$$N_x \times N_y \times N_z * 48\text{bytes}$$

Total computation time can be harder to estimate since it depends on the number of time step as well as the number of grid points. The total number of operations can be approximated by:

$$N_{op} = N_x \times N_y \times N_z \times \frac{\text{fixed\_time}}{\Delta t}$$

Comparing the run time and number of operations of a small FDTD simulation to the number of operations in a larger problem can help estimate the run time of this larger problem.

Examples

The **fdtd_model** statement below is an example for 2-dimensional FDTD simulation using the MEEP interface. A periodic boundary condition is applied only to the x-axis.

```
fdtd_model export_var = density wavel_range =  [0.3,2.5]   &&
    PML_thickness = 1.0 boundary_type =  [1,0,0]   &&
    buffer_y =  [2.0,2.0]  nb_wavel = 20 nb_mesh =  [50,300,0]   &&
    auto_finish = yes auto_dt = 40 auto_dt2 = 5 &&
```

```
watch_point1 =  [2.5,3.0,0] auto_until_ratio=0.1 &&
hdf5out_field=yes hdf5out_comp=Ex hdf5out_dt=5.0 &&
parallel=yes npe_para=4
```

22.309 fdtd_modify

parameter	data type	values [defaults]
efields_monitor_file_prefix	char	
remove_FEM	char	[yes],no
iauto_dt_step_emptyspace	int	
iauto_dt2_step_emptyspace	int	
auto_dt_emptyspace	real	
auto_dt2_emptyspace	real	
fixed_time_emptyspace	real	

The command **fdtd_modify** provides a way to modify some FDTD control parameters.

Parameters

- **efields_monitor_file_prefix** modifies the prefix string of E-field monitor file.
- **remove_FEM** is a switch to direct simulator to disable FFT of E-fields on APSYS FEM mesh points. This switch is useful to reduce computation time only when user wants to obtain results other than relative energy intensity distribution, e.g. resonance frequency analysis.
- **iauto_dt_step_emptyspace** will override **iauto_dt_step** for empty space FDTD simulation (for Acceleware FDTD only)
- **iauto_dt2_step_emptyspace** will override **iauto_dt2_step** for empty space FDTD simulation (for Acceleware FDTD only)
- **auto_dt_emptyspace** will override **auto_dt** for empty space FDTD simulation (for MEEP only)
- **auto_dt2_emptyspace** will override **auto_dt2** for empty space FDTD simulation (for MEEP only)

- **fixed_time_emptyspace** will override **fixed_time** for empty space FDTD simulation.

Examples

```
fdtd_modify fixed_time_emptyspace=100.0
```

Example above modifies **fixed_time** parameter of **fdtd_model** only for empty space FDTD simulation. It is useful to reduce computation time of empty space simulation, since convergence and propagation in empty space is much faster than that in device space.

22.310 fdtd_monitor_box

parameter	data type	values [defaults]
data_file	char	[void]
label	char	[void]
monitor_space	char	[vacuum] device
center	realx3	(um)
size	realx3	(um)
box_point_ll	realx2	(um)
box_point_ur	realx2	(um)

fdtd_monitor_box is used to calculate radiation power of current source by integrating the flux on each side of a monitoring box. This command is useful when normalizing FFP by source power.

Parameters

- **data_file** is the name of a file where the integrated source power will be stored.
- **label** is used to assign a label to the monitoring box. This label can be refered by the **boxlabel_normalize** parameter of the **fdtd_far_field** command.
- **monitor_space** specifies which FDTD simulation, i.e. vacuum or device, is to be used as the source power.
- **center** and **size** are the center position and the size of the monitoring box.

- **box_point_ll** and **box_point_ur** are the lower-left and upper-right corners (in APSYS coordinates) of the monitoring box.

Note that **center**,**size** and **box_point_ll**,**box_point_ur** are mutually exclusive.

Examples

```
fdtd_monitor_box &&
  center=(3.25 6.05 0.0) size=(0.2 0.2 0.0) &&
  label=Box1
```

22.311 fdtd_output_structure

parameter	data type	values [defaults]
variable	char	material_num, real_epsilon, imag_epsilon, real_index, imag_index
data_file	char	
wavelength	real	10.0
resolution	realx3	(1.0 1.0 1.0)

The command **fdtd_output_data** is used to output structural data from the FDTD simulation to a file in the VTK format. This command only works for CLFDTD.

Parameters

- **variable** controls the structural data being saved.
- **data_file** is the name of the VTK file.
- **wavelength** is used to evaluate wavelength-dependent variables before export.
- **resolution** specifies the resolution of the structural data on each axis. This defines a subset of the FDTD grid points used in the simulation so this parameter scales between 0.0 and 1.0: for example, **resolution**=(1.0 1.0 1.0) means that all the structural data on the FDTD grid points will be output to the data file.

Examples

```
fdtd_output_structure &&
  variable=material_num &&
  resolution=(1.0 1.0 1.0) &&
  data_file=materID
```

22.312 fdtd_plane_refl

parameter	data type	values [defaults]
data_file	char	
center	realx3	(um)
size	realx3	(um)

The command **fdtd_plane_refl** defines a plane which monitors the flux of electromagnetic waves reflected back to this plane. Results are be stored in a data file as a reflection spectrum.

Currently, this command only works for MEEP.

Parameters

- **data_file** is the name of the output data file. If not specified, a file name "reflection.dat" will be used as default.
- **center** is the center position of the monitoring plane in APSYS coordinates. It may take negative values to mean being placed below the device (for y-direction): the FDTD simulation space must be large enough to accommodate this.
- **size** is the size of monitoring plane.

Examples

```
fdtd_plane_refl center=(2.5 5.5 0.0) size=(5.0 0.0 0.0)
```

22.313 fdtd_plane_trans

parameter	data type	values [defaults]
data_file	char	
center	realx3	(um)
size	realx3	(um)

The command **fdtd_plane_trans** defines a plane which monitors the flux of electromagnetic waves transmitted through this plane. Result are be stored in a data file as a transmission spectrum.

Currently, this command only works for MEEP.

Parameters

- **data_file** is the name of the output data file. If not specified, a file name "transmission.dat" will be used as default.
- **center** is the center position of the monitoring plane in APSYS coordinates. It may take negative values to mean being placed below the device (for y-direction): the FDTD simulation space must be large enough to accommodate this.
- **size** is the size of monitoring plane.

Examples

```
fdtd_plane_trans center=(2.5 -0.5 0.0) size=(5.0 0.0 0.0)
```

22.314 fdtd_push_job

parameter	data type	values [defaults]
job_id	char	

The command **fdtd_push_job** is used to schedule an immediate call to the FDTD solver. Multiple calls to the FDTD solver can be used in the same .sol file by issuing this command to separate the parameters for each simulation.

Parameters

- **job_id** is the label of the FDTD job.

Examples

```
$ FDTD simulation parameters for job #1 go here
$ .....
fdtd_push_job job_id=job1

$ FDTD simulation parameters for job #2 go here
$ .....
fdtd_push_job job_id=job2
```

22.315 fdtd_replace_FDTDgrid

parameter	data type	values [defaults]
file	char	

The command **fdtd_replace_FDTDgrid** will replace the FDTD grid data (which is normally imported from APSYS) by the user-specified data file. Number of grid points and data ordering should be consistent with other FDTD settings.

Parameters

- **data_file** is the name of a user-supplied data file.

Examples

```
fdtd_replace_FDTDgrid file=MyGridData.txt
```

22.316 fdtd_replace_mater

parameter	data type	values [defaults]
op1	intgx2	
op2	intgx2	
op3	intgx2	
op4	intgx2	
op5	intgx2	
op6	intgx2	
op7	intgx2	
op8	intgx2	
op9	intgx2	
x_range	realx2	
y_range	realx2	
z_range	realx2	

The command **fdtd_replace_mater** will substitute one material number for another within the specified spatial range.

Parameters

- **op1-9** are used to specify replace operation.
- **x_range** specifies spacial range along x-axis.
- **y_range** specifies spacial range along y-axis.
- **z_range** specifies spacial range along z-axis.

Examples

fdtd_replace_mater command shown below will replace material number 2 with 1 in y-coordinate ranging from 0.0 to 2.0.

```
fdtd_replace_mater op1=(2 1) y_range=(0.0 2.0)
```

22.317 fdtd_source

parameter	data type	values [defaults]
component	char	[void],Ex,Ey,Ez,Hx,Hy,Hz
auto_positioning	char	[no] yes
auto_side	char	[above] below
source_type	char	[dipole],planewave,GaussianBeam
polarization	char	[void] TM, TE
wavel	real	
spectral_width	real	
amplitude	realx2	[1.0 0.0] (unitless for MEEP, A/m^2 for the others)
center	realx3	(um)
size	realx3	(um)
auto_rel_pos	real	[0.5]
limit_x	realx2	(um)
limit_y	realx2	(um)
limit_z	realx2	(um)
dir_planewave	realx3	(um)
theta_planewave	real	(deg.)
phi_planewave	real	(deg.)
beam_waist	real	(um)
beam_size	real	(um)
theta_beam	real	(deg)
phi_beam	real	(deg)
beam_points	int	

The command **fdtd_source** defines current sources for FDTD simulations. The FDTD program decides reasonable pulse width/duration based on the spectrum width, which is specified by the **wavel_range** parameter of the **fdtd_model** statement.

Note that multiple sources add up coherently.

Parameters

- **component** is a component of the current source. Electromagnetic fields induced by this current source depend on the device geometry.
- **auto_positioning** turns on/off auto-positioning feature of source plane. If **auto_positioning**=yes, simulator will put the source plane between device

and the PML boundary.

- **auto_side** specifies the side of auto-positioning location. If **auto_side**=above(below), source plane will be located above(below) the device. This parameter is effective only when **auto_positioning**=yes.
- **source_type** is the type of current source.
- **polarization** will set the polarization direction of the plane wave model automatically. This parameter is effective only when **auto_positioning**=yes.
- **wavel** is the center wavelength of FDTD current source. If not specified, center wavelength is automatically calculated from the **wavel_range** parameter of fdtd_model.
- **amplitude** is a complex number specifying amplitude of current source pulse. It is advised that this is not altered except by FDTD experts.
- **center** is the center position of current the source in the original FEM coordinates used by APSYS. It may take negative values to mean being placed below the device (for y-direction). The FDTD simulation space will need to be large enough to encompass the source region.
- **size** is the size of current source. User can set various type of current source (e.g. point, linear shape, planar shape and 3D-box source) by changing the size parameter. To model a plane wave, we can flatten a 3D box source by setting one of its dimensions to zero.
- **auto_rel_pos** is used to specify the relative position of source plane. For example, **auto_rel_pos**=0.5 will put source plane at middle position between device and PML. **auto_rel_pos**=0.0 and 1.0 correspond to boundary of device and PML, respectively.
- **limit_x** is useful when user wants to restrict the x-range of source plane.
- **limit_y** is useful when user wants to restrict the y-range of source plane.
- **limit_z** is useful when user wants to restrict the z-range of source plane.
- **dir_planewave** is a direction vector(k-vector) of for the plane wave model. User SHOULD NOT specify **theta_planewave** nor **phi_planewave** along with this parameter.
- **theta_planewave** is an angle θ, i.e. angle measured from z-axis. User SHOULD NOT specify **dir_planewave** along with this parameter.

- **phi_planewave** is an angle ϕ, i.e. angle measured from x-axis within xy-plane. User SHOULD NOT specify **dir_planewave** along with this parameter.
- **beam_waist** is the waist size of a Gaussian beam. Amplitude and intensity of Gaussian beam drop to $1/e$ and $\frac{1}{e}$ at **beam_waist**, respectively.
- **beam_size** is the size of Gaussian beam source.
- **theta_beam** is the angle θ, which corresponds to the angle between the Gaussian source plane and z-axis.
- **phi_beam** is the angle ϕ, which corresponds to the angle between the Gaussian source plane and x-axis.
- **beam_points** is the number of source points distributed along the Gaussian source plane.

Examples

```
fdtd_source component=Ez  &&
  center=(5.0 2.0 0.0) size = (10.0 0.0 0.0)
```

The above is a linear current source of width 10 μm with center located at x=5, y=2.

22.318 fit_gain_wavel

parameter	data type	values [defaults]
fit_data_file	char	[fitgain.txt]
fit_density	real	[2.e24] (m^{-3})
fit_density_p	real	[-9999.]
wavel_range	realx2	[-9999. -9999.] (μm)
av_index	real	[3.3]
density_vary	real	[0.2e24] (m^{-3})
data_point	intg	[100]

This command is used in the .gain preview and fits a series of gain curves to the analytical model shown in Sec. 22.65.

Parameters

- **fit_data_file** is the output data file for the fitted parameters.
- **fit_density** is the electron density at which the fit is made.
- **fit_density_p** is the hole density at which the fit is made. By default, this is the same as the electron density.
- **wavel_range** is the wavelength range of the fit.
- **av_index** is an average index value used in the gain calculation.
- **density_vary** is the change in carrier density used evaluate derivatives in the analytical model.
- **data_point** is the number of wavelength points used in the fit.

Examples

```
fit_gain_wavel fit_density=1.5e24
```

22.319 force_last_barrier_offset

parameter	data type	values [defaults]
all_complex	char	[yes], no
value	real	[0.0] (eV)
complex	intg	

force_last_barrier_offset forces the conduction band in the last region of a complex MQW to be at a specific position, relative to the first region. This overrides the normal band alignment rules defined in Sec. 10.1; it also instructs the software to use other aspects of the quantum well model for that region, such as the carrier masses.

This statement is only used for the simplified complex library system (c.f. Sec. 3.5.1).

Parameters

- **all_complex** applies the same correction term to all complex MQWs in the simulation. It this is set to *no*, **complex** must be set to identify the MQW region affected by this statement.

- **value** defines the offset of the last barrier, relative to the leftmost barrier.

22.320 fourier_power

parameter	data type	values [defaults]
data_file	char	[void]
input_var	char	
output_var	char	
facet	char	[void] front/back
log_freq	char	[no]
freq_start	real	[3.] (decade)
freq_end	real	[10.] (decade)
scale_lit	real	[2.]
scale_curr	real	[2.]
scale_horizontal	real	[1.]
scale_vertical	real	[1.]
mode_index	intg	[1]

The statement **fourier_power** is used in the post-processing step to obtain the system frequency response function from transient simulation data. The program takes the time-dependent input and output variables, makes a Fourier transform of each and divides their magnitude to obtain the response function.

For example, given Vin(t) and Vout(t) and their respective Fourier transforms, U_{in} and U_{out}, the program computes $20log_{10}\left|\frac{U_{out}}{U_{in}}\right|$.

Parameters

- **data_file** is the data file name to which ASCII is exported.
- **input_var** and **output_var** are the input and output variable names, respectively. Any scan variable defined in Appd. G.1 may be used.
- **facet**, if defined, indicates which facet power is plotted.
- **log_freq** indicates whether a logarithmic frequency scale is used in the plot.
- **freq_start** and **freq_end** are the starting and ending point of the frequency axis for the Fourier analysis.

- **scale_lit** scales the laser light output for L-I plot. This is useful when scaling to take into account device symmetry.
- **scale_curr** scales the current in L-I plot in a laser simulation. This is useful when scaling to take into account device symmetry.
- **scale_horizontal** scales the horizontal axis for non-L-I plots.
- **scale_vertical** scales the vertical axis for non-L-I plots.
- **mode_index** is lateral mode number.

Examples

```
fourier_power input_var=current_1 output_var=laser_power &&
    freq_start=6.  freq_end=10.
```

22.321 flux_plot

flux_plot is used to plot the light energy flux density distribution within the ray-tracing device. This statement is used at the post-processing stage after ray-tracing program is run.

This statement has no parameters.

22.322 freq_control

parameter	data type	values [defaults]
data_points	intg	[601]

The statement **freq_control** is used in PICS3D to define the number of points in certain curves such as the longitudinal mode spectrum, the ASE and the noise analysis (e.g. RIN).

Parameters

- **data_points** is the number of points in spectrum

22.323 front_index

parameter	data type	values [defaults]
real_index	real	[1.]
imag_index	real	[0.]

front_index is an outside index value used as a boundary condition for optical pumping. The default value assumes air/vacuum. This command has no practical effect if **front_reflection** is used.

See the theory section in **front_reflection** for more details.

22.324 front_reflection

parameter	data type	values [defaults]
spectrum_file	char	void
power_transmission	real	[1.]
reflection_phase	real	[0.] (degrees)

front_reflection is used to specify the reflection of light power at the front of a semiconductor device. It should not be used if the coating layers are defined with **optic_coating** or are explicitly included in the layer file.

Theory

Optical pumping defined with **light_power** defines a series of cut lines across the mesh defined for the device in order to get the local index profile. For each of these 1D cuts, a plane wave transfer matrix method is used to calculate the internal reflections and the light distribution in the device. Regions with a similar index are grouped together to save computation time and interfaces are defined when there is a significant change in index values.

The very first interface along the cut lines is defined by **front_reflection**. If this is not defined, the index value from **front_index** will be used as the outside boundary condition for a Fresnel reflection. If that statement is not used either, the outside index is assumed to be air ($n_r = 1$). Coating layers defined with **optic_coating** will be included between this outside index and the first mesh point of the cut line.

Conversely, the last interface along the cut line is defined by **back_reflection**. If this is not defined, the value of **back_index** will be used as the outside boundary condition for a Fresnel reflection. If that statement is not used either, the outside index is assumed to be the same as that of the last layer so that no reflection takes place at the interface.

Parameters

- **spectrum_file** reads the power transmission coefficient from an input file. The data must be arranged in columns with the first column being the wavelength in μm and the second the transmission coefficient.
- **power_transmission** is the power transmission coefficient.
- **reflection_phase** is the phase of the field reflection coefficient. The field transmission coefficient is always assumed to be real.

Examples

```
front_reflection power_tranmission=0.99
```

The above statement is used to model an effective broadband AR coating. It should not be combined with any actual coating definitions.

22.325 full_ionization

parameter	data type	values [defaults]
mater_label	char	
dopant	char	[void], donor, acceptor, donorj (j=1..9), acceptorj (j=1..9)
mater	intg	

The statement **full_ionization** may be used to force all shallow dopants of a material to be fully ionized. By default, the incomplete ionization model based on Fermi statistics is used in all materials.

Parmaters

- **mater** is the material where the dopants are to be fully ionized. If a label has previously been defined for this material, **mater_label** may be used instead. As of the 2016 version of the software, this parameter is now optional: if no material is specified, then this command applies to all materials in the device.
- **dopant** may be used to restrict the effects of the command to one specific kind of dopant. If this parameter is omitted, all dopants are fully ionized; we may otherwise study the effects of ionizing either the donors or acceptors. If multiple species of dopants are used in the simulation, a number may be appended to the dopant name to identify the species which we want to fully ionize; all other dopants present in the simulation will be unaffected.

22.326 gain_density

parameter	data type	values [defaults]
data_file	char	
include_data	char	
conc_log_scale	char	[no], yes
wavel_range	realx2	(μm)
conc_range	realx2	[1.e23 1.e25] (m^{-3})
pn_ratio	real	[1]
av_index	real	[3.3]
temper_range	realx2	[300. 400.] (K)
data_point	intg	
temper_points	intg	[]

gain_density plots the peak material gain versus carrier density in the active region. This statement can be used in the gain preview module.

Parameters

- **data_file** specifies a file name used to save a copy of the plot data.
- **include_data** includes data files from other gain calculations on the plot for comparison purposes.
- **wavel_range** is the wavelength range used to search for the gain peak.

- **conc_range** is the electron concentration range in the well used for the plot.
- **pn_ratio** is the ratio of hole over electron concentrations.

  Note that this ratio is set to an arbitrary number in the gain preview in order to generate a 1-dimensional plot or a family of spectral curves which depend on a single parameter. In the main solver, this ratio is normally unused and the gain calculations automatically make use of the local Fermi levels for both electrons and holes.
- **av_index** is the estimated average refractive index.
- **data_point** is the number of data points in the curve.
- **conc_log_scale** determines if the carrier density points are spaced linearly or on a logarithmic scale.
- **temper_points**, if used, instructs the software to plot a family of curves at various temperatures. This parameter controls how many temperature points are used; these points will cover a temperature range defined in **temper_range**.

Examples

```
gain_density wavel_range=(1.0 1.4) &&
   conc_range=(5.e23 5.e24)  pn_ratio=1 data_point=20
```

22.327 gain_module

parameter	data type	values [defaults]
use_sheet_density	char	[no]
tilt_imref	char	[no]
apply_e_field	real	[0.]
complex_number	intg	[1.]
complex_well_number	intg	[1.]

gain_module is used to modify some conditions in a .gain preview simulation.

Parameters

- **use_sheet_density** instructs the program to use sheet density in .gain preview simulation. See also **band_distance** which defines its own sheet density setting.

- **tilt_imref** indicates whether the IMREF (inverse of the Fermi level) is to be tilted in accordance with the applied field.

- **apply_e_field** is the applied electrical field in V/m.

- **complex_number** describes which active region or complex MQW is used for the material gain calculations in the gain preview. By default, .gain uses the first active region defined in the input or in the included .mater file.

  We note that the .gain preview only shows the material gain for a single quantum-confined region. Gain curves are calculated using the energy separation and wavefunction overlap of quantum-confined states so even in the case of a complex MQW, all wells from that complex are included in the material gain curve.

  However, in a full device a MQW region may consist of several simple/complex wells, not all of which are coupled quantum-mechanically for the purposes of solving the Schrödinger equation. The material gain preview may thus differ from the modal gain in a number of significant ways.

  To view the total modal gain spectrum, it is therefore recommended to run the full simulation in the .sol file and use **gain_spectrum** in the post-processing stage.

- **complex_well_number** describes which well inside a quantum-confined MQW region is used as the reference for the carrier density vs. Fermi level relationship. As of v.2016, this is the well with the smallest bandgap by default; earlier versions of the software defaulted to the first well inside the complex.

  Why is this parameter necessary ? Basic laser theory tells us that the integral used for the material gain depends on the Fermi level separation. In the full .sol simulation, this separation is obtained at each mesh point using the transport models but for .gain, no transport is solved: this value is instead obtained by defining an input carrier density.

  Since no transport is solved in .gain, we assume a single flat Fermi level for the whole MQW but due to the band profile, even this simple model may lead to different carrier densities in each well. A reference well number is thus needed to consistently invert the carrier density vs. Fermi level relationship: we do not need to know the density in each well to define the Fermi level separation, we only need to know one.

Examples

```
gain_module apply_e_field=1.e6
```

22.328 gain_spectrum

parameter	data type	values [defaults]
data_file	char	
variable	char	[gain]
units	char	[void]
user_xlabel	char	[void]
user_ylabel	char	[void]
scale	real	[1.]
wavelength_range	realx2	(um)
asespec_latmd	intg	[1]

gain_spectrum is a post-processing statement used to plot the net modal gain spectrum at different current/voltage bias points corresponding to data sets previously defined in the **xy_data** parameter of **get_data**.

Since this original use, the command has also been re-purposed to plot additional spectral properties. In PICS3D, this also include variables from the coupled round-trip gain method (RTG) and the older analytical AC model described in Sec. 18.6.

Parameters

- **data_file** is a file name that can be used to export the curve spectrum.
- **variable** is the variable being plotted:
  - *gain* is the combined net TE/TM gain.
  - *gain_te* is the net TE gain.
  - *gain_tm* is the net TM gain.
  - *sp.rate* is the combined TE/TM spontaneous emission (maximum).
  - *sp.rate_te* is the TE spontaneous emission (maximum).
  - *sp.rate_tm* is the TM spontaneous emission (maximum).
  - *index_change* is the index change away from equilibrium.
  - *total_sp.rate* is the combined TE/TM spontaneous emission (total).

- *total_sp.rate_te* is the TE spontaneous emission (total).
- *total_sp.rate_tm* is the TM spontaneous emission (total).
- *cavity_rtg_amplitude* is the amplitude of the round-trip gain. It is similar to the round-trip gain preview data generated by the **rtgain_phase** statement except that the photon coupling and imaginary frequency/spontaneous emission contribution is included. (PICS3D)
- *cavity_rtg_phase*; as above but displays the phase information instead of the amplitude. (PICS3D)
- *rtg_spectrum* is the power spectrum (as defined in Appendix E) in the PICS3D round-trip gain method. See the appendix for application notes on the definition on side mode suppression ratio.
- *rtg_spectrum_left* is similar to *rtg_spectrum* but is weighed by the field profile to give the power spectrum on the left side of the cavity.
- *rtg_spectrum_right* is similar to *rtg_spectrum* but is weighed by the field profile to give the power spectrum on the right side of the cavity.
- *rtg_asespec* is the amplified spontaneous emission in the the PICS3D round-trip gain method.
- *rtg_asespec_left* is the amplified spontaneous emission on the left side of the cavity in the PICS3D round-trip gain method.
- *rtg_asespec_left_latmd* is the amplified spontaneous emission on the left side of the cavity in the PICS3D round-trip gain method for a given lateral mode.
- *rtg_asespec_right* is the amplified spontaneous emission on the right side of the cavity in the PICS3D round-trip gain method.
- *rtg_asespec_right_latmd* is the amplified spontaneous emission on the right side of the cavity in the PICS3D round-trip gain method for a given lateral mode.
- *rtg_signal_asespec_left* is the combined amplified spontaneous emission + signal on the left side of the cavity in the PICS3D round-trip gain method.
- *rtg_signal_asespec_right* is the combined amplified spontaneous emission + signal on the right side of the cavity in the PICS3D round-trip gain method.
- *noise_rin* is the relative intensity noise derived from the analytical AC model in PICS3D.
- *noise_fm* is the frequency noise derived from the analytical AC model in PICS3D.
- *analytic_am* and *analytic_amphase* are the magnitude and phase (respectively) of the analytical AM response (PICS3D).

– *analytic_fm* and *analytic_fmphase* are the magnitude and phase (respectively) of the analytical FM response (PICS3D).

When plotting a "maximum" value, the software will locate the position with the maximum light emission in each quantum well ("bright spot"). The spectrum is then calculated for each bright spot giving the emission per *mesh-local* unit area or volume around that spot. The spectrum is then averaged over all quantum wells or active regions before plotting.

For the "total" method, the shape of the spectrum is the same as that of the "maximum" method above. The only difference is that the spectrum is normalized according to the integrated radiative recombination rate which appears in the drift-diffusion equation. Since the rate is integrated over the *whole device*, the units of such a spectrum should be the emission per total simulation area/volume.

- **units** may be used to change the way the spectrum data is presented (to dBm/Åunits)
- **user_xlabel** and **user_ylabel** allow the user to change the labeling of the plot axes.
- **scale** can be used to artificially scale the spectrum data.
- **wavelength_range** is the plotting range for the spectrum data.
- **asespec_latmd** is the lateral mode number used when plotting *rtg_asespec_left_latmd* and *rtg_asespec_right_latmd*.

Examples

```
gain_spectrum variable=gain
```

22.329 gain_spon

parameter	data type	values [defaults]
data_file	char	
include_data	char	
wavel_range	realx2	(μm)
conc_range	realx2	[1.e23 1.e24] (m^{-3})
pn_ratio	real	[1]
av_index	real	[3.3]
data_point	intg	

gain_spon plots the peak material gain versus the spontaneous recombination current density in the active region. The recombination current is defined as the spontaneous combination rate, which is the integration of the spontaneous emission spectrum over all wavelengths, times the thickness of the active region. This statement is only used in the gain preview module.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **include_data** includes data files from other gain calculations for the purpose of comparison.
- **wavel_range** is the wavelength range within which the peak material gain is searched.
- **conc_range** is the electron concentration range in the well.
- **pn_ratio** is the ratio of hole over electron concentrations. Note that this ratio can be set to an arbitrary number in the gain preview. In the main solver, this ratio is automatically determined by the simulator according to the local Fermi levels.
- **av_index** is the estimated average refractive index.
- **data_point** is the number of data points in the curve.

Example(s)

```
gain_spon wavel_range=(1.0 1.4) &&
   conc_range=(5.e23 5.e24)  pn_ratio=1 data_point=20
```

22.330 gain_wavel

parameter	data type	values [defaults]
data_file	char	
include_data	char	
auto_pn_ratio	char	[no]
conc_log_scale	char	[no]
plot_as_abs	char	[no]
versus_energy	char	[no]
add_n_and_k_data	char	
wavel_range	realx2	(μm)
conc_range	realx2	(m^{-3})
pn_ratio	real	[1.0]
av_index	real	[3.3]
scale_gain	real	[1.]
hline background_loss	real	[0.] (1/m)
curve_number	intg	
data_point	intg	
zseg_num	intg	[1]
kp8x8_angle_points	intg	[0]

gain_wavel is a statement used in the gain preview module (.gain) to plot material gain curves at various carrier concentrations. It is also used in the initial stages of a PICS3D simulation (after **equilibrium**) to estimate the gain and provide the data required in **rtgain_phase**.

Parameters

- **data_file** is the file to which the graphic data is written in ASCII format.
- **include_data** includes data files from other gain calculations for comparison purposes.
- **auto_pn_ratio** is used only in the initial stages of a full PICS3D device simulation. It indicates if an averaged hole/electron density ratio from the full drift-diffusion solver is used to compute the gain curves. This parameter, if enabled, will override **pn_ratio**.
- **conc_log_scale** spaces the gain curves so they correspond to concentration values that are evenly spaced on a log rather than linear scale.

- **plot_as_abs** instructs the program to plot the absolute value of the gain.
- **versus_energy** plots the gain curve as a function of the photon energy rather than the wavelength.
- **add_n_and_k_data** allows the user to input a file containing complex refractive index data (n, k) and plot it alongside the computed gain curve.
- **wavel_range** is the wavelength range.
- **conc_range** is the electron concentration range. The gain curves correspond to concentration values that are evenly spaced within this range.
- **pn_ratio** is the ratio of hole over electron concentration. The gain curve depends on both the electron and hole concentrations.
- **av_index** is the estimated average refractive index and is used to compute the gain integral.
- **scale_gain** can used to artificially scale the computed gain curve.
- **background_loss** can be used to add a fixed background loss term to the computed gain curve.
- **curve_number** is the number of gain curves.
- **data_point** is the number of wavelength data points in each gain curve.
- **kp8x8_angle_points** applies to plotting optical gain based on the 8x8 k.p theory. It defines the number of angular values that are used for the in-plane wave vector direction. For each direction, the subband dispersion is calculated and the optical gain is numerically integrated. The final curve shows the averaged result.

Examples

```
gain_wavel wavel_range=(1.0 1.4) &&
   conc_range=(5.e23 5.e24)  curve_number=5 data_point=100
```

22.331 gammak_bar

parameter	data type	values [defaults]
(see) material_par		

The material statement **gammak_bar** (k=1,2,3) is an active layer macro statement used to define the Luttinger numbers in the quantum barrier.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.332 gammak_well

parameter	data type	values [defaults]
(see) material_par		

The material statement **gammak_well** (k=1,2,3) is an active layer macro statement used to define the Luttinger numbers in the quantum well.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.333 generation_rate

parameter	data type	values [defaults]
mater_label	char	void
use_light_profile	char	[yes],no
use_total_charge	char	yes,[no]
carrier	char	[both],electron,hole
rate	real	[0.] (m^{-3}/s) or (C/s)
xrange	realx2	(μm)
yrange	realx2	(μm)
mater	intg	[1]
light_number	intg	[1]

generation_rate may be used to directly define a carrier generation rate in a given material region: this may be used to model generation through heavy ion impacts

or simply to inject carriers without using a contact. Note that a similar effect can be achieved using the **import_gen_rate** parameter of **light_power**.

A more advanced model for heavy ion hits can be found in **radiation_heavy_ion**.

Parameters

- **rate** is the generation rate applied to this region. The units of this value are in m^{-3}/s unless **use_total_charge** is used. In that case, the units are C/s and the size of the affected area controls the generation rate density.
- **use_light_profile** must be used in conjunction with **light_power**. This parameter scales the **rate** value according the relative shape of the light profile. When multiple light sources are present, **light_number** identifies the profile to use.
- **xrange** and **yrange** give the area in which the generation rate occurs.
- **mater** is the material number in which the generation occurs. If a label has been defined for this material, **mater_label** may be used instead.
- **carrier** determines whether the generation will produce electron-hole pairs or only a single kind of carrier. In the vast majority of applications, the default setting should be used.

Examples

```
generation_rate mater_label=poly use_light_profile=no &&
  carrier=electron rate=5.e14 use_total_charge=yes &&
  xrange=(0.65 0.7) yrange=(0 0.1) light_number=1
```

22.334 get_active_layer

parameter	data type	values [defaults]
name	char	AlGaAs/AlGaAs, etc.
var_name1-9	char	void
var_symbol1-9	char	
var1-9	real	
mater	intg	[1]

Similar to the **load_macro** statement, **get_active_layer** is used to simultaneously assign several material parameters (e.g. bandgap, band offet, etc...) to all mesh points sharing a material number. Material properties are then evaluated based on the supplied parameters and certain reserved keywords. Note that this statement is usually automatically generated by processing the .layer file.

Note that unlike **load_macro** statement, **get_active_layer** defines parameters relating to the optical gain calculations and can also be used without defining the mesh during the gain preview (.gain file). Because gain calculations can depend on both well and barrier material parameters, the active macro may override certain parameters from passive macros of the QW and barrier regions. It is ***strongly*** recommended that users check the consistency of the passive and active macros to avoid this; when in doubt, statements such as **use_bulk_property** may be used to avoid this override.

For more information about macros, consult Appendix B and the comments in the *crosslight.mac* and *more.mac* files. See also **use_macrofile** to use custom user-defined macros.

Parameters

- **name** specifies the name of the macro to be used. The convention for active macros is to use mixed-case names such as InGaAs/AlGaAs, AlGaAs, cx-AlGaAs, InGaAsP/InP, etc... Also by convention, names without a slash refer to bulk active regions unless prefixed with "cx-" in which case they refer to a complex MQW macro (multiple layers coupled together quantum mechanically). Macro names with a slash refer to simpler QW macros (symmetric barrier/well/barrier) with the second name referring to either the barrier or substrate material.

  It is ***strongly*** recommended that users review the comments included in text of each macro to ensure they are using the right material and composition convention. Failure to do so can result in significantly de-tuned gain curves or mismatched strain.

- The parameters **var_symbol1-9** are the symbolic variable names used as function arguments in the macro. If defined, they must exactly match the symbols used in the macro function definitions. If not defined, then the simulator will assume an older macro style is being used: in this case, the order of the parameters must be the same as in the macro function definitions.

  Note that certain variables used in functions are reserved keywords and do not need to be defined in this manner.

- The parameters **var1-9** are the values of the variables appearing in the functions within the macro definition. They are most commonly used to represent

material composition in ternary/quaternary compounds.

- The parameters **var_name1-9** replaces **var1-9** when a grading of the macro parameters is used. It is used in conjunction with the **grade_active_mater** statement which describes the spatial variation of the parameters.
- The parameter **mater** is the material number being linked with this macro. This number is the same as the material number assigned to polygons in the .geo file and is inherited by the mesh.

Examples

```
get_active_layer  name=InGaAsP/InP  var1=0.467  var2=1. &&
   var3=0.202  var4=0.440  mater=3 &&
   var_symbol1=xw var_symbol2=yw var_symbol3=xb var_symbol4=yb
```

A simple active layer declaration defining the composition of both the QW and barrier

```
grade_active_mater var_name=a2 variation=function
$ rdist=relative-distance is a reserved internal variable
function(rdist)
0.71+rdist*(0.33-0.71)
end_function

get_active_layer name=cx-AlGaAs mater=   2 &&
  var_symbol1=xw &&
  var_name1=a2
```

A more complex graded well profile

22.335 get_data

parameter	data type	values [defaults]
main_input	char	
sol_inf	char	
xy_data	intgx2	
scan_data	intgx2	

The statement **get_data** is used in the post processor, *i.e.* it is used only after the main solution is complete. It is used to get the output data from the main solver.

- **main_input** is the main input file that was used to drive the main simulation.
- **sol_inf** is the main simulation output data file which contains the current data, carrier distribution data, etc. It is produced by the input file **main_input**.
- **xy_data** specifies the structural data (*i.e.*, data as a function of position x and y) to be used by graphic statements to display position dependent data (*e.g.*, by **plot_1d**). The starting data set and the ending data set numbers are defined by this statement. If one would like to plot data set 1 to 3 on the same plot, one can set **xy_data=(1,3)**.

  This parameter is used by any subsequent plot statement that displays structural data such as electron concentration distribution. Note that each data set corresponds to one **print_step** in the **scan** statement. Not all the bias points have the full structural data. Also see the **scan** statement for more detail on bias points and data sets. The bias voltage or current for each data set can be found in the message file with extension "*.sol.msg".
- **scan_data** defines the beginning and the ending data_set used by any subsequent **plot_scan** statement. For bias data they allow the user to plot the specified variable within this range of data sets.

Example(s)

```
get_data  main_input=bulk1d.sol sol_inf=bulk1d.out &&
  xy_data=(7,7) scan_data=(1,7)
```

This causes the position dependent data set number 7 to be plotted while the bias data (such as I-V data) ranging from data set 1 to set 7 are displayed.

22.336 get_raytrace_data

parameter	data type	values [defaults]
filebase	char	
xy_data	intgx2	
scan_data	intgx2	

get_raytrace_data is a post-processing statement which serves the same purpose as **get_data**. However, it is exclusively used to load ray tracing simulation results in the Optowizard program.

Parameters

- **filebase** is the root filename of the original simulation data, minus the extension (such as .sol or .out).
- **xy_data** is the range of data sets available for plotting structural data.
- **scan_data** is the range of data sets available for plotting bias-dependent scan data.

Examples

```
get_raytrace_data filebase=tip xy_data=(2 3) scan_data=(2 3)
```

22.337 global_model_setting

parameter	data type	values [defaults]
auger_qw_abrupt_prof	char	[no]
classical_conc_profile	char	[no]
range_n	real	[1.e-3] (Vt)
range_p	real	[1.e-3] (Vt)

This statement replaces **adjust_current** as of the 2016 version. It is used to control some overall current flow characteristics; it only needs to be issued when the user wants to override the default settings. There are two major models which are affected by this command.

The first model deals with the switching between the drift-diffusion (DD) current flow and thermionic emission (TIE) models. In the limit of a heterojunction with high energy barrier, a full thermionic emission model is used but when the barrier is low, there is a mixture of DD current and TIE current. This parameter affects heterojunctions in the whole device.

The second and more common application of this command is to deal with certain inconsistencies between the classical and quantum treatment of carriers in quantum wells. It is rather clean that when using a self-consistent model to compute the confined carrier states, the envelope of the confined carrier wave function should be used to account for the charge distribution. However, it is not clear how this same charge distribution is related to the classical charge distribution in the DD/TIE transport model.

If we directly use the envelope function for transport (meaning **classical_conc_profile**=*no*), that means some part of carriers under the envelop may spill over the barriers and become free carriers, just like other unbound carriers: this contradicts with our assumption that they are "confined". On the other hand, one may argue that there is good reason to convert the spill-over carriers into free carriers because of the quantum tunneling effect. After all, tunneling transport occurs when a wave function penetrates a potential barrier and overlaps with the wave function of a free-carrier state. Such an overlap will certainly happen if an applied field lowers one of the barriers to near the confined energy level. However, the numerical accuracy of such a less rigorous quantum tunneling treatment should be the subject of further studies.

The other approach is to force all confined carriers to be within the well (meaning **classical_conc_profile**=*yes*) and let TIE takes care of all the transport. This approach completely ignores any quantum tunneling effects. Another problem is the violation of current continuity as far as carrier dynamics is concerned. One can show that if the Poisson's equation and the current continuity equations take different carrier distributions (i.e., n and p are different), the gradient of the total current (carrier current plus displacement current) will not be zero, thus violating current continuity in the semiconductor device in a transient or AC simulation.

At this point, we find using the same envelope function carrier distribution in all equations yields reasonable results if tunneling model is not activated for the potential barrier involved. On the other hand, if the tunneling model is activated, use of the same envelope function carrier distribution for transport definitely produces an overestimation of the transport current. Thus we recommend the following:

- If no tunneling region is defined, use **classical_conc_profile**=*no*
- If tunneling through the QW barrier is enabled, use **classical_conc_profile**=*yes*

Parameters

- **auger_qw_abrupt_prof** may be used to force the software to use the classical concentration profile to compute the Auger recombination rate
- **classical_conc_profile** is used to indicate whether the classical concentration profile or the envelope of the confined carrier wave function should be used

for the transport.

- **range_n** (**range_p**) is the electron (hole) barrier height (normalized to thermal voltage $V_t = k_b T/q$) below which a mixture of DD and TIE takes place.

Examples

```
global_model_setting range_n=0.01 range_p=0.01
```

22.338 grating_compos

parameter	data type	values [defaults]
hi_macro_name	char	[void]
hi_active_macro	char	[void]
lo_macro_name	char	[void]
lo_active_macro	char	[void]
hi_var_symbol1-5	char	[void]
hi_avar_symbol1-5	char	[void]
lo_var_symbol1-5	char	[void]
lo_avar_symbol1-5	char	[void]
hi_mater_lib	char	[void]
lo_mater_lib	char	[void]
d_high	real	[0.1] (um)
d_fall	real	[0.] (um)
d_low	real	[0.1] (um)
d_rise	real	[0.] (um)
hi_var1-5	real	
hi_avar1-5	real	
lo_var1-5	real	
lo_avar1-5	real	
grating_order	intg	[1]
column_num	intg	[1]

---

grating_compos is used in the .layer file to generate a **grating_model** statement which, combined with the mode profile, is eventually used to define the longitudinal coupling coefficient κ in DFB/DBR lasers. Please refer to **grating_model** for help on most parameters of this statement.

The main difference between this statement and **grating_model** is that here, the index profile is defined using the material macro system and composition parameters instead of directly defining the index profile. Parameters prefixed by "lo" define the values for the low-index part of the grating while those that start with "hi" define values for the high-index part of the grating.

For both regions, both a passive an active macro are defined. Even if there is so significant loss/gain in the grating, doing so allows the software to compute the index change away from equilibrium due to injection. The parameters used to define material macro parameters (and the column number) are similar to those of **layer_mater**.

Please note that as of the 2014 version, the "library" material system of Sec. 3.5 may also be used instead of macros. This will automatically enable gain/loss calculations and result in a complex coupling coefficient.

22.339 grating_model

parameter	data type	values [defaults]
real_index_high	real	
imag_index_high	real	[0.]
real_index_low	real	
imag_index_low	real	[0.]
d_high	real	[0.1] (μm)
d_fall	real	[0.] (μm)
d_low	real	[0.1] (μm)
d_rise	real	[0.] (μm)
grating_xrange	realx2	[0. 5.] (μm)
grating_yrange	realx2	[2. 2.1] (μm)
grating_order	intg	[1]
mode_index	intg	[1]
use_active_mater	intg	[0]

grating_model is used to calculate the longitudinal coupling coefficient κ using the longitudinal index profile defined in this command and the optical field profile from the main solver.

grating_compos can also be used to define the longitudinal index profile through material macros and composition parameters rather than index values.

Parameters

- **real_index_high**, **real_index_high**, **imag_index_high**, **real_index_low**, **imag_index_low**, **d_high**, **d_fall**, **d_low** and **d_rise** are used to describe the variation of effective index along the longitudinal direction (z-direction). They are illustrated in Fig. 22.12.

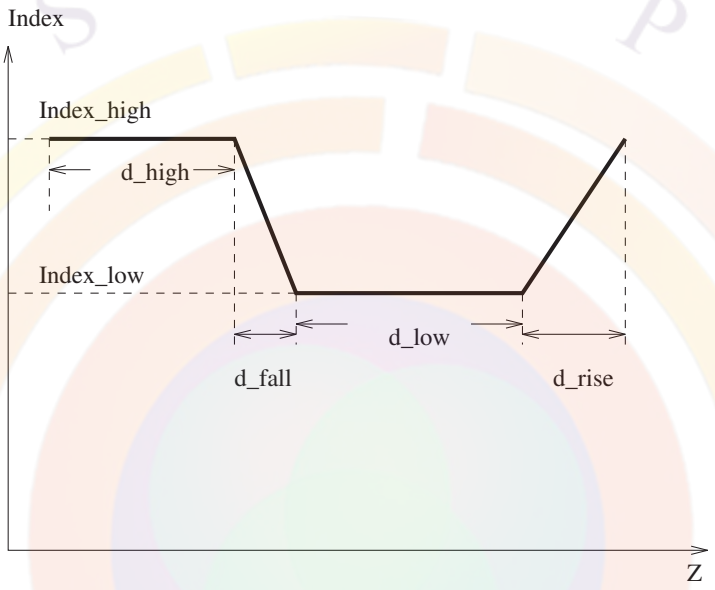


Figure 22.12: Schematics of the index variation in the longitudinal direction for the **grating_model** statement.

IMPORTANT NOTE: The distances specified in this command are only used to compute the strength of the coupling constant κ using Fourier analysis[121]. These distances should therefore be seen as the *relative* distances of the high/low index regions and *do not* determine the pitch of the grating.

The actual pitch of the grating is normally set using the **longitudinal** statement. The grating pitch can either be set directly or a reference wavelength can be used to automatically compute the pitch based on the effective index at equilibrium. In cases requiring more flexibility, the **section** statements can also be used to fix the local grating pitch in different regions of the device.

- **grating_xrange** and **grating_yrange** are used to specify the rectangular cross section on the xy-plane (lateral) of the grating layer. They can be omitted if this statement is used in the .layer file and the range can be inferred from the previous **layer_mater** declaration.

- **grating_order** is the grating order.

- **mode_index** is the lateral mode index used to calculate the coupling coefficient. Mode 1 is the fundamental mode; mode 2 is the 2nd order mode, etc..

- **use_active_mater** is used to decide if the index profile is calculated using an active macro. If zero, the index values should be specified explicitly. Otherwise, this parameter should be defined as the material number of the active region used for the calculation.

Examples

```
grating_model real_index_high=3.31 &&
  imag_index_high=-3.27 real_index_low=0.1 &&
  imag_index_low=-6.54 d_high=0.18 d_fall=0. &&
  d_low=0.12 d_rise=0. &&
  grating_xrange=(0. 1.5) &&
  grating_yrange=(1.962 2.012) &&
  grating_order=2
```

22.340 grade_active_mater

parameter	data type	values [defaults]
var_name	char	
variation	char	[linear]
grade_from	real	
grade_to	real	
grade_points	intg	[3]

grade_active_mater describes the spatial variation of a material parameter (usually composition) in a complex MQW macro. The local value of this parameter is passed as an argument to the various functions inside the macro to define material parameters.

In many ways, this statement duplicates some of the functionality common to all material parameter statements (c.f. Sec. 22.456). However, it operates at a higher level and is usually automatically generated by Crosslight helper tools like the layer.exe program.

Parameters

- **var_name** specifies the rule describing the spatial variation of a material parameter: it must be the same name as in **get_active_layer**.

- **variation** defines the shape of the spatial variation: linear, table or function. All of these options work in the same way as the macro variations in **material_par**.

- **grade_from** is the starting composition of the grading: from bottom to top for a horizontal layer and from left to right for a vertically oriented layer. **grade_to** is the matching ending composition.

- **grade_points** is the number of mesh points used to sample the grading and is mostly used in the mesh-less gain preview mode. This number must be greater than 1 and a larger number often helps with quantities such as bandgap with a non-linear composition variation. However, the program slows down significantly when a large number of grading mesh points is used.

Examples

```
grade_active_mater var_name=c1 grade_from=0.1 grade_to=0.2 grade_points=5
```

The above statement linearly grades (by default) the composition c1 from 0.1 to 0.2.

The following statement defines the variation of composition a2 using an analytical function:

```
get_active_layer name=cx-AlGaAs mater=   2 &&
  var_symbol1=xw &&
  var_name1=a2
grade_active_mater var_name=a2 variation=function
$ rdist=relative-distance is a reserved internal variable
function(rdist)
0.71+rdist*(0.33-0.71)
end_function
```

22.341 graphene_index_model

parameter	data type	values [defaults]
mater_label	char	
fermi_velocity	real	[5.e5] m/s
internal_bias	real	[1.] V
mater	intg	[1]

This statement implements a complex refractive index model[122] for a graphene layer. Please note that this model is for optical properties only and should be used with insulator macros as we do not currently model carrier transport in graphene.

The model implements the following formulas:

$$E_f(V) = hv_f\sqrt{\frac{\pi}{q}\frac{\epsilon}{d}|V - V_0|} \tag{22.17}$$

where $E_f(V)$ is the estimate of the Fermi level in the layer, v_f is the Fermi velocity and $\epsilon = 11\epsilon_0$ is the permittivity of aluminium oxide.

Using this value, the optical conductivity of the layer is given by:

$$\sigma(V,\omega) = \frac{q^2}{4\hbar} \times \frac{sinh\left(\frac{\hbar\omega}{2k_BT}\right)}{cosh\left(\frac{E_f(V)}{k_BT}\right) + cosh\left(\frac{\hbar\omega}{2k_BT}\right)} \tag{22.18}$$

and the relative complex permittivity is finally given by:

$$\epsilon_r(V,\omega) = 5.5 + i\frac{\sigma(V,\omega)}{\omega\epsilon_0 \times 3.8 \times 10^{-10}} \tag{22.19}$$

22.342 group1

group1 is not a statement but a dummy name to group all material parameter statements that use a common set of parameters. See **material_par** in section 22.456 for examples and further details.

22.343 half_mesh

This statement works the same as **double_mesh** except that it reduces the mesh density by one half.

22.344 heat_flow

parameter	data type	values [defaults]
joule	char	[yes]
opt_absorption	char	[yes]
recombination	char	[yes]
thomson	char	[yes]
peltier	char	[yes]
update_dd_eq_temper	char	[yes]
j.e_model	char	[no]
linear_source	char	[no]
thm_transient	char	[no]
jtotal_model	char	[no]
radiation_source	char	[yes]
rescale_heat_source	char	[yes]
original_joule_term	char	[no]
linear_method	char	[yes]
smooth_temperature	char	[no]
set_dd_eq_with_average	char	[no]
peltier_separate_junc	char	[yes], no
damping_step	real	[2.] (K)
var_tol	real	[1.e-5]
res_tol	real	[1.e-5]
max_temp_incr	real	[80.] (K)
set_low_temperature	char	
fit_range	real	[300.] (K)
temperature_limit	real	[800.] (K)
set_low_temperature	real	
heat_source_factor	real	[1.]
smooth_distance	real	[1.] (um)
rescale_range	realx2	[0.5 2.]
max_iter	intg	[150]
print_flag	intg	[2]
joule_mobil_flag	intg	[1]
mf2_flag	intg	[1] 0

heat_flow is used to control the solver for the self-heating (non-isothermal) model. Various heat sources are controlled by this statement. For detailed explanation of the physical origins of various heat source terms, the user is referred to Chap. 11 and Ref. [73].

Since many heating source terms rigorously derived from semiconductor theory are difficult to compute accurately on a sparse mesh, the heat sources are automatically re-scaled to match the power supplied by the DC bias. If this scaling strays too far from unity, the user is advised to check their mesh and other model parameters relating to the heat flow.

Please note temperature dependence of material parameters is implemented via functions in the macro system and the reserved **temper** keyword (the local mesh point temperature). Not all default macros implement this dependence but users may override these functions and define their own as they see fit.

Parameters

- **joule** turns on/off the Joule heating term.
- **opt_absorption** turns on / off the heating term due to optical absorption.
- **recombination** turns on/off the recombination heating source.
- **thomson** turns on/off the Thomson heat.
- **peltier** turns on/off the Peltier heat.
- **thm_transient** turns on/off the thermal transient simulation.
- **update_dd_eq_temper** turns on/off the update of the temperature used in the Drift-Diffusion equations.
- **set_dd_eq_with_average** uses the average temperature of the device (rather than the local value) in the Drift-Diffusion equations.
- **peltier_separate_junc** uses the classical concentration profile to compute the Peltier heat source, even if the self-consistent wave is used elsewhere in the model. See **global_model_setting**.
- **jtotal_model** switches the numerical model used in the Joule heating source. The standard Joule heating source term is written as $q|j_n|^2/(\mu_n n)+q|j_p|^2/(\mu_p p)$ which may be unstable in regions of carrier depletion or minority carrier region with strong diffusion because a small amount of n or p may blow up the numerical expression. This flag turns on a more stable alternative: $q|\mathbf{j_n}+\mathbf{j_p}|^2/(\mu_n n+\mu_p p)$. One can show that under the assumption of isotropic conductivity and small diffusion assumption, this expression is equivalent to the **j.e_model**.
- **original_joule_term** would instruct the simulator to use the original form of Joule term of the form: $\frac{J_n^2}{q\mu n}$ for electron current, where μ is low field mobility and n is electron concentration. Such a form as disadvantage of being numerical

unstable if electron is highly depleted. Setting this parameter to "no" would allow the use of a modified form to avoid such a problem.

- **radiation_source** is a cooling term associated with the loss of energy with emission of photons or it can be a heating term associated with the interband absorption of incident light. It is written as the photon energy times the negative of the recombination rate associated with the photon emission/absorption.
- **damping_step** (in units of degree K) is the step limit used to damp (or reduce) the correction vector in the Newton solver for the temperature distribution.
- **var_tol** (in degree K) is the variable (temperature) tolerance for the thermal equation. If the temperature error is smaller than this tolerance, the thermal equation solver is considered converged.
- **res_tol** is the relative error of the residual of the thermal equation. It is an indication how well the thermal equation is satisfied. If the res_tol is zero, the equation is perfectly satisfied.
- **max_temp_incr** is the maximum temperature increase allowed when bias is changed. If the temperature increase exceeds this limit, the program regards the solution a failure and will restart the solution procedure with a smaller bias.
- **fit_range** is used to determine the range of temperature values at which macro parameters are to be tabulated. The tabulation is further controlled by the **temperature_dep_macro_table** statement.
- **max_iter** is the maximum number of iteration used in the Newton's method.
- **print_flag** is used to flag the printing of the variable and residual errors during the Newton iteration.
- **j.e_model** is to indicate the use of an alternative expression of Joule heat in the form of $\mathbf{J} \cdot \mathbf{E}$ where **J** and **E** are the vectorial forms of current density and electrical fields. Since this term does not involve division of small numbers, it is more stable numerically. If non-convergence related to thermal simulation is encountered, one may wish to use this form of Joule heating source. For VCSEL simulation or 3D simulation involving rough mesh points, one may consider this choice. If Joule heating is the dominant heating source, this term is likely to yield reasonable results since by definition, it should equal to a simple ohmic resistor heating model.
- **linear_source** is to indicate the use of a heating source term proportional to the current. Please also see the statement **linear_heat**.

- **thm_transient** may be used to turn on the thermal transient model.
- **joule_mobil_flag** indicates which mobility model will be used in computing the Joule heating source term. If a value of 1 is used, the field dependent mobility model is used. For a value of 2, the low field mobility is used. The former is more accurate and the latter is compatible with older versions of the thermal model (version 2002 and older).
- **mf2_flag** turns on the multi-frontal linear solver for the thermal equations. Use of this flag should result in better convergence of the thermal equations and improves the speed.
- **set_low_temperature** is related to the **fit_range** parameter. These two parameters determine the lower and upper range use to tabulate temperature-dependent macros.
- **temperature_limit** is an upper limit beyond which the simulator would terminate the present **scan** statement.
- **rescale_heat_source** is used to automatically rescale the heat source to match the input power. This helps correct inaccuracies in the Joule heat term due to the sparse mesh. As of the 2012 version, this is scaling turned on by default. If this is disabled, a manual scaling of the heat sources can be defined with **heat_source_factor**.

  As of version 2017, **rescale_range** may be used to put limits on the automatic scaling factor.
- **linear_method** completely turns off the non-linear Newton solver for the heat equation: a single iteration of the underlying linear solver is used instead. This lowers the accuracy of the thermal results but often improves the convergence of the overall simulation. As of the 2012 version, this is turned on by default.
- **smooth_temperature** can be used to artificially smooth the temperature distribution with a Gaussian and eliminate hot spots which can adversely affect the simulation convergence. While the heat should diffuse naturally while solving the heat equation, this method provides a way to "fix" an insufficiently dense thermal mesh without increasing the workload on the drift-diffusion solver.
- **smooth_distance** is the smoothing distance applied when **smooth_temperature**=*yes*.

Examples

```
heat_flow opt_absorption=no damping_step=2.
```

22.345 heat_flow_simple

parameter	data type	values [defaults]
model	char	circuit, [manual]
thm_transient	char	[yes]
cir_thermal_cond	real	[10.] (3D: W/K, 2D: (W/m)/K)
cir_extern_temp	real	[300.] (K)
ref_current	real	[10.] (3D: A, 2D: A/m)
cir_device_zdim	real	[100.] (um)
ref_contact	intg	[1]

heat_flow_simple is a simplified self-heating model, recently updated for the 2015 version. For more accurate results, the full **heat_flow** model is preferred.

Parameters

- **model** is used to describe the heat source model:
  - *manual* means the heat source is defined as **ref_htsrc** $\times \left(\frac{I}{I_{ref}}\right)^2$. The reference current is defined in **ref_current**.
  - *circuit* means the heat source is defined as $I \times V - output\_power$. The DC input power is defined using the bias on **ref_contact**.
- **thm_transient** enables a transient term to the heat flow equation.
- **cir_thermal_cond** and **cir_extern_temp** are identical to the type 3 thermal boundary parameters defined in **contact**.
- **cir_device_zdim** is used to scale 2D current values to 3D as necessary.

22.346 heteroj_capture

parameter	data type	values [defaults]
all_interface	char	[no]
elec_capture	real	[1.]
hole_capture	real	[1.]
interface_mater	intgx2	

This statement is used to modify the thermal velocity in the thermionic emission model governing the current transport across heterojunction interfaces.

The following current transport model across the heterojunction is used for all abrupt heterojuntions and Schottky metal-semiconductor interfaces.

$$J_{sn} = \gamma_n \overline{v}_n^{therm}(n_s - n_{eq}), \quad (22.20)$$

$$J_{sp} = \gamma_p \overline{v}_p^{therm}(p_s - p_{eq}), \quad (22.21)$$

A quantum well may be regarded as consisting of two heterostructure interfaces as far as current transport is concerned.

From the view point of quantum capture and emission, the thermionic model is equivalent to capture/emission into/out-of the quantum well at thermal velocity which is a very fast process. The actual process may be slower since for a carrier in the barrier to loss energy to be thermalized with the confined states in the well, carrier-phonon/carrier scattering is involved. Phenomenologically, the slower energy relaxation may be represented by a less than unity coefficient multiplied to the thermal velocity. This statement is used to define such coefficients.

velocity

- **all_interface** indicates whether or not the thermionic emission model is to be modified for all heterostructure interfaces.
- **elec_capture** is the electron capture coefficient (γ_n above) multiplied to the electron thermal velocity.
- **hole_capture** is the hole capture coefficient (γ_p above) multiplied to the hole thermal velocity.
- **interface_mater** defines the material numbers (or labels) of the two materials consisting the interface.

Example(s)

```
heteroj_capture all_interface=yes
      elec_capture=0.1 hole_capture=0.1
```

This statement reduces the thermal velocities of electrons and holes for all interfaces to ten percent of the thermal veclosity.

```
heteroj_capture elec_capture=0.2 hole_capture=0.4 interface_mater=[5 6]
```

The above statement only modifies the thermal velocity for the interface between materials 5 and 6.

22.347 hole_carr_loss

hole_carr_loss is a passive macro material statement defining the dependence of the optical loss coefficient on the hole density for a given material. It introduces a loss term equal to $\alpha_n = \text{hole\_carr\_loss} \times (p - p_0)$ so that **hole_carr_loss** has units of m^2.

Note that this term is only used for passive regions as active regions have their own mechanism for carrier-dependent losses; see **passive_carr_loss** for more information.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

Examples

```
hole_carr_loss value=2.e-21 mater=1
```

22.348 hole_dos_energy

parameter	data type	values [defaults]
data_file	char	
energy_range	realx2	
data_point	intg	[50]
subband_valley	intg	[1]

hole_dos_energy plots the hole density of states (DOS) versus energy. This statement is only used in the gain preview module.

Parameters

- **data_file** is a user-specified text file containing a copy of the plot data.
- **energy_range** is the range of energy in eV.
- **data_point** is number of data points in the plot.

- **subband_valley** is the index labeling the subband valley.

Parameters

```
hole_dos_energy  energy_range=(1 1.7)
```

22.349 hole_mass

hole_mass is a passive macro material statement that defines the hole effective mass relative to the free electron mass. Note that this mass is the result of average of density of states (DOS) of light hole and heavy hole masses since only one band is used for the transport in bulk regions.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.350 hole_mobility

The statement **hole_mobility** is usually used as part of a passive material macro and provides a way for the user to define the low field mobility for holes. The user may input a specific value or use a custom function to model the doping or trap dependence.

An alternative way to define the low field mobility is to use the following statements to implement Eq. 5.42 exactly:

- **max_hole_mob** defines μ_{2p}
- **min_hole_mob** defines μ_{1p}
- **hole_ref_dens** defines N_{rp}
- **alpha_p** defines α_p

Note that using **hole_mobility** explicitly overrides this alternative method.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.351 hole_ref_dens

See **hole_mobility**.

22.352 hole_sat_vel

The material statement **hole_sat_vel** is used to define the saturation hole velocity (in m/s). It is used in Eq. 5.38 to define the field-dependent mobility function.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.353 ignore_local_current

parameter	data type	values [defaults]
mater_label	char	
mater	intg	[1]

ignore_local_current forces a material to behave like a perfect insulator: the current continuity equations will not be solved for this material and the software will simply set $J = 0$ for this region.

Parameters

- **mater** is the material number where the impact ionization is defined. If a label has previously been defined for this material, **mater_label** may be used instead.

22.354 impact_baraff

parameter	data type	values [defaults]
turn_on_zener	char	[no]
mater_label	char	
elec_phonon	real	[0.063] (eV)
hole_phonon	real	[0.063] (eV)
elec_lambda0	real	[76] (Å)
hole_lambda0	real	[58] (Å)
elec_eifactor	real	[1.5]
hole_eifactor	real	[1.5]
mater	intg	[1]

impact_baraff activates the Baraff impact ionization model described in Sec. 9.1.

Parameters

- **turn_on_zener**, if active, is the same as using the **zener** statement.
- **elec_phonon** are the **hole_phonon** are the electron and hole phonon energies (E_p).
- **elec_lambda0**and **hole_lambda0** are the zero temperature electron and hole optical phonon mean free paths (λ_0).
- **elec_eifactor**, **hole_eifactor** is the coefficient (for electrons and holes) used to calculate the ionization energy (E_I) as a function of the bandgap.
- **mater** is the material number where the impact ionization is defined. If a label has previously been defined for this material, **mater_label** may be used instead.

Examples

The following example will take the default impact ionization parameters for mater number 4:

```
impact_baraff mater=4
```

22.355 impact_chynoweth

parameter	data type	values [defaults]
turn_on_zener	char	[no]
mater_label	char	
elec_setj(j=1..3)	realx3	[7.03e7 1.231e8 1]
fld_range_nj(j=1..2)	real	[6.e7]
hole_seti(j=1..3)	realx3	[7.03e7 1.231e8 1]
fld_range_pi(j=1..2)	real	[6.e7]
elec_setnum	intg	[1]
hole_setnum	intg	[2]
optical_phonon	real	[0.063] (eV)
mater	intg	[1]

impact_chynoweth implements Selberherr's generalization of the Chynoweth [46] impact ionization model described in Sec. 9.1.

Usually, the above formula is fitted to a specific field range; multiple sets of parameters are often needed to cover the field range of interest in a particular simulation.

Parameters

- **turn_on_zener**, if active, is the same as using the **zener** statement.
- **elec_setj** is a set of 3 parameters (α_n^∞, F_{cn}, κ_n) for the jth field range. It applies to the electron impact ionization rate. **hole_setj** serves the same purpose for holes.
- **fld_range_nj** is the upper limit of the of field range $j+1$ for the electrons. If we divide the problem in N field ranges, there will be $N-1$ field values which serve as boundaries between these regions. No upper value needs to be defined when only one field range is used since it is ∞ by default. **fld_range_pj** serves the same purpose for holes.
- **optical_phonon** is the optical phonon energy; it is used to account for the temperature dependence of the parameters via the γ scaling factor described in Sec. 9.1.
- **elec_setnum** is the number of parameter sets (or field ranges) for electrons. If it is one set of fitted data available, there is no need to use any field range dividers. **hole_setnum** serves the same purpose for holes.
- **mater** is the material number where the impact ionization is defined. If a label has previously been defined for this material, **mater_label** may be used instead.

Examples

The following example will take the default impact ionization parameters for mater number 4:

```
impact_chynoweth mater=4
```

22.356 impact_dopant_dependent

parameter	data type	values [defaults]
turn_on_zener	char	[no]
model	char	baliga,[sze]
mater_label	char	
elec_a_coef	real	[1.316e8] (1/m)
baliga_cfield_factor	real	[4101.] (V/cm)
baliga_cfield_expo	real	[0.125]
hole_a_coef	real	[1.818e8] (1/m)
hole_critical_field	real	[2.036e8] (V/m)
optical_phonon	real	[0.063] (eV)
doping_range	realx2	[1.e21 1.e24] (m^{-3})
sze_cfield_factor	real	[4.e7] (V/m)
mater	intg	[1]

impact_dopant_dependent activates the empirical dopant-dependent impact ionization model described in Sec. 9.1. This is a variation of the Chynoweth model so some parameters are similar.

Parameters

- **turn_on_zener**, if active, is the same as using the **zener** statement.
- **model** activates either the Baliga or Sze models for the doping dependence of the critical field.
- **elec_a_coef** is equal to α_n^∞ in the underlying Chynoweth model. **hole_a_coef** serves the same role for holes.
- **hole_critical_field** is the critical field for holes. It is not affected by doping in this model.
- **baliga_cfield_factor** and **baliga_cfield_expo** are the prefactor and exponent used in the Baliga model for the doping dependence of the electron critical field. Please note the units which deviate from our usual convention and follow Baliga: any changes in the exponent will also likely require a change in the prefactor.
- **sze_cfield_factor** is the numerator in Sze's expression for the doping dependence of the electron critical field. It corresponds to a reference value for $N_D = 10^{16} cm^{-3}$.

- **optical_phonon** is the optical phonon energy; it is used to account for the temperature dependence of the parameters via the γ scaling factor described in Sec. 9.1.

- **doping_range** is the range of doping values where the model is used. The software will not extrapolate but will simply use the extremum values when doping is outside of this range.

- **mater** is the material number where the impact ionization is defined. If a label has previously been defined for this material, **mater_label** may be used instead.

Examples

The following example will take the default impact ionization parameters for mater number 4:

```
impact_dopant_dependent mater=4
```

22.357 impact_lackner

parameter	data type	values [defaults]
turn_on_zener	char	[no]
mater_label	char	
elec_a_coef	real	[1.316e8] (1/m)
elec_critical_field	real	[1.474e8] (V/m)
hole_a_coef	real	[1.818e8] (1/m)
hole_critical_field	real	[2.036e8] (V/m)
optical_phonon	real	[0.063] (eV)
mater	intg	[1]

---

impact_lackner activates the Lackner impact ionization model described in Sec. 9.1. This is a variation of the Chynoweth model so some parameters are similar.

Parameters

- **turn_on_zener**, if active, is the same as using the **zener** statement.

- **elec_a_coef** is equal to α_n^∞ in the underlying Chynoweth model. **hole_a_coef** serves the same role for holes.

- **elec_critical_field** is the critical field F_{cn}; **hole_critical_field** is the equivalent term for holes.

- **optical_phonon** is the optical phonon energy; it is used to account for the temperature dependence of the parameters via the γ scaling factor described in Sec. 9.1.

- **mater** is the material number where the impact ionization is defined. If a label has previously been defined for this material, **mater_label** may be used instead.

Examples

The following example will take the default impact ionization parameters for mater number 4:

```
impact_lackner mater=4
```

22.358 impact_lackner

parameter	data type	values [defaults]
turn_on_zener	char	[no]
mater_label	char	
elec_a_coef	real	[1.316e8] (1/m)
elec_lamda0	real	[1.474e8] (V/m)
hole_a_coef	real	[1.818e8] (1/m)
hole_lamda0	real	[2.036e8] (V/m)
optical_phonon	real	[0.063] (eV)
mater	intg	[1]

---

impact_mean_free_path activates the simplified mean free path impact ionization model described in Sec. 9.1. This is a variation of the Chynoweth model so some parameters are similar.

Parameters

- **turn_on_zener**, if active, is the same as using the **zener** statement.
- **elec_a_coef** is equal to α_n^∞ in the underlying Chynoweth model. **hole_a_coef** serves the same role for holes.
- **elec_lamda0** is the mean free path used to calculate the critical field F_{cn}; **hole_lamda0** is the equivalent term for holes.
- **optical_phonon** is the optical phonon energy; it is used to account for the temperature dependence of the parameters via the γ scaling factor described in Sec. 9.1.
- **mater** is the material number where the impact ionization is defined. If a label has previously been defined for this material, **mater_label** may be used instead.

Examples

The following example will take the default impact ionization parameters for mater number 4:

```
impact_mean_free_path mater=4
```

22.359 impact_model

parameter	data type	values [defaults]
use_fermi_level	char	[no]
balance_zero_bias	char	[yes]
subtract_backg_rate	char	[no]
post_process_only	char	[no]
elec_vel_factor	real	[0.7]
hole_vel_factor	real	[0.7]

The statement **impact_model** is used to control certain aspects of the impact ionization model in Crosslight such as the driving force behind the model (electric field, hot carrier gradient, etc...) and how to balance out the impact ionization rate at equilibrium so the net generation rate is zero.

Parameters

- **use_fermi_level** is used to indicate that the gradient of the Fermi level rather than the electric field is used to drive the impact ionization (i.e. $G = f(\nabla E_{fn})$). Since the Fermi level is flat at equilibrium, this guarantees a net zero generation rate but can cause convergence problems later on where the Fermi level is not smooth. A mixture of field-driven and Fermi-level-driven impact ionization can be used based on **impact_fermi_factor** in the **equilibrium** statement.

- **balance_zero_bias** is used to balance out the generation rate at equilibrium so that the net generation rate is zero. This is the default behavior starting with the 2011 version and a setting of *no* recovers the previous model. Crosslight strongly suggests that the new default model be used.

  The exact method used to accomplish this zeroing out of the net generation rate depends on the **subtract_backg_rate** setting:

  - *yes*: the background rate at equilibrium (due to the built-in potential) is calculated and subtracted from the generation rate at other bias values ($G = f(F) - f(F_0)$). This model is new to the 2012 version.
  - *no*: the background generation rate is suppressed by subtracting the built-in field at equilibrium from the driving force of the impact ionization models ($G = f(F - F_0)$).

- **post_process_only** is used to simply make the impact ionization rate available for post-processing printing without actually using it the solver. This may simplify the convergence in the avalanche region but it obviously leaves out important physics.

- **elec_vel_factor** and **hole_vel_factor** are used in the hot carrier (hydrodynamic) model to translate the carrier temperature (or energy) into an effective field which drives the impact ionization (i.e. $G = f(T_e)$). This is the default model when the hot carrier model is turned on but the actual impact ionization rate formula must still be activated by using commands such as **impact_baraff**.

  The conversion between the carrier temperature gradient and the effective field is given by:

$$\frac{3}{2}k_B(T_e - T_l) = \tau_{eng} f_{w2f} v_{sat} F \tag{22.22}$$

  which states that when a hot carrier travels with an average velocity of $f_{w2f}v_{sat}$ for a duration of τ_{eng} under the electrical field F, it gains extra energy which raises the carrier temperature T_e above the lattice temperature T_l. Here, k_B is

the Boltzman constant and τ_{eng} is the energy relaxation time specified in the hydrodynamic model.

elec_vel_factor and **hole_vel_factor** correspond to the f_{w2f} parameter above for electrons and holes, respectively. They should always be between 0 and 1 as they define the average carrier velocity as a fraction of the saturation velocity.

Examples

```
impact_model elec_vel_factor=0.7
```

22.360 impact_okuto_crowell

parameter	data type	values [defaults]
turn_on_zener	char	[no]
mater_label	char	
elec_a_coef	real	[1.316e8] (1/m)
elec_critical_field	real	[1.474e8] (V/m)
elec_a_temper_coef	real	[3.05e-4] (1/K)
elec_cfield_temper_coef	real	[6.86e-4] (1/K)
elec_a_coef_field_expo	real	[1.]
elec_a_cfield_expo	real	[2.]
hole_a_coef	real	[1.818e8] (1/m)
hole_critical_field	real	[2.036e8] (V/m)
hole_a_temper_coef	real	[3.05e-4] (1/K)
hole_cfield_temper_coef	real	[6.86e-4] (1/K)
hole_a_coef_field_expo	real	[1.]
hole_a_cfield_expo	real	[2.]
mater	intg	[1]

impact_okuto_crowell activates the Okuto-Crowell impact ionization model described in Sec. 9.1.

Parameters

- **turn_on_zener**, if active, is the same as using the **zener** statement.

- **elec_a_coef** and **elec_a_temper_coef** express the temperature dependence of the prefactor in Eq. 9.15:

$$a(T) = a_0 \left(1 + \delta_a (T - T_0)\right) \tag{22.23}$$

  A similar expression for the holes can be written using **hole_a_coef** and **hole_a_temper_coef**.

- **elec_critical_field** and **elec_cfield_temper_coef** express the temperature dependence of the critical field in Eq. 9.15:

$$F_{cn}(T) = F_{cn0} \left(1 + \delta_F (T - T_0)\right) \tag{22.24}$$

  A similar expression for the holes can be written using **hole_critical_field** and **hole_cfield_temper_coef**.

- **elec_a_coef_field_expo** is the exponent term on the field in the prefactor of Eq. 9.15. **hole_a_coef_field_expo** is the equivalent term for the holes.

- **elec_a_cfield_expo** is the exponent term applied to the critical field in Eq. 9.15. **hole_a_cfield_expo** is the equivalent term for the holes.

- **mater** is the material number where the impact ionization is defined. If a label has previously been defined for this material, **mater_label** may be used instead.

Examples

The following example will take the default impact ionization parameters for mater number 4:

```
impact_okuto_crowell mater=4
```

22.361 import_basic

parameter	data type	values [defaults]
name	char	

---

import_basic imports the underlying passive macro into a material library.

See **material_lib** and **basic_var_symbol** for further information.

Examples

```
begin_library AlGaAs
import_basic name=algaas
import_complex name=cx-AlGaAs
complex_var_symbol  var_lib=x var_complex=xw
end_library
```

This set of commands defines the "AlGaAs" library as being composed of the "algaas" passive macro and the "cx-AlGaAs" active macro. The material parameter x used when invoking the library is translated into the xw parameter of the active macros and used "as-is" in the passive macro.

22.362 import_complex

parameter	data type	values [defaults]
name	char	

---

import_complex imports the underlying passive macro into a material library.

See **material_lib** and **complex_var_symbol** for further information.

Examples

```
begin_library AlGaAs
import_basic name=algaas
import_complex name=cx-AlGaAs
complex_var_symbol  var_lib=x var_complex=xw
end_library
```

This set of commands defines the "AlGaAs" library as being composed of the "algaas" passive macro and the "cx-AlGaAs" active macro. The material parameter x used when invoking the library is translated into the xw parameter of the active macros and used "as-is" in the passive macro.

22.363 import_gain_data

parameter	data type	values [defaults]
file	char	[void]
extrap_energy	char	[yes]
extrap_conc	char	[yes]
extrap_pn_ratio	char	[no]
extrap_temper	char	[yes]
all_active_mater	char	[yes]
active_macro	char	[void]
extrap_field	char	[no]
read_auger	char	[no]
read_density	char	[no]
set_constant	real	
mater_active	intg	
active_and_embedded	intg	

import_gain_data imports the gain/index/PL data from an ASCII file generated by the **export_gain_data** statement. It can also import gain tables from another source provided they follow the same structure.

Parameters

- **file** the data file from which the gain data is to be imported. If not specified, a default file name *gain_datafile.txt* will be used.
- **all_active_mater** allows all active layers to use parameters in this statement, unless a specific material number is defined by **active_macro** or **mater_active**.
- **active_macro** indicates that this statement affects all active layers with the above active macro name.
- **extrap_energy** indicates if extrapolation is used when photon energy is outside of the range of imported data.
- **extrap_conc** indicates if extrapolation is used when concentration is outside of the range of imported data.
- **extrap_pn_ratio** indicates if extrapolation is used when hole/electron ratio is outside of the range of imported data.

- **extrap_temper** indicates if extrapolation is used when temperature is outside of the range of imported data.
- **extrap_field** indicates if extrapolation is used when field is outside of the range of imported data.
- **mater_active** if used, will use the imported gain for one particular material. Otherwise, all active material layers will use the imported data.
- **active_and_embedded** is the number of active and embedded regions sharing parameters in this statement. If not specified, all active and embedded regions will use this command.
- **set_constant** set the imported gain to fixed value rather than importing the full gain spectrum from a data file.
- **read_auger** will read computed Auger parameters from the data file. This can be used in cases where an external gain model also computes Auger recombination rates.
- **read_density** reads carrier density data from a separate file so that external calculations giving the relationship between carrier density, Fermi level, temperature and external field can be used in the simulation.

Example(s)

```
import_gain_data
```

will import data from gain_datafile.txt.

Code Sample

The following Fortran code shows the import procedure. It can be used to generate compatible tables from another source.

```
      open(61,file=impg(imp)%gainfile_imp)
      if(impg(imp)%jdense_loaded.eq.1) then
      densfile='_ds_'//trim(impg(imp)%gainfile_imp)
      open(361,file=densfile)
      endif

      read(61,'(a)') line80 ! skip a header line
! three 0/1 numbers to indicate gain/index/PL data used.
```

```
      read(61,*)  impg(imp)%ixg_use_gain
     *,impg(imp)%ixg_use_index
     *,impg(imp)%ixg_use_spon
     *,impg(imp)%ixg_use_auger

      read(61,'(a)') line80 ! skip an info. line
! range of energy (eV) and number of point in energy (uniform)
      read(61,*) ev_wvl11, ev_wvl22,npnt

      read(61,'(a)') line80 ! skip an info. line
! range of well concent. in 1/m^3 (bulk value) and number of points (uniform)
      read(61,*) an1,an2,ncur


      read(61,'(a)') line80 ! skip an info. line
! range of hole/elec conc. in well and number of points.
! this can be sent to 1. 1. 1 if you always assum hole=electron
      read(61,*) impg(imp)%gxi_pn_ratio_range(1)
     *  ,impg(imp)%gxi_pn_ratio_range(2),impg(imp)%num_pn_ratio_points

      read(61,'(a)') line80 ! skip an info. line
! range of field and number of points (uniform)
! if only one field is available, it is OK.  for example:
!  1.e7 1.e7 (V/m) 1   is OK.
      read(61,*) impg(imp)%gxi_field_range(1)
     *  ,impg(imp)%gxi_field_range(2),impg(imp)%num_field_points

      if(impg(imp)%num_field_points.gt.1) then
             impg_field_dep=1
      endif

      read(61,'(a)') line80 ! skip an info. line
! range of temperature and number of points (uniform)
! if only one temperature is available, it is OK.  for example:
!  300. 300. 1   is OK.
      read(61,*) impg(imp)%gxi_temper_range(1)
     *  ,impg(imp)%gxi_temper_range(2),impg(imp)%num_temper_points



      allocate(impg(imp)%gxi_ev1d(npnt))
      allocate(impg(imp)%gxi_conc1d(ncur))
      allocate(impg(imp)%gxi_pn_ratio1d(
```

```
*  impg(imp)%num_pn_ratio_points))
 allocate(impg(imp)%gxi_field(impg(imp)%num_field_points))
 allocate(impg(imp)%gxi_temper1d(impg(imp)%num_temper_points))
 if(impg(imp)%ixg_use_gain.eq.1) then

 allocate(impg(imp)%gxi_gain(npnt,ncur
*,impg(imp)%num_pn_ratio_points
*,impg(imp)%num_field_points
*,impg(imp)%num_temper_points))

 endif

 if(jread_dense.gt.0) then
 allocate(impg(imp)%gxi_elec(
* impg(imp)%num_dense_points
*,impg(imp)%num_field_points
*,impg(imp)%num_temper_points))
 allocate(impg(imp)%gxi_hole(
* impg(imp)%num_dense_points
*,impg(imp)%num_field_points
*,impg(imp)%num_temper_points))
 endif

 if(impg(imp)%ixg_use_spon.eq.1) then
 allocate(impg(imp)%gxi_spon(npnt,ncur
*,impg(imp)%num_pn_ratio_points
*,impg(imp)%num_field_points
*,impg(imp)%num_temper_points))
 allocate(impg(imp)%gxi_spint(ncur
*,impg(imp)%num_pn_ratio_points
*,impg(imp)%num_field_points
*,impg(imp)%num_temper_points))
 endif

 if(impg(imp)%ixg_use_index.eq.1) then

 allocate(impg(imp)%gxi_index(npnt,ncur
*,impg(imp)%num_pn_ratio_points
*,impg(imp)%num_field_points
*,impg(imp)%num_temper_points))

 endif
```

```
 if(impg(imp)%ixg_use_auger.eq.1) then

 allocate(impg(imp)%gxi_auger(ncur
*,impg(imp)%num_pn_ratio_points
*,impg(imp)%num_field_points
*,impg(imp)%num_temper_points))

 endif

 impg(imp)%ncur_ixg=ncur
 impg(imp)%npnt_ixg=npnt

 do ix=1,npnt
   impg(imp)%gxi_ev1d(ix)=f_incr(ix,ev_wvl11,ev_wvl22,npnt)
 enddo
 do ix=1,ncur
   impg(imp)%gxi_conc1d(ix)=f_incr(ix,an1,an2,ncur)
 enddo
 do ix=1,impg(imp)%num_pn_ratio_points
 impg(imp)%gxi_pn_ratio1d(ix)=f_incr(ix
*,impg(imp)%gxi_pn_ratio_range(1)
*,impg(imp)%gxi_pn_ratio_range(2)
*,impg(imp)%num_pn_ratio_points)
 enddo
 do ix=1,impg(imp)%num_field_points
 impg(imp)%gxi_field(ix)=f_incr(ix
*,impg(imp)%gxi_field_range(1)
*,impg(imp)%gxi_field_range(2)
*,impg(imp)%num_field_points)
 enddo
 do ix=1,impg(imp)%num_temper_points
   impg(imp)%gxi_temper1d(ix)=f_incr(ix
*  ,impg(imp)%gxi_temper_range(1)
*  ,impg(imp)%gxi_temper_range(2),impg(imp)%num_temper_points)
 enddo


 numprn=0
 if(impg(imp)%ixg_use_gain.eq.1) then
   numprn=numprn+1
 endif
 if(impg(imp)%ixg_use_index.eq.1) then
   numprn=numprn+1
```

```
      endif
      if(impg(imp)%ixg_use_spon.eq.1) then
        numprn=numprn+1
      endif
      if(impg(imp)%ixg_use_auger.eq.1) then
        numprn=numprn+1
      endif

! starting read in the data
      read(61,'(a)') line80  ! skip an info line

      do ix5=1,impg(imp)%num_temper_points ! number of temperature points
      do ix4=1,impg(imp)%num_field_points  ! number of field points
      do ix3=1,impg(imp)%num_pn_ratio_points  ! number of p/n ratio points
      do ix2=1,ncur  ! number of conc. points
      do ix1=1,npnt  ! number of energy points


! 1st column is list of energy (eV),
! 2nd column is gain in 1/m^3
! 3rd column is change of real refractive index with respect to
!     equilibrium concentration, or index(n)-index(n0) where
!     n0=equilibrium well concentration.
! 3rd column is PL data in 1/m^3/s/eV (also in bulk value)

!      read(61,'(4e21.12e3)') xdum,(tmpprn(kk),kk=1,numprn)
      read(61,*) xdum,(tmpprn(kk),kk=1,numprn)

      numprn=0
      if(impg(imp)%ixg_use_gain.eq.1) then
       numprn=numprn+1
        impg(imp)%gxi_gain(ix1,ix2,ix3,ix4,ix5)=tmpprn(numprn)
      endif
      if(impg(imp)%ixg_use_index.eq.1) then
        numprn=numprn+1
        impg(imp)%gxi_index(ix1,ix2,ix3,ix4,ix5)=tmpprn(numprn)
      endif
      if(impg(imp)%ixg_use_spon.eq.1) then
        numprn=numprn+1
        impg(imp)%gxi_spon(ix1,ix2,ix3,ix4,ix5)=tmpprn(numprn)
      endif
      if(impg(imp)%ixg_use_auger.eq.1) then
        numprn=numprn+1
```

```
        impg(imp)%gxi_auger(ix2,ix3,ix4,ix5)=tmpprn(numprn)
      endif



      enddo
! there must be a blank line here so that the file is GNUPLOT friendly.
! for example, you can plot the gain in GNUPLOT using command:
! plot "mygainfile.txt" using 1:2
! for index change:
! plot "mygainfile.txt" using 1:3
! etc.
      read(61,'(a)',end=8126) line80


! integral
      if(impg(imp)%ixg_use_spon.eq.1) then
        impg(imp)%gxi_spint(ix2,ix3,ix4,ix5)=
     *    f_integr(impg(imp)%gxi_ev1d
     *    ,impg(imp)%gxi_spon(1,ix2,ix3,ix4,ix5),npnt)
      endif


      enddo
      enddo
      enddo
      enddo

8126  continue


      if(impg(imp)%jdense_loaded.eq.1) then
      do ix5=1,impg(imp)%num_temper_points
      do ix4=1,impg(imp)%num_field_points

      do ix1=1,impg(imp)%num_dense_points
      read(361,'(5e21.12e3)')
     *  impg(imp)%gxi_qfln(ix1),impg(imp)%gxi_elec(ix1,ix4,ix5)
      enddo
      read(361,'(a)') line80

      do ix1=1,impg(imp)%num_dense_points
      read(361,'(5e21.12e3)')
```

```
*  impg(imp)%gxi_qflp(ix1),impg(imp)%gxi_hole(ix1,ix4,ix5)
 enddo
 read(361,'(a)') line80

 enddo
 enddo
 close(361)
 endif ! jdense_loaded

 close(61)
```

22.364 import_kp_data

import_kp_data is the reverse of **export_kp_data**. For other parameters, please see **import_gain_data**.

22.365 import_fdtd_data

parameter	data type	values [defaults]
data_file	char	

The command **import_fdtd_data** is used to import FDTD result from a data file. Since FDTD simulations can be quite lengthy, it can be useful to reimport an existing simulation and concentrate on the APSYS electrical modeling only.

Also, single wavelength calculation(e.g. IQE spectrum calculation) can be done by using this command: the FDTD transfer function is reused but the optical pumping is monochromatic.

Parameters

- **data_file** is a name of FDTD data file to be imported. If not specified, a file name "meep_density.dat" will be used as default.

Examples

```
import_fdtd_data data_file=meep_density.dat
```

22.366 import_qcl_gain_para

parameter	data type	values [defaults]
directory	char	
file	char	
escape_fraction	real	[0.5]
total_active_qw_thick	real	[0.016] (um)
adjust_resonant_field	real	[0.] (V/m)
sp.rate_2d_integral	real	[1.e19] (1/m/s)

import_qcl_gain_para is used during the full simulation of a QCL laser. When using this approach, the first step is to do a microscopic gain calculations using the gain preview mode (.gain file) and a single period of the QCL superlattice. These results are then imported in the larger macroscopic project (.sol) using this statement; the larger projects includes all the QCL layers and turns on the full Drift-Diffusion model and other important effects.

This statement requires the use of **qc_laser_preview** to generate the QCL properties that will be imported. The full simulation done in the macroscopic project is similar to the LI curve generated by this preview statement but is more accurate.

Parameters

- **directory** is the location of the microscopic project which computes the optical gain.
- **file** is the file containing the QCL gain parameters.
- **escape_fraction** is the fraction of the current which escapes the QW region; this determines the overflow leakage.
- **total_active_qw_thick** is the total thickness of the QWs belonging to the active region in one period of the QCL superlattice.
- **adjust_resonant_field** can be used to manually shift the broadening curve for the resonant tunneling. This can account for inaccurate values of the field in the microscopic gain project.
- **sp.rate_2d_integral** is the total integrated spontaneous emission rate ($\int r_{sp}(E)$). Unlike typical active regions, this value is not computed alongside the optical gain in the QCL model.

Examples

```
import_qcl_gain_para directory=micro file=qcl_para.txt &&
  escape_fraction=0.0002 total_active_qw_thick=0.0163 &&
  adjust_resonant_field=0.1e7
```

22.367 import_raytrace

parameter	data type	values [defaults]
device_type	char	[led], pd
mater_label	char	
mater	intg	[1]
data_set	intg	[1]

import_raytrace is used to import the post-processing ray tracing results of a previous simulation into a new device simulation. This provides increased self-consistency by importing the photon density into the simulation.

Parameters

- **device_type** is the type of device, *led* for a light emitting diode or *pd* for a photodetector or solar cell.
- **mater** is the material number into which the imported photon density is introduced. If a label has previously been defined for this material, **mater_label** may be used instead.
- **data_set** selects which photon density profile is loaded into the simulation. Since the original data is only available at a few select data sets at which the post-processing ray tracing was previously performed.

Examples

```
import_raytrace mater=3 data_set=6
```

22.368 include

parameter	data type	values [defaults]
file	char	[void]
ignorej (j=1..9)	char	[void]

include is used to include another file as part of the present input file. It is similar to

```
#include
```

Parameters

- **file** is the name of the file being included.
- **ignorej (j=1..9)** are statements in the included file that are to be ignored and not included in the merged input file.

Examples

```
include file=laser1.doping
```

22.369 independent_mqw

independent_mqw is used in .layer file to override the default behavior and assign different material numbers for active layers with the same composition. The default simplification is done to speed up the calculations since it allows the Schrödinger solver to be called only once for each material. Using **independent_mqw** will thus increase the computation time.

Overriding the default behavior is nevertheless recommended for self-consistent quantum well calculations. In this case, the shape of the confining potential also depends on the local field for each well.

22.370 independent_zdir_mqw

parameter	data type	values [defaults]
zseg_range	intgx2	[1 99999]

independent_zdir_mqw is equivalent to the **independent_mqw** statement. However, it applies to quantum wells defined by stacking mesh planes in the z-direction rather than those defined in the .layer file.

Parameters

seg_range can be used to restrict the range of application of this command.

Examples

```
independent_zdir_mqw  zseg_range=(4 6)
```

22.371 index_field_dependence

parameter	data type	values [defaults]
mater_label	char	
linear_term	real	[0.](m/V)
square_term	real	[0.](m^2/V^2)
mater	intg	[1]

index_field_dependence is used to describe a field-dependent refractive index.

Parameters

- **linear_term** is the linear term of the field dependence. It is related to the Pockels constant.
- **square_term** is the square term of the field dependence. It is related to the Kerr effect.
- **mater** is the number assigned to the material affected by this statement. If a label has previous been defined as an alias, **mater_label** may be used instead.

Examples

```
index_field_dependence linear_term=1.e-12 square_term=1.e-19 mater=2
```

22.372 index_model

parameter	data type	values [defaults]
sample_mesh	char	[no]yes
use_l.w.e.factor	char	[no]yes
free-carrier	char	[yes] no
interband	char	[yes] no
linear_active	char	[no] yes
linear_passive	char	[no] yes
free_carr_passive	char	yes [no]
force_update	char	yes [no]
eg_fac1	real	[10.]
eg_fac2	real	[2.]
l.w.e.factor	real	[1.0]
an_active	real	[0.0] (m^3)
bn_active	real	[0.0] (m^3/K)
ap_active	real	[0.0] (m^3)
bp_active	real	[0.0] (m^3/K)
an_passive	real	[0.0] (m^3)
bn_passive	real	[0.0] (m^3/K)
ap_passive	real	[0.0] (m^3)
bp_passive	real	[0.0] (m^3/K)
p_expo	real	[1.0]
sample_point	intg	[200]

index_model is used to calculate the index change with respect to equilibrium. In effect, the actual index of a material during the simulation is given by the sum of the value defined by **real_index** and this index change.

If this statement is not used, the default settings shown above will be used to turn on most important index change models. This statement should be used to alter this default selection; for example, to enable free-carrier index change in passive regions.

There are various independent contributions to the index change; these models are outlined in Sec. 8.4. Some terms can be turned on/off for passive and/or active regions according to the parameters of this statement.

To enable models which are for active regions only, it is possible to simply modify the input file so that a region which does not contribute significant gain is still treated as a bulk active region. This causes the program to perform additional work which is not useful in terms of the gain but as a result, the index change terms will become available.

- **sample_mesh** is used to compute the index change due to interband transitions (Kramers-Kronig). If turned on, the index change is computed for all mesh points. Otherwise only one calculation is used, based on the average carrier density in the active region.

- **use_l.w.e.factor** indicates whether or not the user-defined line with enhancement factor (given by the **l.w.e.factor** parameter) is used to compute the index change.

- **free_carrier** and **free_carrier_passive** indicate whether index change induced by free carrier absorption is included for the active and passive layers, respectively.

- **interband** turns on the index change due to interband transitions (Kramers-Kronig).

- **linear_active** and **linear_passive** turn on the linear index change model for active and passive layers, respectively.

  Note that the linear model is not exactly linear as it supports exponent control on the hole density. The formula is more properly written as:

$$a_n * (n - n_0) + b_n * (p - p_0)^{p_{expo}} \tag{22.25}$$

  where $n, n0, p, p0$ are the electron and hole carrier densities under bias and at equilibrium. The a and b coefficients can be defined separately for active and passive layers.

- **force_update** is used to force the update of the index change with bias. To save on computation time, by default only laser simulators like LASTIP and PICS3D will compute the index change. To force APSYS to have the same behavior, turn on this option.

- **eg_fac1** and **eg_fac2** are used to define the energy bounds of the Kramers-Kronig integral. These upper and lower limits (E_{01}, E_{02}) are defined by:

$$E_{01} = MAX[MIN(E\_g, E) - eg\_fac1 E_{scat}, 0] \tag{22.26}$$
$$E_{02} = eg\_fac2 E_g \tag{22.27}$$
$$E_{scat} = \hbar/\tau_{scat} \tag{22.28}$$

  where the E_g is the bandgap and E is the photon energy of interest.

- **sample_point** is the number of sample points in spectrum used to evaluate the integrand in the K-K relation. The size of this parameter directly influences the speed of the integration and therefore the speed of the simulator. Please note that the K-K relation used to compute the index change due to carrier injection into the active region involves an integral over the energy range and is rather time consuming. Turning on this model slows down the simulator substantially.

Examples

```
index_model use_l.w.e.factor=yes l.w.e.factor=2
```

22.373 index_spectrum

parameter	data type	values [defaults]
spectrum_file	char	[void]
mater_label	char	[void]
mater	intg	[1]

index_spectrum is used to assign experimental complex refractive index data (n, k) to a material. This is used in combination with **light_power** to obtain wavelength-dependent optical generation rate. The primary application of this statement is solar cells.

Parameters

- **spectrum_file** is the file name containing the refractive index data. The first number on each line must be the wavelength in μm, while the second and the third numbers must be the corresponding real and imaginary indices.
- **mater** is the material number to which the index data is assigned. If the material has previous been assigned a label, **mater_label** may be used instead.

Examples

```
index_spectrum spectrum_file=solar.silicon mater=1
```

The first few lines of the file referenced above are shown below:

```
0.250000E+00  0.137933E+01  0.366056E+01
0.260000E+00  0.172880E+01  0.407596E+01
0.270000E+00  0.207826E+01  0.468393E+01
0.280000E+00  0.295235E+01  0.525848E+01
0.290000E+00  0.416276E+01  0.516935E+01
0.300000E+00  0.452816E+01  0.413007E+01
0.310000E+00  0.489356E+01  0.355234E+01
0.320000E+00  0.495612E+01  0.325949E+01
0.330000E+00  0.502415E+01  0.307249E+01
```

To avoid any issues with data extrapolation, the user is strongly encouraged to provide index data for the entire wavelength range used in the simulation.

22.374 index_wavel

index_wavel is the same as **gain_wavel** except it plots the index change.

22.375 init_wave

parameter	data type	values [defaults]
steady_state_photon	char	[no]
sidemode_enh_model	char	[no]
power_dep_facet	char	[],left,right,both
photon_bal_model	char	[no]
lateral_mode_bal_model	char	[no]
frozen_gain_thm_model	char	[no]
point_ll	realx2	(μm)
point_ur	realx2	(μm)
fld_center	realx2	(μm)
init_wavel	real	(μm)
spon_mode	real	[1.e-3]
length	real	[300.0] (μm)
backg_loss	real	[0.0] (1/m)
wavel_range	realx2	[0.5 1.8] (μm)
mirror_ref	real	[0.3]
prn.gain_range	real	[-9999. -9999.] (μm)
gain.sat	real	[0](m^3)
photon_fac	real	[1.e7]
front_back	realx2	[-9999. -9999.]
delta_index	real	[0.3]
n_ubdata	realxn	
n_lbdata	realxn	
rtg_omega_scale	real	[1.e13]
power_dep_coef1	real	[0.] (W^{-1})
power_dep_coef2	real	[0.] (W^{-2})
power_dep_coef3	real	[0.] (W^{-3})
ase_mode_deg_factor	real	[1.]
boundary_type	intgx4	[1 1 1 1]
ubdata_num	intg	[0]
lbdata_num	intg	[0]
prn.gain_num	intg	[50]

The statement **init_wave** initializes several parameters needed to solve the wave equation. Note that geometric dependent parameters such as **point_ll** and **point_ur** can be overridden by corresponding parameters in the **wave_boundary** statement.

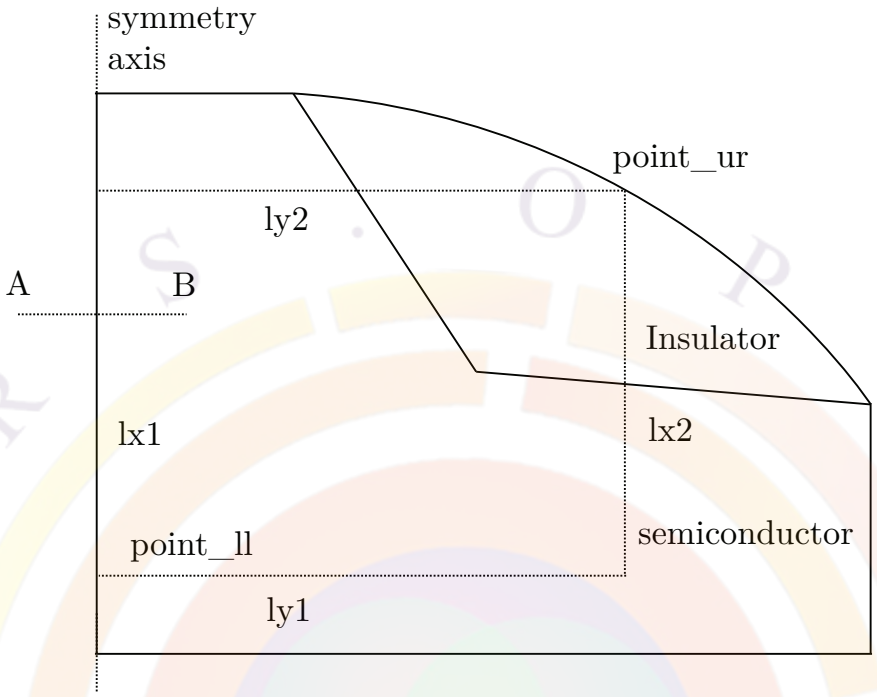


Figure 22.13: Schematics of the boundary condition for the wave equation.

Parameters

- **steady_state_photon** forces the photon density to stay fixed in transient simulations ($\frac{dS}{dt} = 0$).

- **point_ll** and **point_ur** are lower-left and upper-right points of the window within which the wave equation is solved (see also Fig. 22.13). This window must be less than or equal to the full device size defined in the .geo input file.

  This window must be reduced in certain cases to prevent the mode solver from finding certain non-lasing optical modes. For example, many designs (such as gain-guided devices) support modes with a high effective index but with poor overlap with the active region. These modes will be picked ahead of the actual lasing mode by the solver since they are sorted in order of decreasing effective index.

  The user should carefully set the window size in these structures to capture the right lasing modes without perturbing their boundary conditions too much.

- **fld_center** is the estimated optical mode center. Note that this information is only used to initialize the wave function and may or may not coincide with the optical field center in the final solution.

- **init_wavel** is the initial (or guess) emission wavelength corresponding to the peak modal gain. After finding the wave distribution using this wavelength, the software calculates the modal gain spectrum within **wavel_range** and determines the peak gain wavelength. Therefore, **init_wavel** is used only once.

- **length** is the optical cavity length. This is needed to compute the mirror loss or emission loss in LASTIP. This term is not used in PICS3D and the **section** statements are used instead.

- **backg_loss** is the minimum background loss coefficient assigned by default to all regions outside the active region. When the **absorption** statement is used in a material macro or as an override, the larger of the two values will take precedence. By default, most Crosslight macros are set up to define zero absorption so that **backg_loss** is used for all the cladding layers of a laser.

  Additional carrier-dependent loss terms can also be added by the appropriate statements: this is often important in thermal simulations.

- **wavel_range** is the search range used to determine the peak lasing gain in LASTIP.

- **mirror_ref** is the mirror reflectivity needed to calculate the mirror loss at a given cavity length in LASTIP. When using asymmetric mirrors, **front_back** may be used instead.

  When LASTIP is used to model a DFB laser, **mirror_ref** should be set to an effective value that will approximate the grating loss. However, it is preferable to use PICS3D to model these devices.

  In PICS3D, this parameter is not used. Mirror boundary conditions are set using the **longitudinal** statement.

- **prn.gain_range** is the wavelength range within which the gain spectral data are computed and printed at each set of structural data. The default settings force this parameter to be the same as **wavel_range**.

- **gain.sat** is the gain saturation (or gain suppression) coefficient ε given by the following two-level system formula:

$$g = \frac{g_0}{1 + \varepsilon \sum_i S_i} \tag{22.29}$$

  where g and g_0 are the optical gain with and without gain saturation, respectively, and S_i is the photon density of mode i.

  Cross-saturation between optical modes is only supported in PICS3D through the **sidemode_enh_model** setting. This enables a special gain saturation formula that mimics the effects of the standing wave pattern: this is omitted in

the normal coupled-wave formalism currently used for edge-emitting devices. When the model is turned on, the gain saturation for mode j is expressed as:

$$g_j = g_0 \frac{1 + \varepsilon \sum_{i \neq j} S_i}{1 + \varepsilon \sum_i S_i} \tag{22.30}$$

With this model turned on, the mode which carries the most photons (i.e. main mode) is subject to the gain suppression to a larger extent than the side modes. This is expected to degrade the side mode suppression ratio as the bias increases which may help to explain mode hopping behavior in certain devices.

- **photon_fac** is the factor used to divide the photon number in the rate equation so that the order of magnitude is the same as for the other equations. In PICS3D, the round-trip gain equation is also affected by this parameter.

  If the device is very large, the photon number in the laser cavity may also be very large. In that case, the residue of even a well-converged rate equation may show a large error in output log (5th column in printout). which can prevent the solver from converging. Increasing this value gives less weight to the photon rate equation in the Newton solver and can help fix these convergence issues at the cost of less precision in the solution for the photon number.

- **spon_mode** is the fraction of the isotropic spontaneous emission that couples into the lasing mode inside the laser cavity. It appears in the photon rate equation and determines how much spontaneous emission couples into the lasing mode. The coefficient β is often used in textbooks to represent this fraction.

- **front_back** is used to specify the front and the back facet power reflectivity. This will override the value in **mirror_ref**. If only **mirror_ref** is used, both the front and back facets are assumed to be the same.

- **n_ubdata** and **n_lbdata** are window specifications used to override the default rectangular window defined by **point_ll** and **point_ur**. The format is (x1 y1 x2 y2 .... xn yn) where x and y are point coordinates and n is the total number of points in the boundary. The value defined in **ubdata_num** and **lbdata_num** must be equal to 2*n.

- **rtg_omega_scale** is used to scale the laser frequency variable in the coupled-RTG method. The laser frequency is converted into a unitless number of the order of unity via this parameter.

- **delta_index** is used to describe a boundary where the wave function decays exponentially by a function:

$$W(x) = W_0 exp(-k_0 * delta\_index * x)$$

where k_0 is the wave number and **delta_index** is estimated from the effective index (which may be unknown before the simulation):

$$delta\_index^2 = effective\_index^2 - local\_index^2$$

- **boundary_type** is the boundary types on the sides of the window. The boundary type is represented by four integers in the order (lx1,lx2,ly1,ly2). The corresponding boundary is indicated in Fig. 22.13. The integers can only take on values of 1, 2, 3 or 5:
  - Type 1 is a Dirichlet boundary condition where the wave intensity must be zero at the boundary. This type is used when the boundary is far enough away from the optical mode. In the case where the boundary is a symmetry axis (lx1 in Fig. 22.13), a zero intensity boundary condition produces an optical mode with odd symmetry.
  - Type 2 is a Neumann boundary condition where the slope of the wave function must be zero at the boundary. This boundary produces modes with even symmetry so the wave function at point A is the same as at point B in Fig. 22.13.

    In a one-dimensional simulation with semiconductor layers parallel to the x-direction, the optical mode has translational symmetry in the x-direction. Therefore, both lx1 and lx2 boundaries should be type 2 while ly1 and ly2 should be type 1.
  - Type 3 is used to represent a truncated wave function that decays exponentially. This type of boundary may be useful in cases where the user wishes to save some mesh points by cutting a large portion of the device area out of the wave solution window. In such case, we may assume the wave decays exponentially at the boundary. This choice works together with the parameter **delta_index**.
  - Type 5 represents a boundary of perfectly matched layer (PML). Refer to Sec. 12.6 for details.

- **ubdata_num** and **lbdata_num** specify the number of variables used in **n_ubdata** and **n_lbdata**. It must always be equal to twice the number of points in the matching boundary declaration.

- **prn.gain_num** is the number of wavelength data points used to compute the gain and spontaneous emission spectra at each data set.

- **power_dep_facet** turns on a power-dependent mirror reflectivity model. The mirror reflectivity is adjusted by $\Delta R = aP + bP^2 + cP^3$ where the a, b, c coefficients are given by **power_dep_coefi,i=1..3**.

- **photon_bal_model** enables an alternate model for the photon rate equation PICS3D; this makes the model closer to LASTIP and guarantees energy conservation.

  By default, PICS3D uses the cavity round-trip gain and solves a complex equation for unity gain and phase matching: $RTG = 1 + 0i$. When the photon balance model is turned on, the wavelength of the mode is taken from solution of the imaginary part of the RTG; the magnitude of the RTG is then used to compute the effective mirror loss and the photon rate equation compares this loss term to the integrated stimulated recombination and internal loss rates.

  As of version 2017 of the software, this model should be considered experimental.

- **lateral_mode_bal_model** serves a similar function to the *ignore_phase* parameter of the **longitudinal** statement but only applies when **photon_bal_model**=*yes*. With this model enabled, the wavelength of the mode is taken from the peak gain instead of the solution of the RTG equation.

  As of version 2017 of the software, this model should also be considered experimental.

- **frozen_gain_thm_model** is a model intended to improve convergence of high-power lasers in the thermal roll-over region. When this model is enabled, the solver will disable the solution of the full drift-diffusion equation at high temperatures if it detects that the convergence is stalling. When those equations are disabled using this model, the software will switch to using a simplified linear equation to model all layers as simple ohmic resistors.

- **ase_mode_deg_factor** is used to scale the photon density of states for the amplified spontaneous emission (ASE) propagation model in PICS3D. This model is mostly used for superluminescent diodes and optical amplifiers.

  For cases where only half the structure is modeled, this value should be set of $\frac{1}{2}$ to correctly count $D(E)$; if the full width of the device is modeled, this value should be set to 1. A default of value of 1 is used for pre-v2017 backwards compatibility purposes.

Examples

For a 1-D structure:

```
init_wave point_ll=(0., 0.0) point_ur=(1.5, 3.08) &&
  fld_center=(0.5, 1.5) length=300 backg_loss=1000. &&
  boundary_type=(2 2 1 1 )  init_wavel=1.3 mirror_ref=0.32 &&
  wavel_range=(1.2, 1.4)
```

```
For a 2-D structure with lx1 boundary as the symmetry axis:

init_wave point_ll=(0., 0.0) point_ur=(3.5, 3.0) &&
  fld_center=(0.5, 1.5) length=300 backg_loss=1000. &&
  boundary_type=(2 1 1 1 )  init_wavel=1.3 mirror_ref=0.32 &&
  wavel_range=(1.2, 1.4)

For a 2-D structure without any symmetry axis:

init_wave point_ll=(0., 0.0) point_ur=(7.0, 3.0) &&
  fld_center=(3.5, 1.5) length=300 backg_loss=1000. &&
  boundary_type=(1 1 1 1 )  init_wavel=1.3 mirror_ref=0.32 &&
  wavel_range=(1.2, 1.4)

For a generic boundary with 4 corner points:

init_wave ... &&
  ubdata_num=8 8_ubdata=(0. 3.16 0.773 3.16 1.5 2.16 6. 2.16)
```

22.376 inner_bar_gain

The statement **inner_bar_gain** is used to specify that inner barriers of a complex MQW region are to be treated like any other quantum-coupled region.

This statement was originally created for historical reasons: the original design of our MQW model assumed that barrier and well layers alternated in the following pattern: b/w/b/w/.../b. As such, the optical gain was only calculated in the even-numbered parts of the complex region which corresponded to the quantum wells.

When this statement is turned on, the pattern may be regarded as being b/w/w/w/..../w/b: the exact shape, thickness and profile of the outer barriers is still ignored by the Schrödinger solver as those regions only serve as boundary conditions for the solver. See Sec. 8.2 and Fig. 8.4 for more details.

Please note that this statement is obsolete when using the "library" (Sec. 3.5) method of declaring quantum regions: the equivalent of **inner_bar_gain** is automatically assumed when declaring a quantum region in this manner.

This statement has no parameters.

22.377 insert_mesh_order

parameter	data type	values [defaults]
layer_index	intg	[1]

insert_mesh_order is used by the geo3d mesh connection tool to decide the order of connection for the inserted mesh planes, respective to the order to the mask layers in the GDS files.

This statement is usually created automatically by the MaskEditor GUI program and loaded into the main simulation via a .zst file.

Parameters

- **layer_index** is the GDS mask layer number at which additional mesh planes are added.

Examples

```
insert_mesh_order layer_index=2
```

22.378 insert_mesh_plane

parameter	data type	values [defaults]
xz_data	char	
loop	char	yes,[no]
xy_mesh	char	
side	intg	

insert_mesh_plane is used to insert additional mesh planes with the geo3d mesh connection tool and increase the mesh density in a specific region of a z-segment. This is most often used to added "bent" mesh planes that conform to rounded device shapes: this is a more economical solution compared to adding traditional mesh planes.

This statement is usually created automatically by the MaskEditor GUI program and loaded into the main simulation via a .zst file.

Parameters

- **xz_data** is the mesh plane profile in the xz plane. The file name follows a specific pattern which indicates the mask layer after which the plane is inserted.
- **loop** specifies that the plane profile in the xz plane forms a closed loop.
- **xy_mesh** is the mesh triangle information in the xy plane.
- **side** is a parameter automatically added by the MaskEditor GUI and CSUPREM; it describes a group of lines corresponding to the same set of inserted mesh planes.

Examples

```
insert_mesh_plane xz_data=layer2_ixz33.txt loop=yes &&
  xy_mesh=layer2_ixy33.msh
```

22.379 insert_mesh_range

parameter	data type	values [defaults]
xrange	realx2	

insert_mesh_range defines the range of x coordinates where additional mesh planes are present. This statement is used by the geo3d mesh connection tool.

22.380 integer_func

The command is identical to **loop_integer**.

22.381 interface

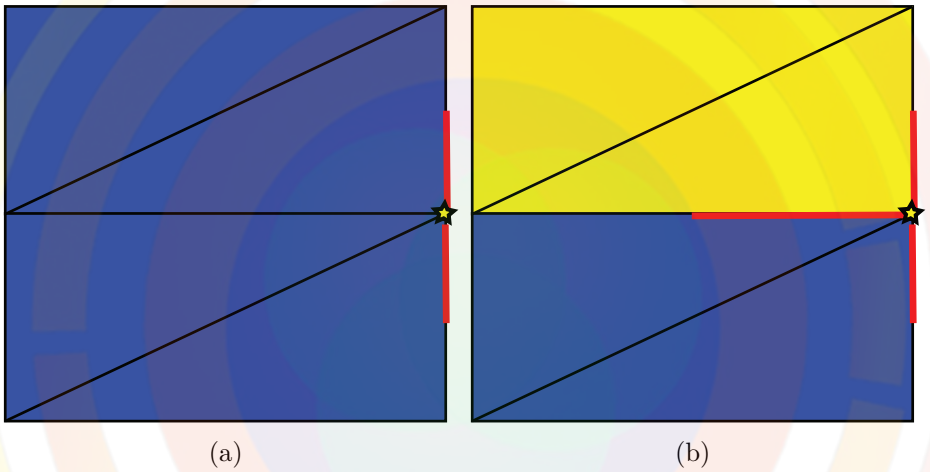
parameter	data type	values [defaults]
model	char	[recomb], charge, trap
trap_type	char	[donor], acceptor
x_label	char	[void]
y_label	char	[void]
within_x1_label	char	[void]
within_x2_label	char	[void]
within_y1_label	char	[void]
within_y2_label	char	[void]
zplane_label	char	[void]
upper_lower_cosine	char	yes,[no]
left_right_sine	char	yes,[no]
outer_recomb_only	char	yes,[no]
traplevel_model	char	[void], gaussian, expo_tail, uniform
traplevel_tail_side	char	[conduction], valence
within_x	realx2	[-1.e5 1.e5] (um)
within_y	realx2	[-1.e5 1.e5] (um)
velocity_n	real	[1.e6] (m/s)
velocity_p	real	[1.e6] (m/s)
fix_charge	real	[1.e14] ($1/m^2$)
trap_density	real	[1.e16] ($1/m^2$)
trap_level	real	[0.5] (eV)
trap_life_n	real	[1.e-7] (s)
trap_life_p	real	[1.e-7] (eV)
x	real	(um)
y	real	(um)
traplevel_stddev	real	[0.1] (eV)
traplevel_tail	real	[0.05] (eV)
traplevel_width	real	[0.5] (eV)
surftrap_num	intg	[1]
upper_lower_mater	intgx2	
left_right_mater	intgx2	
zplane_num	intg	

The **interface** statement is used to specify semiconductor interface states and the physical models used by the simulation software. The interface may be between semiconductor and vacuum or between different types of semiconductors. This statement may be used multiple times to define different interfaces.

Note that the solver internally deals with 3D quantities while this statement uses 2D sheet quantities as inputs. All values are there converted internally by software by using the local mesh area A and the equivalent interface length d:

$$\rho_{3D} = \frac{\sigma_{2D} * d}{A}$$

Note that for historical reasons (modeling of silicon devices), the default interface length is obtained by the adding up the triangle edge lengths at the boundaries between different materials: as per the usual finite volume method, half of the connecting triangle edge lengths is assigned to each mesh point. Unless special care is taken, this convention may lead to corner effects as seen in Fig. 22.14.



(a) (b)

Figure 22.14: Schematic interface representation of the interface length around a mesh point: the equivalent interface length is outlined in red. Note that the outside of the mesh is on the right of the figure: this exposed area automatically counts as a different material. a) interface between two points sharing the same material b) interface at the corner between two materials

Parameters

- **model** indicates the type of model to be used to describe the physics of the interface. The available options are:
  - *recomb* defines surface recombination velocities.
  - *charge* defines fixed interface charge densities.
  - *trap* defines surface traps.
- **trap_type** is the charge type of the deep surface traps (donor or acceptor).

- **within_x**, **within_y** defines the extent of the interface in the x and y directions, respectively. Usually, only one of these is used and the other coordinate is given with a fixed value.
- **velocity_n** and **velocity_p** are the surface recombination velocities for electrons and holes, respectively.
- **fix_charge** is the fixed charge surface density. It can be negative or positive.
- **trap_density** is the surface trap density.
- **trap_level** is the trap energy level measured from the conduction band.
- **trap_life_n** and **trap_life_p** is the trap life time of minority electrons and minority holes, respectively.
- **traplevel_model** may be used to define a continuous distribution of trap states. If this statement is omitted, then a single energy level is used.
- **traplevel_stddev** is the trap level standard deviation if the trap level model is gaussian.
- **traplevel_tail** is the characteristic decay constant (L) of the exponential tail model $e^{-E/L}$. The energy E is measured from either the conduction of valence band depending on the value of **traplevel_tail_side**.
- **traplevel_width** is the energy width for a uniform distribution of trap states.
- **x** and **y** are fixed coordinate values that define the position of the interface. Usually only one of these is used and the other coordinate is specified using a range.
- **surftrap_num** is a numeric label used to identify a specific kind of trap for later use. Multiple traps with different identifiers can coexist in the same region.
- **x_label** and **y_label** are pre-defined labels that can be used instead of the fixed values in **x** and **y**. These labels must be defined using the **x_position** and **y_position** statements. See the help for these statements for details on how they can be automatically generated.
- **within_x1_label**, **within_x2_label**, **within_y1_label** and **within_y2_label** are predefined labels similar to **x_label**. They control the ranges defined in **within_x** and **within_y**, respectively.
- **zplane_num** applies the interface settings from this command to entire mesh plane specified. If a label has previously been defined, **zplane_label** may be used instead.

- **upper_lower_mater** restricts the interface to the region where these 2 material numbers touch. Note that this kind of interface is meant to be used between horizontal layers, such as in the textures surfaces of a solar cell. Since there can be different interfaces at the top and bottom of a given layer, the order of the numbers is important: for example, (2 4) and (4 2) represent different interfaces.

  left_right_mater is similar to the above but may be used for materials that are in separate columns (e.g. at a regrowth region or at the edges of an etched mesa). Here again, the order of the materials is important, with the first number representing the material on the left side of the interface.

- **upper_lower_cosine** is used to apply a cos() scaling to surface charges/traps based on the shape of the interface. It can only be used in conjunction with **upper_lower_mater**.

  left_right_sine is similar but applies a $sin()$ scaling when used in conjunction with **left_right_mater**.

- **outer_recomb_only** modifies the calculation method for the equivalent interface length in Fig. 22.14 so that only the triangle edges exposed to the outer surface count. This correction eliminates the corner effects mentioned in the introduction to this command.

Examples

```
$ For a surface between 0.5 to 2 um, we define a Fermi level
$   pinning acceptor mid-gap trap
interface model=trap trap_type=acceptor &&
  within_x=(0.5 2.) within_y=(0.4 0.6) &&
  trap_density=1.e16 trap_level=0.8 &&
  trap_life_n=1.e-7 trap_life_p=1.e-7  surftrap_num=1
$
$ We can use a surface recombination model here:
interface model=recomb &&
  within_x=(0.5 2.) within_y=(0.4 0.6) &&
  velocity_n=1.e6 velocity_p=1.e6
```

22.382 interface_leakage

parameter	data type	values [defaults]
add_hopping	char	yes, [no]
conductance	real	[1.] ($\mho/m$ (2D),$\mho$ (3D))
conductance_hopping	real	[0.] ($\mho/m$ (2D),$\mho$ (3D))
trap_level_depth	real	[1.2] (eV)
hopping_distance	real	[1.] (μm, $(\mu m)^2$)
trap_size_hopping	real	[1.e-3] (μm)
trap_level_ref	real	[1.2] (eV)
between_contacts	intgx2	[1,2]

interface_leakage defines an additional current term which bypasses the normal mesh and is added directly to the contacts. This current term may be used to represent *ad hoc* leakage effects from interface traps.

Parameters

- **add_hopping** turns on a trap hopping model for the leakage current, with a conductance defined by **conductance_hopping**.

  The trap parameters for the hopping model have the same definition as in trap_assisted_tunneling: **trap_level_depth**, **hopping_distance**, **trap_size_hopping**, **trap_level_ref**.

- **conductance** determines the magnitude of the leakage current.

- **between_contacts** is a pair of contact numbers where the additional current term is added.

22.383 interface_mesh

parameter	data type	values [defaults]
accuracy	real	[1e-4] (um)

interface_mesh determines the accuracy of statements which define an interface at a particular location. Some fuzziness is required since the mesh points may not lie at exactly the desired location.

Examples

```
interface_mesh accuracy=1e-3
interface model=recomb y=1.0 &&
  velocity_n=1.e6 velocity_p=1.e6
```

With the default accuracy setting, the software will look for an interface at y=1.0 μm, ± 1e-3 μm.

22.384 interface_trap_capacitor

parameter	data type	values [defaults]
surf_trap_density	real	[1.e16] m^{-2}
effective_range	real	[1.] (μm(2D),μm^2(3D))
dft_dv	real	[1.e-3] (V^{-1})
between_contacts	intgx2	[1,2]

interface_trap_capacitor defines an additional current term which bypasses the normal mesh and is added directly to the contacts. This current term may be used to represent *ad hoc* capacitance effects from interface traps.

Parameters

- **surf_trap_density** is the density of surface trap states.
- **effective_range** defines the size of the virtual capacitor plates in this model.
- **dft_dv** describes the shift in trap occupancy due to the applied voltage.
- **between_contacts** is a pair of contact numbers used to determine the applied voltage on the virtual capacitor and on which the additional current term is added.

22.385 internal_z_xpoint

This statement is similar to **internal_xpoint** in that it specifies the position of extra mesh points at heterojunctions. The difference is that it applies to stacked mesh planes in a 3D simulation rather than to the interface between layers and columns.

22.386 integer_func

The command is identical to **loop_integer**.

22.387 internal_xpoint

parameter	data type	values [defaults]
all_interfaces	char	[yes],no
material_label1	char	
material_label2	char	
auto_mater_priority	char	[no],yes
turn_off_xp	char	[no],yes
xp_size	real	[0.0001] (μm)

The **internal_xpoint** statement is used to control the internal extra point spacing which defines the sharpness of the heterojunctions. The default behavior of the software is to always add an extra point slightly offset from the interface at polygon and contact boundaries. This extra point is necessary because mesh points in the Crosslight system can only be assigned a single material number so the actual boundary between two materials (and the accompanying change in the material properties) is mid-way between two neighboring mesh points. Without an extra point at interfaces, the position of the boundary and the thickness of certain layers may depend on the regular mesh spacing and affect the results of the simulation.

See also the **put_mesh** statement which defines the regular mesh point spacing for a given polygon edge. If the regular mesh spacing in certain regions is smaller than the internal point spacing given in this command, the software may decide to omit the superfluous extra point.

This statement may be used in both the .geo and .layer input files; it may be issued more than once to define different extra point spacings for various interfaces inside the device.

- **all_interfaces** flags whether all material interfaces are affected by this statement.
- **material_label1** and **material_label2** are used by .layer only. If used, it indicates that this statement is valid for the interface between these two materials. The labels used here must have been previously defined in a **layer_mater** declaration.

- **auto_mater_priority** decides which material the extra interface point is assigned to. If this parameter is set to *no*, the material with the lowest material number will be assigned the extra point. If this parameter is set to *yes*, the extra point will be assigned to the semiconductor region whenever a semiconductor/oxide interface is located.

- **turn_off_xp** may be use to deliberately turn off the creation of the extra mesh point. This is not usually recommended unless the mesh spacing near the interface is sufficiently small.

- **xp_size** is the extra point spacing described above.

22.388 isolate_complex

This statement is used in the .layer file to break a complex MQW region (cx-macro) into smaller pieces. When used after a **layer** or **column** statement, the complex region will immediately end and a new complex region will begin. This can be used to add some localization to a thick complex MQW region, much like the **independent_mqw** statement.

However, there is a significant difference:

- **independent_mqw** will assign a different active region number to each well in a complex MQW region. The confined level search is done in each well but the wave function is solved for the entire complex MQW region.

- **st:isolate_complex** will break up a complex MQW region into smaller pieces with different active region numbers. The Schrödinger equation and confined energy levels will be solved separately in each new complex region.

This statement has no parameters.

22.389 isolate_mesh_segment

parameter	data type	values [defaults]
add_z_space	real	(um)

This statement is used between **z_structure** statements to insert a void (unmeshed) region between these z-segments. This offset also has the effect of cutting

off mesh connections between these segments, much like the **z_connect** parameter of **3d_solution_method**.

The primary use of this statement is to treat multiple 2D devices within the same TCAD simulation. For example, a external circuit simulation using **minispice** might have more than one mixed-mode device. Since those devices are not necessarily monolithically integrated on-chip, adding a spacer region is necessary to prevent their mutual interaction at the FEM level. The spacer also helps clarify plots of the various quantities by separating the devices, an added benefit compared to using **z_connect**=*no*.

22.390 jdos_energy

parameter	data type	values [defaults]
data_file	char	
energy_range	realx2	
data_point	intg	[50]

jdos_energy plots the joint density of states (JDOS) versus energy. This statement is only used in the gain preview module. It is otherwise identical to **elec_dos_energy**.

22.391 kane_para_f_bar

kane_para_f_well is an active layer macro parameter which defines Kane's parameter F[61, Eq. 2.5] in a barrier region.

This parameter is only used in basic QW macros that contain parameters for both the well and barrier regions; it is otherwise identical to **kane_para_f_well**

22.392 kane_para_f_well

kane_para_f_well is an active layer macro parameter which defines Kane's parameter F[61, Eq. 2.5] in a quantum well region. This parameter defines the strength of the interaction between the conduction band and valence band in the zincblende $k \cdot p$ method; it is also included as a second-order correction to the dipole moments when calculating the interband transition strength.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par**

in section 22.456.

22.393 kp_model_setting

parameter	data type	values [defaults]
correction_for_8x8	char	[no], yes
optical_matrix_para_ep	real	(eV)
all_material	char	[yes]
mater_label	char	
mater	intg	

kp_model_setting is used to control some of the internal model settings in the zincblende $k \cdot p$ model.

Note that this command should only be used if the user desires more control over internal values used inside the Hamiltonian: the default settings should be correct but depend on correct calibration of macro parameters.

Parameters

- **correction_for_8x8** determines whether a correction term is applied to carrier mass and the Luttinger parameters when solving an 8x8 or block-diagonalized 4x4 Hamiltonian. If **kp_model_setting** is not used or if this parameter is equal to *yes*, a correction term is applied to account for the fact that the conduction band states are explicitly solved. In terms of Löwdin perturbation theory, the correction occurs because we remove the perturbative effects of the conduction band by moving it away from class B and into class A: the existing perturbation term thus needs to be removed from the coupling coefficients in the Hamiltonian.

  When this correction is applied, the following modified parameters are used to solve the Hamiltonian[66]:

$$\frac{m_0}{m_c}|_{\text{modified}} = \frac{m_0}{m_c} - E_p \frac{E_g + 2\frac{\Delta}{3}}{E_g + \Delta} \tag{22.31}$$

$$\gamma_1|_{\text{modified}} = \gamma_1 - \frac{E_p}{3E_g + \Delta} \tag{22.32}$$

$$\gamma_2|_{\text{modified}} = \gamma_2 - \frac{1}{2}\left(\frac{E_p}{3E_g + \Delta}\right) \tag{22.33}$$

$$\gamma_3|_{\text{modified}} = \gamma_3 - \frac{1}{2}\left(\frac{E_p}{3E_g + \Delta}\right) \tag{22.34}$$

where m_c is the electron mass, $\gamma_{1,2,3}$ are the Luttinger parameters, E_g is the bandgap, Δ is the spin-orbit coupling coefficient and E_p is the optical matrix parameter.

Under this correction approach, the parameters defined in material macros always refer to the experimental values that appear on the right-hand side of the above equation. These parameters are also appropriate to use as-is for lower-order $k \cdot p$ models which only take into account the coupling between valence bands and where conduction band perturbation effects remain in class B.

If **kp__model__setting** is used but **correction__for__8x8**=no, then no modification of the parameters occurs inside the software and the parameters from the macro are used as-is to solve the Hamiltonian. Note that since these parameters are used as-is for simulations that include conduction band coupling, their values would not be appropriate to use in lower-order $k \cdot p$ models.

We note that other authors[61] give a different correction term for the Luttinger parameters:

$$\gamma_1|_{\text{modified}} = \gamma_1 - \frac{E_p}{3E_g} \tag{22.35}$$

$$\gamma_2|_{\text{modified}} = \gamma_2 - \frac{E_p}{6E_g} \tag{22.36}$$

$$\gamma_3|_{\text{modified}} = \gamma_3 - \frac{E_p}{6E_g} \tag{22.37}$$

Therefore, disabling the automatic correction in the software and using the macro parameters as-is enables users to experiment with these different correction methods.

- **optical__matrix__para__ep**, if used, manually defines the value of the optical matrix parameter (E_p) in the correction formulas above. To explain the

importance of this parameter, let us recall the definition of Kane's parameter (F) from perturbation theory [61, 66]:

$$F = \frac{1}{m_0} \sum_j^{\Gamma_5} \frac{|< S|p_x|X >|^2}{E_c - E_v} \tag{22.38}$$

so that the experimentally-measured conduction band mass is given by:

$$\frac{m_0}{m_c} = 1 + 2F + E_p \frac{E_g + 2\frac{\Delta}{3}}{E_g + \Delta} \tag{22.39}$$

When applying the usual correction term for the 8x8 Hamiltonian, the perturbation from the conduction band must be removed and the modified mass used in the Hamiltonian is thus given by:

$$\frac{m_0}{m_c}|_{\text{modified}} = 1 + 2F \tag{22.40}$$

As we can see from the above, F and E_p are not independent fitting coefficients: this value is used to define the influence from both class A and class B states when solving the 8x8 Hamiltonian. While tabulated data may provide both values[61], for the sake of consistency only one of these parameters should be used for the entire model.

If **optical_matrix_para_ep** is not defined or if **kp_model_setting** is not used at all, the electron mass defined in the macro is assumed to be the experimentally-measured mass and the value of F is taken from **kane_para_f_well** and **kane_para_f_bar**. E_p is then obtained by inverting Eq.22.39.

If **optical_matrix_para_ep** is defined then this value is used for E_p and both **kane_para_f_well** and **kane_para_f_bar** will be ignored.

We note that defining this value has no influence on whether or not the 8x8 correction is applied in the first place: it merely controls what value of E_p is used for this correction.

- **all_material** instructs the software to use the same model settings for all materials in the simulation: this should not be used if this command is used to define the value of E_p.
- **mater** applies this command to a single material number; if labels have been defined for this material, **mater_label** may be used instead.

Examples

```
kp_model_setting correction_for_8x8=no optical_matrix_para_ep=22.4 &&
mater=1 all_material=no
```

22.394 lastip_compact_model

parameter	data type	values [defaults]
file	char	[crosslight_compact.txt]
polarization	char	[te], tm
active_region_width	real	[10] (μm)

lastip_compact_model is used to export device-scale simulation results from LASTIP as a simplified compact model which may be used in other software tools. This approach is often used to represent a Fabry-Perot laser diode in system-scale simulations which contain multiple photonic devices.

The format of the compact model may be obtained by simply running this command and examining the output file (text format). This file contains basic information about the laser cavity (length, mirror reflectivity, etc...), important simulation inputs (non-linear gain coefficient, active region width, etc...) and tabulated values for other key variables (material gain and spontaneous emission spectra, confinement factor, etc...).

The LASTIP compact model was developed for use in tools by Lumerical. Other software vendors interested in collaborating on this topic are invited to contact Crosslight.

Parameters

- **file** is the output filename used for exporting the compact model.
- **polarization** determines whether TE or TM data is exported in the compact model.
- **active_region_width** is the ridge width of the waveguide.

22.395 lateral_mode3d

parameter	data type	values [defaults]
sort_modes	char	yes,[no]
mode_num	intg	[1]

lateral_mode3d is used to enable the multi-lateral mode capability in PICS3D: each lateral mode is propagated so that it has its own set of longitudinal modes. Note that the mode solver itself must also be configured to find multiple lateral modes.

Parameters

- **sort_modes** determines the order in which lateral modes are used for PICS3D. If *yes*, the lateral mode with highest emission power will always be assigned as mode number one; otherwise, the lateral mode order in the 3D model will be the same as that in 2D wave equation solutions.

- **mode_num** is the total number of lateral modes to be simulated in PICS3D.

Examples

```
lateral_mode3d sort_modes=no mode_num=2
```

22.396 lattice_bar

This statement is used to define the unstrained lattice constant a (in Å) in the barrier region of a wurtzite active structure. See also **lattice_c_bar**.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.397 lattice_base

This statement defines the lattice constant a (in Å) on which a bulk (passive) wurtzite region is grown (e.g. strained to match a GaN buffer layer). It is used by the software as a reference lattice with which to compute the as-grown strain in the layer. See also **lattice_c_base**.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.398 lattice_bulk

This statement is used to define the unstrained lattice constant a (in Å) in bulk (passive) wurtzite regions. See also **lattice_c_bulk**.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.399 lattice_c_base

This statement defines the lattice constant c (in Å) on which a bulk (passive) wurtzite region is grown (e.g. strained to match a GaN buffer layer). It is used by the software as a reference lattice with which to compute the as-grown strain in the layer. See also **lattice_base**.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.400 lattice_c_bar

This statement is used to define the unstrained lattice constant c (in Å) in the barrier region of a wurtzite active structure. See also **lattice_bar**.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.401 lattice_c_bulk

This statement is used to define the unstrained lattice constant c (in Å) of a bulk (passive) wurtzite region. See also **lattice_bulk**.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.402 lattice_c_constant

This statement defines the lattice constant c (in Å) on which a wurtzite active region is grown (e.g. strained to match a GaN buffer layer). It is used by the software as a reference lattice with which to compute the as-grown strain in the layer. See also **lattice_constant**.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.403 lattice_c_well

This statement is used to define the unstrained lattice constant c (in Å) in the well region of a wurtzite active structure. See also **lattice_well**.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.404 lattice_constant

This statement defines the reference lattice constant a (in Å) on which an active region is grown. It also serves as a reference lattice constant to fix the size of the Brillouin zone for all calculations done in momentum (k) space.

For wurtzite materials, this value is often set to the lattice constant of a GaN buffer layer; this value is then used by the software to compute the as-grown strain in the layer. See also **lattice_c_constant** which defines the lattice constant along the c axis.

Note that materials grown on AlN for UV applications should not use the value of GaN as this will not produce the correct strain.

For zinclende materials, the strain is explicitly defined using the **strain_well** and **strain_bar**. This statement therefore does not determine the strain but must still be specified for other reasons.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.405 lattice_well

This statement is used to define the unstrained lattice constant a (in Å) in the well region of a wurtzite active structure. See also **lattice_c_well**.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.406 lax_mass_bar

lax_mass_bar is similar to **tax_mass_bar** but defines the longitudinal mass parallel to the symmetry axis rather than the transverse mass.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.407 lax_mass_well

lax_mass_well is identical to **lax_mass_bar** but defines the longitudinal/parallel mass in the well region rather than in the barrier.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.408 layer

parameter	data type	values [defaults]
vcsel_type	char	void
section_tag	char	void
add_leftmesh	char	void
d	real	[0.5]
r	real	[1.]
angle	real	[0.]
wj (j=1,2,...,30)	real	
lower_wj	real	
upper_wj	real	
n_dopingj	real	
p_dopingj	real	
gaussian_tail	real	[0.001] (μm)
shift_center	real	[0.] (fraction)
grading_fromj	real	
grading_anglej	real	[90.](degree)
z_gaussian_tail	real	[0.0] (μm)
n	intg	[5]
mj	intg	[1]
use_dbr_period	intg	[0]

layer defines the dimensions and mesh parameters of a material layer in the .layer file.

Please note that this statement can only be used to define a material with either n-type or p-type of doping, not both. To define a material with both types (that is, compensating doping), the statement **layer_mater** should be used. Unlike **layer**, **layer_mater** can define both types of doping at the same time.

Parameters

- **vcsel_type** is used to label the present layer as part of a VCSEL longitudinal section. Please note that this is only a label to be defined by the user. It works together with the same parameter in the **vcsel_section** statement. **section_tag** also serves the same purpose.
- **add_leftmesh** is used to add mesh-generating commands to the left side of polygons when generating the .geo file. This may be necessary in cases where there is a gap in the structure and mesh lines do not propagate as they should.

- **d** is the thickness of the layer.
- **r** is the ratio which determines mesh spacing characteristics. See Fig. 22.22 for details.

  shift_center is a related parameter which applies for negative mesh ratios and symmetric mesh distributions. This parameter is used to adjust the center of the distribution.
- **angle** is used in conjunction with **upper_w#** and **lower_w#** to specify the shape of the polygon formed by a column and layer.
  - **angle** is defined as the angle between the y-axis (left side of the column) and the edge of the material (left side of the polygon), in degrees.
  - **lower_w#** gives the width of the lower edge of a polygon in this layer, where # is the column number.
  - **upper_w#** gives the width of the upper edge of a polygon in this layer, where # is the identification number of its column.
- **n_doping#** and **p_doping#** define the n-type or p-type doping, respectively, in column #. This will generate a doping profile in the .doping file which can later be included in the main simulation file (.sol).

  Note that only one type of dopant can be used in a given layer: use **layer_mater** when both types of dopant need to be defined. Another important note is that the doping must be specified in SI units ($1/m^3$) rather than the frequently used $1/cm^3$.
- **gaussian_tail** is the Gaussian tail (standard deviation) of the doping profile defined by **n_doping#** or **p_doping#**.
- **grading_fromj** and **grading_anglej** are used to define linear grading profiles in column j. The maximum value of the doping in this layer is given by **n_doping#** or **p_doping#**.

  grading_fromj is the relative starting value (e.g. 0.01) of the doping profile. **grading_anglej** is the direction of variation of the doping: its value is in degrees and the reference direction (angle=0) is the +x direction.
- **n** is the number of mesh lines to be placed in this layer.
- **m#** can be used to assign a certain material number to the polygon formed by this layer and column #. This parameter is there for historical reasons only. For any new simulations, it is recommended to let the software assign material numbers automatically based on the macro name and composition parameters in **layer_mater**.

- **use_dbr_period** overrides the thickness parameter **d** and instead ensures that the layer thickness is equal to an integer multiple (the value of which is given in this parameter) of the previously-defined optical DBR period thickness. This parameter must be used in conjunction with the **vertical_dbr_layer_mater** statement to define the individual components of the DBR embedded in this layer.

  For DBRs that consist of a fractional number of pairs (e.g. 30.5 pairs of AlAs/GaAs), the layer should be split with the leftover fraction of the DBR as a separate layer with its own single-layer optical section.

- **z_gaussian_tail** is used to extend the xy doping defined in this layer along the z-direction. The value defined here corresponds to the **z_stddev** setting of the **doping** statement once the layer file is processed.

Examples

```
layer d=0.29  n=5  r=1.
```

22.409 layer_conf

parameter	data type	values [defaults]
layer_xrange	realx2	[0. 5.] (μm)
layer_yrange	realx2	[2. 2.1] (μm)
mode_index	intg	[1]

layer_conf is used as a post-processing statement to calculate the optical confinement factor for a rectangular region (not necessarily the active region).

- **layer_xrange** and **layer_yrange** are used to specify the rectangular cross section on the xy-plane (lateral) for which the optical confinement factor is to be calculated.

- **mode_index** is the lateral mode index. Mode 1 is the fundamental mode; mode 2 is the 2nd order mode, etc..

Example:

```
layer_conf layer_xrange=(0. 1.5) layer_yrange(1.7 1.85)
```

22.410 layer_height_ref

parameter	data type	values [defaults]
location	char	[left], middle, right
delta_x_from_left	real	(um)
delta_x_from_right	real	(um)

layer_height_ref is used in conjunction with **modify_layer_height** to define a reference position on the x-axis where the height is not modified.

This command's parameters are identical to those of **column_position** and positions are defined relative to the last **column** command that was issued.

22.411 layer_input_convention

parameter	data type	values [defaults]
layer_unit	char	[absolute], relative
ignore_active_thickness	char	yes,[no]
ref_wavelength	real	[1.] (μm)
ref_temperature	real	[300.] (K)
set_active_thickness	real	(μm)
absolute_scale	real	1.

This parameter is used in the .layer file to change default behaviors during processing.

Parameters

- **layer_unit** switches between absolute units for the layer thicknesses and a value relative to the wavelength.
- **ref_wavelength** is the reference wavelength when using relative values.
- **ref_temperature** is the temperature at which relative values are calculated.
- **ignore_active_thickness** instructs layer.exe not to use the layer thickness as an input to active region declarations; the thickness defined in **set_active_thickness** is used instead. This may be used in devices such as nanowires where quantum confinement occurs laterally rather than vertically.

- **absolute_scale** scales all thicknesses uniformly when processing the layer file. Unlike the **layer_unit** setting, the refractive index of the material in the .layer is ignored when scaling the thickness.

Examples

```
$ DBR average index=3.2 emission wavelength=0.834
$ For convenience, define reference wavelength=0.834/3.2=0.26

layer_input_convention layer_unit=relative ref_wavelength=0.26
```

22.412 layer_mater

parameter	data type	values [defaults]
macro_name	char	[void]
active_macro	char	[void]
solve_wave	char	[yes] (no)
material_label	char	[void]
contact_type	char	[ohmic]
n_impurity	char	[void]
p_impurity	char	[void]
zdir_grading	char	[no]
mater_lib	char	
var_symbolk(k=1..9)	char	
avar_symbolk(k=1..9)	char	
vari(i=1..9)	real	
grade_from	real	
grade_to	real	
avari(i=1..9)	real	
n_doping	real	
p_doping	real	
agrade_from	real	
agrade_to	real	
barrier	real	
work_function	real	
n_level	real	(eV)
p_level	real	(eV)
thick_zdir_grading	real	(um)
column_num	intg	[1]
grade_var	intg	[0]
n_newdoping_index	intg	
p_newdoping_index	intg	
agrade_var	intg	[0]
contact_num	intg	[0]

layer_mater is used in the .layer pre-processing to define the material in a particular layer and column. It can also be used to define the doping in that region, a capability which is partially shared with the **layer** statement. However, geometric and mesh properties for a given layer are solely the purview of the **layer** statement.

When used, **layer_mater** allows the layer.exe program to detect changes in material composition and automatically assign material numbers to the various polygons. It

also generates a .mater file with material macro loading statements that can be used in the full .sol simulation or the .gain preview.

Parameters

- **macro_name** is the name of the bulk/passive macro for this material. This macro contains all information related to the electrical transport properties of the material and any other useful information which is not related to optical gain calculations.

  The online help or Wizard can list available macro names in the default macros in the installation but custom macro names can be used provided the corresponding macro is later defined and included in the simulation with **use_macrofile**.

  A few special macro names deserve a bit more explanation:

  - *void* defines a region without any mesh and can be used to create columns with different numbers of layers. This should not be confused with *air* or *vacuum* macros which define dielectric materials and where mesh points will be assigned. This can have a strong effect on optical mode solutions as it changes the position of the wave boundary conditions.
  - *contact* defines the polygon as an equipotential boundary region. This can be used in place of **top_contact** and other similar commands in atypical situations where the contact is not located at the outer edge of the mesh. It is also used in 3D situations when stacking mesh planes and the contact area can be defined as an in-plane polygon.

    As with all contact declaration, it is also necessary to define the **contact_type** (ohmic or Schottky) and contact number (**contact_num**). For Schottky contacts, the electron barrier height (**barrier**) or work function (**work_function**) must also be defined. It is also recommended to define large doping concentrations in contact layers to align the quasi-Fermi levels with neighboring polygons more easily.

- **active_macro** is the same as **macro_name** but refers to the active macros that define the parameters for optical gain calculations. Active macros are therefore designed so that they can used in the .gain preview mode and are thus completely independent from the passive macros. This also means that quantum well active macros define parameters for both the quantum well and the barrier layers.

  Also note that by default, active macro settings will almost always override passive macro settings for the same parameter (e.g. bandgap). This means that a quantum well active macro will likely override parameters from the bulk macro in both the quantum well and barrier layers.

- **mater_lib** is the same as **macro_name** and **active_macro** but it assigns material properties to this layer using the combined "library" method of Sec. 3.5. Using this parameter will generate a **material_lib** statement when the .layer file is processed.

- **solve_wave** is used to exclude a particular region from the optical mode calculations. This will affect the settings of the **wave_boundary** statement that is automatically generated by layer.exe.

- **material_label** is a user-defined label that can be used to refer to this material in other statements.

- **var_symbolk**,k=1,...,9 are the symbolic variable names used as function arguments in the bulk material macro. With the exception of a few reserved keywords such as *temper*, the symbols defined here must exactly match those in the macro declaration.

  Note that all default macros now use the free-from style and thus depend on the symbol declarations to function properly. However, the older fixed-form style can still be used with older macros. In this case, the symbol declarations may be omitted and the software will assume that the order of the variables in **layer_mater** matches exactly that of the macro declaration.

  A similar set of parameters, **avar_symbolk**, define the symbols for the active macro. The same rules for free-form and fixed-form macros apply.

- **vari,(i=1,2,...9)** defines a fixed value for the ith symbol (free-form style) or function parameter (fixed-form style) in the passive macro. A similar set of parameters, **avari**, define the values for the active macro.

- **grade_from** and **grade_to** are the compositions values at the bottom and top of the layer, respectively, for a linearly graded region. These parameters only affect the passive macro parameter or symbol number specified in **grade_var**. Since the composition varies, the corresponding **vari** declaration should be omitted.

  Note that it is not currently possible to vary two or more parameters at the same time. However, users may rewrite macros so that all composition parameters change according to a single control variable (e.g. $z = 1 - x - y$).

  A similar set of parameters are used to grade active regions: **agrade_from**, **agrade_to** and **agrade_var**.

- **n_doping** and **p_doping** are the doping concentration in this layer/column polygon. This setting may conflict with the doping definition in the **layer** statement so doping should be define in either one statement or the other but not both. Note that unlike **layer**, both n and p doping can be defined in the same polygon.

Additional doping parameters can also be set here:

- **n_impurity** and **p_impurity** define the type of dopant.
- **n_level** and **p_level** define the activation energy of the dopant.
- **n_newdoping_index** and **p_newdoping_index** are used to identify a particular set of doping regions so that the doping concentration can be during the simulation.

All these settings are explained in greater detail in the **doping** statement, which is generated automatically by layer.exe.

- **column_num** is the column number on which the present statement operates.
- **zdir_grading** is used to indicate that the grading defined in this layer is along the z direction. This is mostly used when the .layer file serves as an input to 3D mesh generation GUI tools. The thickness of the layer should be indicated by **thick_zdir_grading** in this case since size information from **layer** does not carry over easily.

Examples

```
layer_mater macro_name=algaas var1=0.71 column_num=1
layer d=1.35  n=9   r=0.8 n_doping1=1.e24
```

The above two statements are used to define a layer of AlGaAs (Al=0.71)in column 1 with thickness 1.35 μm and n-type doping of 1×10^{24} $1/m^3$.

```
layer_mater macro_name=algaas grade_var=1 &&
  grade_from=0.71 grade_to=0.33 column_num=1
layer d=0.1462 n=6 r=0.8
```

The above two statements are used to define a graded AlGaAs with Al varying from 0.71 to 0.33 over a thickness of 0.1462 μm.

22.413 layer_position

parameter	data type	values [defaults]
label	char	
hline location	char	[top]
hline delta_y_from_bottom	real	[-9999.] (μm)
delta_y_from_top	real	[-9999.] (μm)

This statement is used to mark a specific y coordinate for later use. This is useful to automatically track changes to a device structure without having to redefine coordinates in other commands.

All positions defined by this command are relative to the preceding **layer** statement.

Parameters

- **label** defines a variable name that can be reused in other commands to refer to this position.
- **location** specifies a specific part of the layer at which the label is to applied (bottom, top or middle). More control is available by using the "delta" parameters below.
- **delta_y_from_bottom** specifies a point measured from the bottom of the layer. Positive values point towards the middle of the layer.
- **delta_y_from_top** specifies a point measured from the top of the layer. Positive values point towards the middle of the layer.

Examples

```
layer_mater macro_name=gan column_num=1 n_doping=1.e+24
layer  d=0.06  n=5  r=1.
$ n-GaN 1um
layer_mater macro_name=gan column_num=1 n_doping=1.e+24
layer  d=1.  n=10  r=0.8

$ GaN/AlGaN SL, effective medium
layer_mater macro_name=algan column_num=1 var_symbol1=x var1=0.08 &&
   n_doping=1.e+24
layer  d=0.28875  n=7  r=1.

layer_position  label=nsl_start  location=bottom
```

This defines a position named "nsl_start" at $y = 1.06$ when the .layer file is processed.

22.414 layer_type

parameter	data type	values [defaults]
type	char	(see list)
doped_organic	char	[no]
ox_use_extern_spec	char	[yes]
ox_use_exciton_diff	char	[yes]
dox_triplet	char	[no]
doxk(k>=2)_triplet	char	[no]
valley_gamma	intg	[1]
valley_l	intg	[4]
valley_hh	intg	[1]
valley_lh	intg	[1]
valley_ch	intg	[1]
mater	intg	[1]
ox_max_phonon	intg	[5]
ox_phonon_cloud	intg	[4]
ox_level_req	intg	[80]
dox_max_phonon	intg	[5]
dox_phonon_cloud	intg	[4]
dox_level_req	intg	[80]

layer_type is used in active layer macros to denote some general properties. In general, this statement should only be used by the end-user to design new material macros.

Parameters

- **type** activates certain physical models depending on the kind of active region being modeled in the active macro. It takes one of the following values:
  - bulk: thick active region with no confining potential.
  - unstrained_well: single QW with symmetric barriers and no strain effects.
  - strained_well: single QW with symmetric barriers and possibility of strain effects depending on composition.
  - complex: complex MQW region without strain effects.
  - strained_complex: complex MQW region with strain effects.
  - wurtzite_well: single QW with strain effects in the wurtzite material system.

  - wurtzite_complex: complex MQW region with strain effects in the wurtzite material system.
  - mos_complex: complex MQW region used in Quantum-MOS applications.
  - general_cx_strain: general-purpose complex MQW region with strain effects. Used when more flexibility in the parameters is required.
  - general_bulk: general-purpose bulk active region. Used when more flexibility in the parameters is required.
  - bulk_organic: bulk organic semiconductor such as those used in OLEDs.

  Refer to Sec. 8.1 for a comparison for the various QW models in Crosslight.

- **doped_organic** is to an organic material doped with another polymer that also contributes to the optical properties.
- **ox_use_extern_spec** indicates whether an experimental or measured EL spectrum is used instead of Crosslight's internal EL model for organic light-emitting regions.
- **ox_use_exciton_diff** indicates whether exciton diffusion model parameters are used in the present macro.
- **dox_triplet** defines whether a triplet-harvesting dopant is present.
- **doxk_triplet, k=2..5** defines the presence of additional triple-harvesting dopants
- **valley_gamma** is the number of band valley of the Gamma conduction band.
- **valley_l** is the number of band valleys of the L conduction band.
- **valley_hh** is the number of band valleys of the HH valence band.
- **valley_lh** is the number of band valleys of the LH valence band.
- **valley_ch** is the number of band valleys of the CH valence band.
- **mater** is the internal material number. This should be omitted when defining a macro but must be used when overriding macro parameters in the .sol file.
- **ox_max_phonon** is the maximum phonon occupation number at a lattice site in the organic crystal. It is part of the internal EL model for organic semiconductors.

- **ox_phonon_cloud** is the number of neighboring lattice sites to be considered for the construction of a phonon cloud. It is part of the internal EL model for organic semiconductors.

  A phonon cloud is defined as a set of localized excited lattice vibration states to interact with an exciton state. In the Frenkel exciton theory, a larger phonon cloud tends to give more accurate description of a realistic exciton state.

- **ox_level_req** is the number of energy levels requested when solving the exciton-phonon Hamiltonian. It is part of the internal EL model for organic semiconductors.

- **dox_max_phonon** is the maximum phonon occupation number at a dopant lattice site in the organic crystal. It is part of the internal EL model for organic semiconductors.

- **dox_phonon_cloud** is similar to **ox_phonon_cloud** except it is for the dopant material.

- **dox_level_req** is similar to **dox_level_req** except it is for the dopant material.

Examples

```
layer_type type=strained_well valley_gamma=1 valley_l=4 &&
  valley_hh=1 valley_lh=1
```

22.415 layers_for_semicrafter

parameter	data type	values [defaults]
plane_size_x	real	[2.0] (um)
plane_size_y	real	[2.0] (um)
plane_start_x	real	[0.0] (um)
plane_start_y	real	[.0] (um)
plane_mesh_x	intg	[30]
plane_mesh_y	intg	[30]
column_num	intg	[1]

layers_for_semicrafter is a .layer pre-processing statement used to generate a set of CSUPREM mesh generation commands. It is meant to be used as quick alternative to the SemiCrafter GUI for simple rectangular shapes.

When used, this statement changes the normal operation of the .layer file. Instead of defining the structure of an xy mesh plane, the layers now define the vertical stacking in the z direction with each layer corresponding to a z-segment. The shape of the structure in the xy plane (i.e. the top view) is a rectangle defined by using this statement.

See also **sym_polygon_for_semicrafter** for a more complex version of this command which allows symmetric polygons in the xy plane.

Parameters

- **plane_start_x** and **plane_start_y** determine the position of the lower left corner of the plane/polygon in the xy plane.
- **plane_size_x** and **plane_size_y** determine the extent of the plane/polygon in the xy plane.
- **plane_mesh_x** and **plane_mesh_y** determine the number of mesh points in the xy plane.
- **column_num** is the column number of the .layer file which is used to determine the z-segments.

Examples

```
layers_for_semicrafter plane_size_x=300 plane_size_y=300 &&
     plane_mesh_x=10 plane_mesh_y=10
```

22.416 lband_bar

parameter	data type	values [defaults]
(see) material_par		

The material statement **lband_bar** is an active layer macro statement used to define the L-band energy with respect to the Gamma band (eV) in the quantum barrier.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.417 lband_well

parameter	data type	values [defaults]
(see) material_par		

The material statement **lband_well** is an active layer macro statement used to define the L-band energy with respect to the Gamma band (eV) in the quantum well.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.418 led_control

parameter	data type	values [defaults]
efficiency_model	char	[void] xy, y uniform
spectrum_all_bias	char	yes [no]
spon_all_points	char	[no]
use_intern_refl	char	[no]
photon_recycle	char	[no], yes
efficiency	real	[0.1]
wavelength	real	(μm)
delta_wavel	real	[0.1]
refl_x1	real	[0.3]
refl_x2	real	[0.3]
refl_y1	real	[0.3]
refl_y2	real	[0.3]
group_index	real	[3.8]
led_xrange	realx2	(μm)
led_yrange	realx2	(μm)
spectrum_num	intg	[35]
x_segment	intg	[30]

The statement **led_control** is used to control the light emitting diode modeling with the assumption of simple cubic waveguide structure. Such a simple structure will allow some reasonable estimate of light power emission given the basic material and current injection conditions. Please note that this statement works with **init_wave** to defined a Fabry-Perot cavity.

For a more realistic power extraction model for devices different than the idea cubic waveguide structure, the ray tracing model in combination with the **led_simple** should be preferred over **led_control**.

Parameters

- **efficiency_model** is used to compute power extraction efficiency. As we have described in Sec. 14.2, the power emission depends on the available optical mode density distribution in the directions perpendicular to the direction of emission. Thus, the power emission depends on if the device is designed to confine optical modes in a certain direction.

  If this parameter takes *void*, the external emission efficiency is not computed and is set to the value of the **efficiency** parameter of this statement.

  If **efficiency_model**=*xy*, the software assumes a waveguide structure with well confined lateral / transverse modes as if we are treating an edge type of semconductor laser. For emission in the z-direction, the lateral modes are computed in the same way as for a semiconducor laser (i.e., in combination with **multimode** statement). For emission in the x-direction we assume 1D mode confinement in y-direction. For emission in the y-direction, no mode confinement is assumed.

  If **efficiency_model**=*y*, the software assumes that there is 1D single mode confinement in y-direction for emission in both z- and z-directions. Again, no mode confinement is assume for emission in y-direction.

  Finally, if the LED device is not designed to confine modes in any direction, we set **efficiency_model**=*uniform*. Only plane waves are considered when extraction efficiency is computed for this model.

- **spon_all_points** indicates whether all mesh points are used to evaluate the spontaneous emission spectrum. Computation time is longer if all points are used.

- **use_intern_refl** is used to turn on the internal reflection model for the 1-dimensional (or broad-area) emission calculation. Using this model results in an extraction efficiency lower than realistic values while not using it tends to overestimate the power extraction. Due to advanced power extraction geometrical design for modern LED's, the measured power extraction is closer to the broad-area emission model without internal reflection. Versions of APSYS before to Sept. 2004 used the internal reflection model as default.

- **photon_recycle** is used to enable the generation term due to the internal photon density. In thermal simulations, this setting allows heat to be generated

by optical absorption. However, please note that APSYS does not posses a photon rate equation: it is expected that LEDs operate below transparency.

- **spectrum_all_bias** indicates whether spontaneous emission spectrum is computed at all biases or only at data set print points.
- **wavelength** is the estimated peak emission wavelength in microns.
- **delta_wavel** is the wavelength range to cover the whole emission spectrum. The total emission power is the integration of the spectrum over this wavelength range.
- **refl_x1** is the power reflectivity of the LED cavity on the left side in the x-direction.
- **refl_x2** is the power reflectivity of the LED cavity on the right side in the x-direction.
- **refl_y1** is the power reflectivity of the LED cavity on the bottom side in the y-direction.
- **refl_y2** is the power reflectivity of the LED cavity on the top side in the y-direction.
- **group_index** is the group index of the traveling waves of the LED.
- **led_xrange** is the range in the x-direction that we consider to be a Fabry-Perot cavity. If not specified, the program will search the device geometry and decide the range.
- **led_yrange** is the range in the y-direction that we consider to be a Fabry-Perot cavity. If not specified, the program will search the device geometry and decide the range.
- **spectrum_num** is the number of points used to evaluate the spontaneous emission spectrum.
- **x_segment** is the number of segments in the x-direction that the program is to sample the material properties and carrier densities for emission in the x and y directions.

Examples

```
led_control wavelength=0.437 efficiency_model=uniform &&
  refl_y1=0.8 refl_y2=0.4 delta_wavel=0.03 &&
  led_xrange=(0 400) group_index=2.9
```

22.419 led_eff_distr

parameter	data type	values [defaults]
data_file	char	
side	char	[top]bottom

led_eff_distr plots the LED external efficiency distribution along the x-direction.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **side** indicates top or bottom side the efficiency is to be plotted.

Example(s)

```
led_eff_distr side=top
```

22.420 led_farfield

parameter	data type	values [defaults]
data_file	char	
side	char	[top]bottom

led_eff_distr plots the relative far field intensity distribution.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **side** indicates top or bottom side the efficiency is to be plotted.

Example(s)

```
led_farfield side=top
```

22.421 led_simple

The statement **led_simple** is a simplified version of **led_control** which provides the bare minimum parameters needed to compute the internal quantum efficiency (IQE) of a LED. All its parameters are explained in the latter statement.

Despite its simplicity, this model is recommended over **led_control** because it can be combined with the ray tracing post-processing to obtain a better estimate of the power extraction from the device.

22.422 led_spectrum

parameter	data type	values [defaults]
data_file	char	
total_amount	char	[yes] no
mode	char	[mix],te,tm
scale	real	[1.]
scale_wavelength	real	[1.]

led_spectrum is a post-processing statement that plots the LED spontaneous emission spectrum.

Parameters

- **data_file** can be used to export the spectrum data to a text file
- **total_amount** indicates whether the total amount (integrated amount) be plotted.
- **mode** determines whether the TE, TM or a combined (default) emission spectrum is being plotted.
- **scale** is the scale of the vertical axis.
- **scale_wavelength** is the scale of the horizontal axis.

22.423 led_top_coating

parameter	data type	values [defaults]
number_of_layers	intg	[1]
real_index_led	real	[3.5]
real_index_out	real	[1.5]
thicknessj (j=1...9)	real	[0.1] (μm)
real_indexj (j=1...9)	real	[2.2]
imag_indexj (j=1...9)	real	[0.]

led_top_coating is used as part of the **led_control** model and specifies an optical coating on the top of LED. Such a coating is usually applied to improve the extraction efficiency in the device.

Note that use of this statement overrides the reflection coefficient **refl_y2** in **led_control**. Instead, a thin-film transfer matrix method (TMM) is used to compute the reflectivity coefficient.

Parameters

- **number_of_layers** is an actual number of layers in the coating. The coating can contain up to 9 different layers.
- **real_index_led** is the real-valued refractive index of the LED material for the purposes of the TMM.
- **real_index_out** is the real-valued refractive index of the outer medium. for the purposes of the TMM.
- **thicknessj** is the thickness of the j-th layer from optical coating.
- **real_indexj** is the real part of the refractive index of the j-th layer in the optical coating.
- **imag_indexj** is the imaginary part of the refractive index of the j-th layer in the optical coating.

Examples

```
led_top_coating number_of_layers=2 real_index_led=3.5 real_index_out=1.5 &&
thickness1=0.15 real_index1=2.05 imag_index1=0.  thickness2=0.14  &&
real_index2=2.25 imag_index2=0.002
```

22.424 left_contact

left_contact is the same as **top_contact** except that the contact is placed on the left side.

One other difference is that the column number is irrelevant. The location of the contact is given by the last **layer** statement before the contact definition.

22.425 lifetime_model

parameter	data type	values [defaults]
dependence	char	[current], field
mater_label	char	
ref_elec_current	real	[1.e5](A/m^2)
ref_hole_current	real	[1.e5](A/m^2)
ref_tunnel_field	real	[1.e7] (V/m)
mater	intg	

lifetime_model modifies the SRH lifetime of the carriers based on the applied bias.

Parameters

- **dependence** switches between current and field control of the SRH lifetime.
  - If *current*, the capture cross-section is modified as $1 + \frac{I}{I_{ref}}$: more current increases the capture coefficient and reduces the SRH lifetime.
  - If *voltage*, the capture cross-section is modified as $e^{\frac{F-F_0}{F_{ref}}}$ where F_0 is the local field value at equilibrium. Therefore a larger applied field reduces the SRH lifetime.
- **ref_elec_current** and **ref_hole_current** are the values of I_{ref} for the electrons and holes, respectively.
- **ref_tunnel_field** is the value of F_{ref}.
- **mater** is the number of the material affected by this command. If a label has previously been defined as an alias, **mater_label** may be used instead.

Examples

```
lifetime_model ref_elec_current=1.e5 ref_hole_current=1.e5 &&
  dependence=current mater=5
```

would define a current dependent lifetime for both electrons and holes.

22.426 lifetime_n

The material statements **lifetime_n** and **lifetime_p** define the minority life time (in seconds) for carriers (_n for electrons and _p for holes) in the SRH recombination model. The basic mechanism is electron or hole capture by deep level traps. This statement is therefore related to other trap statements. The relation between lifetime and other trap quantities are written as:

$$\frac{1}{\text{lifetime}} = \text{trap density} \times \text{thermal velocity} \times \text{capture cross section} \qquad (22.41)$$

If the user does not specify a trap distribution but only gives a minority carrier lifetime, then a uniform distribution of donor mid-gap traps with density of 10^{10} m^{-3} is assumed. The carrier capture cross section is given by the above formula. However, if the user specifies the capture cross section explicitly with **trap_ncap_i** or **trap_pcap_i**, then the minority carrier lifetime is overridden.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.427 lifetime_p

See **lifetime_n**.

22.428 light_current

parameter	data type	values [defaults]
data_file	char	[void]
use_macro	char	[no]
wavel_range	realx2	
conc_range	realx2	[(1.e23 1.e24)] (m^{-3})

pn_ratio	real	[1.]
conc_log_scale	char	[no]
init_wavel	real	[0.83] (μm)
length	real	[300.] (μm)
mirror_ref	real	[0.3]
clad_bot	realx3	[1. 3.0 500.]
guide_bot	realx3	[.1 3.2 500.]
active_layer	realx3	[.007 3.33 500.]
guide_top	realx3	[.1 3.2 500.]
clad_top	realx3	[1. 3.0 500.]
front_back	realx2	
delta_index	real	[0.3]
av_index	real	[3.3]
auger_n	real	[2.e-42] (m^6/s)
auger_p	real	[2.e-42] (m^6/s)
life_n	real	[1.e-6] (s)
life_p	real	[1.e-6] (s)
spon_out	real	[1.e-4]
width	real	[5.] (μm)
scale_lit	real	[1.]
scale_curr	real	[1.]
max_curr	real	0.5 (A)
boundary_type	intgx4	[1 1 1 1]
well_num	intg	[1]
data_point	intg	[30]
mesh_point	intg	[80]

The statement **light_current** is a gain preview (.gain) command which allows the user to solve a simplified laser diode model and obtain the light-current characteristics. The basic recombination model is expressed as:

$$R = R\_spon + R\_auger + n/tau\_n + p/tau\_p$$

where the spontaneous emission term is given by the quantum well model.

Parameters

- **data_file** is a text file use to save the output data

- **use_macro** instructs the software to look for model parameters inside available material macros files.
- **wavel_range** is a wavelength search range used to find the gain peak for laser operation. For a device which locks the emission wavelength (e.g. a DFB/DBR laser or VCSEL), this range should be fixed to a single fixed wavelength.
- **conc_range** is the electron density range in the well. The hole density is given by this range and the $\frac{p}{n}$ ratio defined in **pn_ratio**.
- **conc_log_scale** spaces the electron density of **conc_range** on a logarithmic rather than linear scale.
- **init_wavel** is an initial guess of the emission wavelength for the optical eigenmode solver.
- **length** is the laser cavity length.
- **mirror_ref** is the mirror reflectivity in a Fabry-Perot cavity. For a DFB/DBR laser or for a VCSEL, this value should be set as an equivalent value that represents the feedback strength from the distributed mirror.

  For asymmetric facets (e.g. HR/AR configurations), **front_back** should be used to override **mirror_ref**.
- **clad_bot** is a triplet pair that defines the thickness (μm), refractive index and material loss (m^{-1}) of the bottom cladding layer.

  Similar triplets are defined for other layers in the simplified structure solved by this command:

  - **guide_bot** is used for the bottom guiding layer (SCH).
  - **active_layer** is used for the active layer (MQW region). In this simplified model, the entire MQW region is considered as a block for the optical mode solver.
  - **guide_top** is used for the top guiding layer (SCH).
  - **clad_top** is used for the top cladding layer.
- **delta_index** and **boundary_type** are used to set up the optical solver. Please refer to the definition of these terms inside the main **init_wave** statement.
- **av_index** is the estimated average refractive index.
- **auger_n** and **auger_p** are the Auger coefficients C_n and C_p, respectively.

-

- **life_n** and **life_p** are the minority carrier lifetimes for the electrons and holes, respectively.

- **spon_out** is the fraction of the isotropic spontaneous emission that couples into the optical propagating mode; it is often written as β.

- **width** is the width of the cavity.

- **scale_lit** and **scale_curr** are scaling factors for the light and current, respectively; they can be used when modeling half of a laser device to exploit left/right symmetry.

- **max_curr** is the maximum injection current.

- **well_num** is the total number of quantum wells in the active region.

- **data_point** is the number of data points to be used in the current versus concentration plot.

- **mesh_point** is the number of mesh points used in this simplified laser model.

22.429 light_power

parameter	data type	values [defaults]
spectrum_file	char	[void]
light_dir	char	[top] left right bottom
waveguide_mode	char	[no] yes
gen_datafile	char	[void]
gen_datatype	char	[sorted] random
const_index	char	[no]
fresnel_reflections	char	[yes] no
slit_diffr	char	[no]
ox_pump_only	char	[no] yes
std_wave	char	[yes] no
old_tmm	char	[no] yes
spectrum_field_file	char	[void] myfile
force_update	char	[no]
import_gen_rate	char	[no]
optical_generation	real	[0.] $(m^-3)/s$
penetration_depth	real	[1.] (μm)
incident_power	real	[0.] (W/m^2) or W
wavelength	real	[1.] (μm)
profile	realx4	[-1.e8 1.e8 1. 1.] (μm)
fraction_mode2	real	[0.]
fraction_mode3	real	[0.]
fraction_mode4	real	[0.]
fraction_mode5	real	[0.]
angle	real	[0.]
fraction_TE	real	[0.5]

The statement **light_power** is used to specify the incident light power in a photosensitive device like a solar cell or an optically pumped laser. The model is based on thin film multi-layer propagation of plane waves. Based on the structure of the device, the software will create one or more 1D cut lines across the device in the direction of the light propagation; these are used to compute transfer matrices and get the local photon density.

Note that when using **light_power**, a **scan** statement must be used to ramp up the effective light power in the device since the equilibrium calculations assume no light input. Using

```
scan var=light value_to=1.0
```

will slowly ramp up the relative light input to the value defined in **incident_power** or **spectrum_file**. This can be combined with other scan variables such as time (for transient pumping) or electrode bias.

Using this model triggers the requirement for the Advanced Optics optional module which contains and the associated thin film model. A more basic model with simple exponential decay can still be used for users who do not have this module by explicitly turning off the Fresnel reflections and standing wave effects.

Parameters

- **spectrum_file** is used to specify the file name of the data file describing the incident light power spectrum (e.g. input solar spectrum). The data must be in a text format with at least 2 columns: the wavelength in μm in the first column and the power spectral density in $W/(m^2 \mu m)$ in the second column. Other columns will be ignored.

  For single-wavelength light input (e.g. a laser), the power and wavelength of the light input are defined by **incident_power** and **wavelength**, respectively.

- **light_dir** specifies the direction of the incident light.

- **waveguide_mode** is used to indicate that the incident light is directly coupled to one of the optical modes instead of using the thin film transfer matrix. In that case, mode solver statements such as **init_wave** must be used. For multimode simulations, **fraction_modek** k=2,...,5 are used to describe the fraction of the input power which is coupled into the higher-order modes.

  Note that this is only used for 2D simulations. For light injection into a photo-absorbing waveguide or optical amplifier, see **waveguide_input** to take longitudinal effects into account.

- **gen_datafile**, is the name of a text file which contains the incident optical wave intensity profile. In general, it is preferable to use GnuPlot-compatible data files. For 2D simulation, the optical intensity profile on x-y plane should be given in three columns as follows

  ```
  x y optical\_profile
  ```

  with x varying the fastest. Please leave a blank line to separate data lines between different y-coordinates.

  For 3D simulation, the data should be arranged in the form

  ```
  x y z optical\_profile
  ```

with x varying the fastest and y the second fastest. Again, blank lines are used to separate data lines between different y coordinates.

Note that "optical_profile" should be considered as a relative value: the actual power used in the simulation will be the local profile value multiplied by **incident_power**. For waveguide input problems, it is convenient to set **incident_power** to the total input power in W while the profile defines a normalized intensity distribution in m^{-2}. The local value of the incident power will therefore be in W/m^2 and the integrated value will be in W.

Please note that we use the optical profile here to compute the local energy/photon density and automatically scale by the group velocity to get units of W/m^2. This value includes standing wave effects and should not be confused with the power flux density, which does not.

- **gen_datatype** controls the format of **gen_datafile**. If set to *sorted*, the data order is arranged in a form compatible with GnuPlot with left most column varies the fastest and with a blank line space separating different drawing line data. For unsorted data, a simple listing without blank space line is required.

- **const_index** is used to force the simulator to average the complex refractive index along the cut line before the transfer matrix is computed. This will automatically suppress all internal Fresnel reflections and force a uniform exponential decay of the light. Since this model is highly unphysical, the user is warned against using this option.

- **fresnel_reflections** may be used to artificially turn off the internal reflections between layers of the device. This forces all material interfaces to have 100% power transmission but unlike **const_index**, the absorption coefficient is not averaged and the result will be a piecewise exponential decay of the light.

  Since this is unphysical, the user is warned against turning off this effect. However, it may be necessary to do so for users without access to the Advanced Optics optional module. It can still provide a reasonable model for devices with a single material such as silicon solar cells.

  Fresnel reflections at the front and back facets may also be artificially defined using the **front_reflection** and **back_reflection** statements. However, **optic_coating** statements will be useless unless this effect is turned on.

- **slit_diffr** would enable the single slit diffraction model be used to compute the optical intensity profile. The single slit would be defined by the first two numbers of the **profile**.

- **ox_pump_only** forces the optical generation to create bound excitons instead of free carriers. This is the expected behavior in organic semiconductors.

- **std_wave** determines whether the standing wave component of the transfer matrix is computed. This term is generated by over-sampling the optical mesh and averaging fast oscillations over the local mesh point to ensure smoothness and ensure convergence. Users without the Advanced Optics module will have to turn this effect off to use **light_power**.

  Note also that for optically pumped VCSELs in PICS3D, this term refers only to the standing wave of the optical pump: the round-trip gain calculations automatically include this effect for the laser light.

- **old_tmm** is used to activate an older transfer matrix model. This parameter is not recommended.

- **spectrum_field_file** is used to specify the optical field intensity profile for different wavelengths when the light source is continuous. The data format and units are the same as that defined for **gen_datafile** above, except that a header line "spectrum_field_file" followed by a line containing the wavelength (in micron meters) is used to start a field profile for each wavelengths.

- **force_update** is used to force the update of the optical transfer matrix at each bias step. This is rarely needed since other conditions in the code often make this decision on behalf of the user based on self-consistency requirements.

- **import_gen_rate** tells the software to import the local generation rate instead of the optical wave profile in **gen_datafile**. This is different from **optical_generation** defined below which uses the penetration depth to get the local generation rate profile.

- **optical_generation** may be used to directly specify the carrier generation rate at the starting point of the light absorption path. This parameter is used only when no optical input power is defined: when both **incident_power**=*0* and **spectrum_file**=*void*. When used, the actual distribution of the optical generation rate is calculated by this number multiplied by a attenuation factor due to the absorption of the semiconductor material.

- **penetration_depth** is used together with **optical_generation** only. It is used to describe the exponential decay of the incident light along the absorption path.

- **profile** is the cross section profile of the incident light: (x1, x2, dx1, dx2). This profile is flat between x1 and x2 and has Gaussian tails on each side (dx1 and dx2) to ensure smoothness.

- **angle** is the angle of incidence of the light with respect to the surface. It is used to consider the case of off-normal illumination. To model angled and textured surfaces, users should use ray tracing or FDTD simulations.

- **fraction_TE** is the fraction of TE polarized light; the remainder is assumed to be TM. Appropriate boundary conditions are used for the plane waves in the thin film model. The default value of 0.5 was chosen to represent unpolarized light common in most solar cell applications.

Examples

```
light_power incident_power=3.29e7 wavelength=0.82 &&
  profile=(0.5, 2.0, 0.01, 0.01)
```

The above statement is used to define an incident light of single wavelength at 0.82 μm with power density of 3.29e7 W/m^2. The incident light direction is from the top (default) and the cross section profile is uniform from 0.5 to 2 μm with a Gaussian tail of 0.01 μm width on each side.

22.430 light_power_qwip

parameter	data type	values [defaults]
abs_spectrum_file	char	[qwip_specfile.txt]
light_dir	char	[top],bottom
uniform_extraction	char	[no],yes
gen_datafile	char	[void]
gen_datatype	char	[sorted],random
incident_power	real	[1.e4](W/m^2)
wavelength	real	[5.] (um)
profile	realx4	[-9000. 9000. 1. 1.] (um)
power_couple	real	[0.6]
absorption_coef	real	[1.e-12](m)
active_layer_depth	real	[0.](um)
average_index	real	[3.2]
surf_elec_dens	real	[2.e15](1/m^2)
qwip_period	real	[0.04](um)
scale_abs_spec	real	[1.](um)

The statement **light_power_qwip** is used to specify the incident light power and other related parameter for a quantum well infrared photodetector (QWIP) .

- **abs_spectrum_file** is the absorption spectrum file to be imported. Its format is 3-column with first column being the wavelength in meters, 2nd

column being the absorption in 1/m, and the 3rd column being the absorption divided by the surface electron density.

- **light_dir** is the direction of light.

- **uniform_extraction** is used to indicate whether uniform or global field is used to extract the photo-carriers. One possible explanation for uniform field model is that photo-carriers with higher energies are not well localized and thus tend to transport with an averaged field instead of the inhomogeneous local field.

- **gen_datafile**, if specified, will provide the external input data of incident optical power flux profile. The format of the data is compatible with the public domain program GNUPLOT. For 2D simulation, the optical intensity profile on x-y plane is given in three columns as follows

  ```
  x y optical\_profile
  ```

  For 3D simulation, the data are arrange in the form

  ```
  x y z optical\_profile
  ```

  The unit of "optical_profile" here depends on the unit used in **incident_power**. The program multiplies the "optical_profile" in the data file with **incident_power** to provide the actual optical power flux density (in units of $Watt/m^2$). If it is a problem of 2D waveguide power problem, it is convenient to set **incident_power** to the total waveguide power (in Watt) while the data file provides a normalized intensity distribution (in units of $1/m^2$ so that the profile integrates to unity). For problem of incident light, it may be convenient to let **incident_power** to represent incident power in units of $Watt/m^2$ while "optical_profile" being the distribution of power transmission coefficient. In any kind of arrangements, it is necessary to remember that **incident_power** and the figure in data file must multiplied to obtain light intensity in units of $Watt/m^2$.

- **gen_datatype** If it is set to "sorted", the data order is arranged in a form compatible with GNUPLOT with left most column varies the fastest and with a blank line space separating different drawing line data. For unsorted data, a simple listing without blank space line is required.

- **incident_power** is the incident power density.

- **wavelength** is the wavelength of the QWIP.

- **profile** is the cross section profile of the incident light. It is in the format of (x1, x2, dx1, dx2) for light traveling in the $\pm y$ direction, where the light intensity is assumed to be uniform between x1 and x2 and decay in the left and right sides with standard deviations of dx1 and dx2, respectively. Similar explanation applies to light traveling in $\pm x$ direction with a format of (y1, y2, dy1, dy2).
- **power_couple** is a power coupling coefficient of the incident light into the device
- **absorption_coef** is the absorption divided by the surface density. It is not used if **abs_spectrum_file** is used.
- **active_layer_depth** is the depth of the MQW from the exposed surface of QWIP. This is used to locate the starting point of the MQW layers so that optical field profile can be established.
- **average_index** is the average index of the QWIP.
- **surf_elec_dens** is the estimated density so that the absorption can be calculated to produce the incident light intensity profile. The absorption is calculated by multiplying this parameter with the **absorption_coef** above.
- **qwip_period** is the QWIP period.
- **scale_abs_spec** is used to scale the abspectrum spectrum.

Example(s)

```
$ power-coupling into absorption mode is treated as fitting parameter for now.

light_power_qwip abs_spectrum_file=qwip_specfile.txt light_dir=top  &&
      incident_power=1.e4 wavelength=8 surf_elec_dens=1.5e15 &&
      qwip_period=0.0376 power_couple=0.1 uniform_extraction=yes
```

The surface electron density is used to multiply to the 3rd column in spectrum-file to obtain the absorption which is used to estimate the optical field intensity profile.

22.431 linear_heat

parameter	data type	values [defaults]
(see) material_par		

The material statement **linear_heat** is used to define a heating source term proportional to the current density. The unit is $Watt/(msAmp)$. It may also take a negative value to represent cooling effect.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.432 load_macro

parameter	data type	values [defaults]
name	char	
var_namej (j=1...9)	char	
var_symbolj (j=1...9)	char	
varj (j=1...9)	real	
mater	intg	[1]

As in many other FEM-based software, the device layout in Crosslight is discretized so that every mesh node is assigned a material number: the **load_macro** statement is used to simultaneously assign several material properties (e.g. bandgap, mobility, etc...) to a given material number. Material properties are then evaluated based on the supplied parameters and certain reserved keywords. Note that this statement is usually automatically generated by processing the .layer file.

For more information about macros, consult Appendix B and the comments in the *crosslight.mac* and *more.mac* files. See also **use_macrofile** to use custom user-defined macros and **get_active_layer** which is a similar statement used to load active macros.

Parameters

- The parameter **name** is the name of the macro being activated. The usual convention for passive macro is to use lower-case names.
- The parameters **var_symbol1-9** are the symbolic variable names used as function arguments in the macro. If defined, they must exactly match the symbols used in the macro function definitions. If not defined, then the simulator will assume an older macro style is being used: in this case, the order of the parameters must be the same as in the macro function definitions.

Note that certain variables used in functions are reserved keywords and do not need to be defined in this manner.

- The parameters **var1-9** are the values of the variables appearing in the functions within the macro definition. They are most commonly used to represent material composition in ternary/quaternary compounds.
- The parameters **var_name1-9** replaces **var1-9** when a grading of the macro parameters is used. It is used in conjunction with the **mater_var** statement which describes the spatial variation of the parameters.
- The parameter **mater** is the material number being linked with this macro. This number is the same as the material number assigned to polygons in the .geo file and is inherited by the mesh.

Examples

```
load_macro name=ingaas mater=2 var1=0.32 var_symbol1=x
```

This statement activates a material macro named ingaas (for $In_xGa_{1-x}As$) for material number 2 with material variable $x = 0.32$.

```
mater_var name=c2   variation=linear_y data_num= 4 &&
  4_values=[ &&
    0.135000000000E+001  0.710000000000E+000 &&
    0.149620000000E+001  0.330000000000E+000 ]
load_macro name=algaas mater=   2 &&
  var_symbol1=x var_name1=c2
```

This statement activates the macro named algaas (for $Al_xGa_{1-x}As$) for material number 3. The composition (x) is graded according to the rule "c2" which is further described in the accompanying **mater_var** statement.

22.433 load_mesh

parameter	data type	values [defaults]
mesh_inf	char	
version2002	char	[no]
suprem_import	char	[no]
stl_import	char	[no]
suprem_cpl_import	char	[no]
zseg_num	intg	1

load_mesh loads a previously generated mesh file into the device simulation.

Parameters

- **mesh_inf** is the data file containing the mesh. In practice, this file is generated by statements such as **mesh_output** that are automatically added to the .geo file when the .layer file is processed.

- **version2002** is used to accept mesh files generated by versions older than 2002. The older mesh format can also be generated with newer versions of the simulator with special settings of the **internal_xpoint** statement in the .geo file.

- **suprem_import** is used to import a mesh structure from CSUPREM. If **suprem_cpl_import** is used in a 3D simulation, the device simulator will also load the inter-plane (z) mesh coupling coefficients which can greatly speed up the simulation time.

- **stl_import** is used to import a mesh using Standard Tessellation Language (STL format).

- **zseg_num** defines the z-segment where the mesh is used. In 3D simulations, each z-segment can have its own material numbers, geometry and mesh

Examples

```
load_mesh mesh_inf=bulk1d.msh2
```

22.434 longitudinal

parameter	data type	values [defaults]
fphase_input	char	[yes],no
layer_exit_model	char	[refractive_index],reflectivity
ring_mode	char	[void],cc,cw
length_seg_match	char	[no]
use_eff_index	char	[no]
ignore_rtg_phase	char	[no]
left_phase	real	[0.] (π)
right_phase	real	[0.] (π)
left_f_refl	real	[0.]
right_f_refl	real	[0.]
ref_wavel	real	(m)
ref_pitch	real	(m)
add_overlap	real	[100.]
bot_real_ind	real	[1.]
bot_imag_ind	real	[0.]
top_real_ind	real	[1.]
top_imag_ind	real	[0.]
ring_power_exit	real	[0.1]
round_trip_time_factor	real	[1.]
ref_active_point	real	(um)
more_mesh	intg	[5]
cavity_num	intg	[1]

The **longitudinal** statement is used to specify some boundary conditions and control settings relating to the longitudinal model in PICS3D. For vertical cavities, left/right should be understood as bottom/top.

Parameters

- **fphase_input** determines how the residual facet phase is computed. If *yes*, the values of **left_phase** and **right_phase** are used. Otherwise, the facet phase is obtained from the length of the laser and the grating period and then split equally between the left and right facets.

- **left_f_refl** and **right_f_refl** are the left and right facet reflectivity (on a power basis) used as cavity boundary conditions for edge-emitting devices.

- **layer_exit_model** determines the cavity boundary conditions for surface-emitting devices.

  - If set to *reflectivity*, the left and right facet reflectivity defined above are used for the bottom and top, respectively.
  - If set to *refractive_index* (the default), the refractive index is used to compute the Fresnel reflection coefficient. **bot_real_ind** and **bot_imag_ind** define the outside index for the bottom layer while **top_real_ind** and **top_imag_ind** define the outside index for the top layer.

- **ring_mode** determines the main injection/propagation direction in a ring laser cavity. *cw* means clockwise and *cc* means counter-clockwise.

- **ring_power_exit** defines the fraction of power that is extracted at each loop of a ring laser: this is used instead of the left/right facet reflectivity in this kind of device since no facets can be clearly defined.

- **length_seg_match** will force an error message to be printed if the optical (section) length of the cavity does not match the electrical (z-segment) length.

- **use_eff_index** is used to determine how the effective index of the propagation constant β is calculated. If this is turned on, the exact value from the eigenmode solver is used which improves the numerical accuracy when the wave function is discontinuous and derivative terms become large (e.g. E-field in vectorial TM mode).

- **ignore_rtg_phase** is an experimental model added in v. 2015. It is used to make PICS3D approximate the behavior of LASTIP.

  One of the main differences in LASTIP vs. PICS3D is the longitudinal treatment. LASTIP is a 2D model and ignores the phase matching condition; it assumes that a single longitudinal mode exists and that lasing occurs on the modal gain peak. This approach is reasonable for Fabry-Perot lasers in which a large number of closely-spaced longitudinal modes exist near the gain peak. Some inaccuracies in the 2D model creep up for longer-cavity lasers though, especially for high-power applications: in that case, longitudinal spatial hole burning is a real effect which limits device performance.

  On the other hand, PICS3D includes modeling of LSHB automatically: using multiple mesh planes (for edge-emitters), it explicitly calculates the longitudinal propagation and solves for both unity and phase matching conditions. This means PICS3D will find multiple longitudinal modes and lase at wavelengths where the *cavity* round-trip gain peaks rather than on the peak of the modal gain curve. This approach works well in DFB/DBR lasers and VCSELs

but is problematic for Fabry-Perot lasers: without a wavelength-selecting feedback mechanism, there are too many longitudinal modes to consider and the numerical requirements/solver stability often become unmanageable.

Setting **ignore_rtg_phase**=*yes* bridges this gap in capability. Instead of solving for multiple longitudinal modes, PICS3D will search for a single wavelength which maximises the round-trip gain value: this ignores the phase-matching condition while still including the effects of LSHB. This hybrid approach is expected to be especially useful in modeling long-cavity high-power FP lasers.

- **ref_wavel** is the reference wavelength which is used to fix the search range for the longitudinal modes. This is fixed once the coupling of the round-trip gain equations is turned on so it is important to capture the right set of longitudinal modes for the simulation. In most laser cavities with a grating, the lasing mode will be close to the Bragg wavelength so the latter should be used as the reference wavelength.

  ref_pitch may be used instead of the above to directly specify the reference pitch of the Bragg grating.

- **round_trip_time_factor** can be used to artificially scale the cavity round-trip time in transient simulations.

- **ref_active_point** defines a reference point in the cavity which is used to compute the round-trip gain. While in theory, this value should be equal everywhere in the device, numerical evaluation of this quantity can be sensitive. In general, a point near the middle of the active region or where the photon density is expected to peak should be used to represent the round-trip of an "average" photon.

  This value is automatically chosen by the software by default; users may choose to override it if convergence problems occur or if anomalies in the round-trip gain spectrum are observed. Also note that this parameter replaces ref_midpoint from older versions of the software and unlike the old method, absolute coordinates must be used instead of a fraction of the optical cavity.

- **add_overlap** is used to add a small term (a multiple of $2j/v_g$) to the expression $\frac{dW}{d\omega}$ which appears in the complex pole expansion of Sec. 18.2. This term seems to improve the stability of the longitudinal mode solver by preventing divide by zero errors. It may slightly affect the amount of spontaneous emission power in ASE simulations such as superluminescent diodes (SLED) and optical amplifiers (SOA).

- **more_mesh** is a multiplication factor used to oversample the longitudinal mesh in order to get more details on the fast oscillations of the standing wave pattern in VCSELs.

- **cavity_num** can be used to specify different longitudinal settings when working with multiple optical cavities. See also **begin_cavity**.

Examples

```
longitudinal ref_wavel=1.55d-6 left_f_refl=0. right_f_refl=0.
```

22.435 loop_integer

loop_integer serves the same function as **loop_real** but defines an integer loop variable. Both statements share the same set of parameters.

Examples

```
start_loop symbol=%i value_from=1 value_to=5 step=2
loop_integer symbol=%j value_from=2 value_to=6
plot_scan scan_var=voltage_2 variable=total_curr_3 scanline=%i
plot_scan scan_var=voltage_3 variable=total_curr_3 scanline=%j
```

In the above example, symbol %j is a linear function of the loop variable %i. %i would vary as 1,3, and 5 while %j would go as 2,4 and 6. The equivalent commands without the loop are as follows:

```
plot_scan scan_var=voltage_2 variable=total_curr_3 scanline=1
plot_scan scan_var=voltage_3 variable=total_curr_3 scanline=2
plot_scan scan_var=voltage_2 variable=total_curr_3 scanline=3
plot_scan scan_var=voltage_3 variable=total_curr_3 scanline=4
plot_scan scan_var=voltage_2 variable=total_curr_3 scanline=5
plot_scan scan_var=voltage_3 variable=total_curr_3 scanline=6
```

22.436 loop_real

parameter	data type	values [defaults]
symbol	char	[void]
value_from	real	[1]
value_to	real	[2]

loop_real works together with the **start_loop** statement. It is used to define a new real-valued variable which has a linear dependence on the iteration number.

Parameters

- **symbol** is the symbol used by the new loop variable.
- **value_from** and **value_to** are the starting and ending values, respectively, of the loop variable.

Examples

```
start_loop symbol=%i value_from=1 value_to=5
loop_real symbol=%vg value_from=0.2 value_to=1.0
scan var=voltage_2 value_to=%vg
end_loop
```

This means %vg linearly varies with %i. %i would take values of 1,2,3,4 and 5 while %vg would vary as 0.2, 0.4, 0.6, 0.8 and 1.0. The equivalent commands without the loop are as follows:

```
scan var=voltage_2 value_to=0.2
scan var=voltage_2 value_to=0.4
scan var=voltage_2 value_to=0.6
scan var=voltage_2 value_to=0.8
scan var=voltage_2 value_to=1.0
```

22.437 loopif

loopif is a conditional execution statement used in conjunction with symbolic loop variables and **start_loop**.

The end of the conditional block is indicated by **endloopif**.

Parameters

This statement does not take any formal parameters but expects a logical expression based on common Fortran syntax:

.lt.	less than
.le.	lesser or equal than
.gt.	greater than
.ge.	greater or equal than
.eq.	equal to

The logical expression should be kept as simple as possible: parentheses and other logical expressions such as .and., .or. and .not. are not supported.

Examples

```
define_symbol symbol=%mob value=1.
define_symbol symbol=%mfp value=0.1

start_loop symbol=%i value_from=1 value_to=25
integer_func symbol=%k value_from=0 value_to=24
integer_func symbol=%j1 value_from=2 value_to=26
integer_func symbol=%j2 value_from=3 value_to=27
integer_func symbol=%j3 value_from=4 value_to=28

loopif   %j1 .le. 25
nonlocal_path y_start_label=active%i_end y_end_label=active%j1_start  &&
  mobility=%mob mean_free_path=%mfp field_dep_model_id=1
endloopif
```

22.438 low_field_mobility_model

parameter	data type	values [defaults]
el_doping_dependence_model	char	[void], masetti, arora, uni_bologna
el_carrier_carrier_model	char	[void], conwell_weisskopf, brooks_herring
el_degradation_model	char	[void], enhanced_lombardi, uni_bologna_inversion
el_philips_unified_model	char	yes,[no]
hole_doping_dependence_model	char	[void], masetti, arora, uni_bologna
hole_carrier_carrier_model	char	[void], conwell_weisskopf, brooks_herring
hole_degradation_model	char	[void], enhanced_lombardi, uni_bologna_inversion
hole_philips_unified_model	char	yes,[no]
channel_interface_dir	char	[void], horizontal, vertical
channel_interface_label	char	
mater	intg	[1]

This statement is used to implement various contributions to the low-field carrier mobility model. The mobility is then further enhanced by the field depending on the velocity models defined in **material**.

When different contributions are turned on, the total mobility is given by:

$$\frac{1}{\mu} = \frac{1}{\mu_{dop}} + \frac{1}{\mu_{carr}} + \frac{1}{\mu_{surf}} \tag{22.42}$$

where μ_{dop} represents bulk impurity scattering, μ_{carr} is the contribution from carrier-carrier scattering and μ_{surf} defines the mobility reduction from surface effects.

The relationship between the different models can be seen in Fig. 22.15.

Full equations for each of these models will be given below ; in all of these models, we note that $N_i = N_A + N_D$ is the total dopant concentration, T is the local mesh point temperature and T_0 is a reference temperature of 300K.

Main Parameters

Since this statement handles a very large number of parameters, only those related to model selection are described here: see sub-sections below for other model-specific parameters. Please note that the default values for model-specific parameters are taken from the literature and apply only for silicon: these models should be re-calibrated if used for other materials.

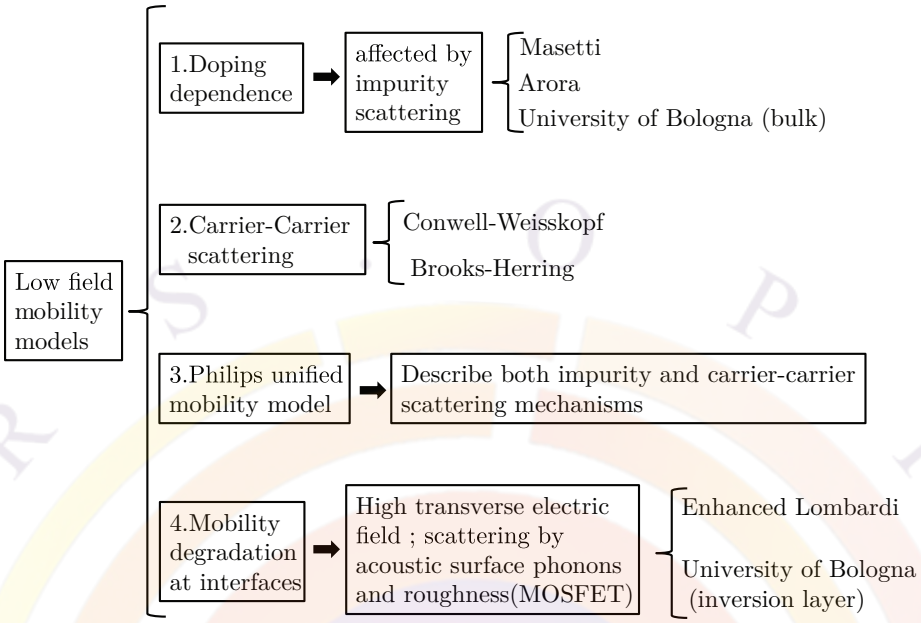


Figure 22.15: Relationship between the different low-field mobility models.

- **el_doping_dependence_model** and **el_doping_dependence_model** specify the doping/trap scattering model for electrons and holes, respectively. If this parameter is *void* and the unified Philips model is also disabled, the software will default back to the standard model of Eq. 5.42.

- **el_carrier_carrier_model** and **hole_carrier_carrier_model** define the carrier-carrier scattering model for electrons and holes, respectively.

- **el_degradation_model** and **hole_degradation_model** describe the surface degradation model for electrons and holes, respectively.

  If a surface degradation model is enabled, then the direction of the interface must be specified with **channel_interface_dir**. The position of the interface must also be set through a position label given in **channel_interface_label**. See **layer_position** and other related commands for details on position labels.

- **el_philips_unified_model** and **hole_philips_unified_model** turn on the unified Philips model for electrons and holes, respectively. If this model is turned on, the doping dependence and carrier-carrier models will automatically be disabled in favor of this unified model.

Parameters For Doping Dependence Models

Masetti

This model implements the following relationship[123]:

$$\mu_{dop} = \mu_{min1} e^{-\frac{P_c}{N_i}} + \frac{\mu_{max}\left(\frac{T}{T_0}\right)^{-\zeta} - \mu_{min2}}{1 + \left(\frac{N_i}{C_r}\right)^{\alpha}} - \frac{\mu_1}{1 + \left(\frac{C_s}{N_i}\right)^{\beta}} \tag{22.43}$$

The other parameters of this model (along with their default values) are defined in the table below:

symbol	parameter name	electron value	hole value	units
μ_{max}	umax	0.1417	0.04705	$m^2/(Vs)$
ζ	exponent	2.5	2.2	
μ_{min1}	umin1	0.00522	0.00449	$m^2/(Vs)$
μ_{min2}	umin2	0.00522	0	$m^2/(Vs)$
μ_1	u1	0.00434	0.0029	$m^2/(Vs)$
P_c	pc	0	9.3E22	m^{-3}
C_r	cr	9.68E22	2.23E23	m^{-3}
C_s	cs	3.34E26	6.1E26	m^{-3}
α	alpha	0.68	0.719	
β	beta	2.0	2.0	

Note that the parameter names are abbreviated here for the sake of convenience. When the Masetti model is activated, an el_mase_ (for electrons) or hole_mase_ (for holes) prefix must be added to the name shown in the table.

Arora

This model implements the following relationship[124]:

$$\mu_{dop} = \mu_{min} + \frac{\mu_d}{1 + \left(\frac{N_i}{N_0}\right)^{A^*}} \tag{22.44}$$

The other parameters for this model (along with their default values) are defined in the following equations and table:

$$\mu_{min} = A_{min}\left(\frac{T}{T_0}\right)^{\alpha_m} \tag{22.45}$$

$$\mu_d = A_d\left(\frac{T}{T_0}\right)^{\alpha_d} \tag{22.46}$$

$$N_0 = A_N\left(\frac{T}{T_0}\right)^{\alpha_N} \tag{22.47}$$

$$A^* = A_a\left(\frac{T}{T_0}\right)^{\alpha_a} \tag{22.48}$$

symbol	parameter name	electron value	hole value	units
A_{min}	umin	0.0088	0.00543	$m^2/(Vs)$
α_m	alpha_m	-0.57	-0.57	
A_d	d	0.1252	0.0407	$m^2/(Vs)$
α_d	alpha_d	-2.33	-2.23	
A_N	n	1.25E33	2.35E23	m^3
α_N	alpha_n	2.4	2.4	
A_a	a	0.88	0.88	
α_a	alpha_a	-0.146	-0.146	

Note that the parameter names are abbreviated here for the sake of convenience. When the Arora model is activated, an el_ar_ (for electrons) or hole_ar_ (for holes) prefix must be added to the name shown in the table.

University of Bologna

This model implements the following relationship[125]:

$$\mu_{dop} = \mu_0 + \frac{\mu_L - \mu_0}{1 + \left(\frac{N_D}{C_{r1}T_n^{\gamma_{r1}}}\right)^{\alpha} + \left(\frac{N_A}{C_{r2}T_n^{\gamma_{r2}}}\right)^{\beta}} - \frac{\mu_1}{1 + \left(\frac{N_D}{C_{s1}T_n^{\gamma_{s1}}} + \frac{N_A}{C_{s2}}\right)^{-2}} \tag{22.49}$$

where $T_n = \frac{T}{T_0}$ and:

$$\mu_L = \mu_{max}T_n^{-\gamma+cT_n} \tag{22.50}$$

$$\mu_0 = \frac{\mu_{0d}T_n^{-\gamma_{0d}}N_D + \mu_{0a}T_n^{-\gamma_{0a}}N_A}{N_A + N_D} \tag{22.51}$$

$$\mu_1 = \frac{\mu_{1d}T_n^{-\gamma_{1d}}N_D + \mu_{1a}T_n^{-\gamma_{1a}}N_A}{N_A + N_D} \tag{22.52}$$

The other parameters for this model (along with their default values) are defined in the following table:

symbol	parameter name	electron value	hole value	units
μ_{max}	umax	0.1441	0.04705	$m^2/(Vs)$
c	c	0.07	0.0	
γ	gama	2.45	2.16	
γ_{0d}	gama_0d	0.6	1.3	
μ_{0d}	u_0d	0.0055	0.009	$m^2/(Vs)$
γ_{0a}	gama_0a	1.3	0.7	
μ_{0a}	u_0a	0.0132	0.0044	$m^2/(Vs)$
γ_{1d}	gama_1d	0.5	2.0	
μ_{1d}	u_1d	0.00424	0.00282	$m^2/(Vs)$
γ_{1a}	gama_1a	1.25	0.8	
μ_{1a}	u_1a	0.00735	0.00282	$m^2/(Vs)$
γ_{r1}	gama_r1	3.65	2.2	
C_{r1}	C_r1	8.9E22	1.3E24	m^{-3}
γ_{r2}	gama_r2	2.65	3.1	
C_{r2}	C_r2	1.22E23	2.45E23	m^{-3}
γ_{s1}	gama_s1	0.0	6.2	
C_{s1}	C_s1	2.9E26	1.1E24	m^{-3}
C_{s2}	C_s2	7E26	6.1E26	m^{-3}
α	alpha	0.68	0.77	
β	beta	0.72	0.719	

Note that the parameter names are abbreviated here for the sake of convenience. When the University of Bologna model is activated, an el_uni_bolo_ (for electrons) or hole_uni_bolo_ (for holes) prefix must be added to the name shown in the table.

Parameters For Carrier-Carrier Scattering Models

In this section, n and p are the carrier densities.

Conwell-Weisskopf

This model implements the following relationship[126]:

$$\mu_{carr} = \frac{D\left(\frac{T}{T_0}\right)^{\frac{3}{2}}}{\sqrt{np}}\left[ln\left(1+\frac{F}{\sqrt[3]{np}}\left(\frac{T}{T_0}\right)^2\right)\right]^{-1} \tag{22.53}$$

The other parameters for this model (along with their default values) are defined in the following table:

symbol	parameter name	electron value	hole value	units
D	D	1.04E16	1.04E16	$m^{-1}V^{-1}s^{-1}$
F	F	7.452E17	7.452E17	m^{-2}

Note that the parameter names are abbreviated here for the sake of convenience. When the Conwell-Weisskopf model is activated, an el_con_wei_ (for electrons) or hole_con_wei_ (for holes) prefix must be added to the name shown in the table.

Brooks-Herring

This model implements the following relationship[127]:

$$\mu_{carr} = \frac{c_1 \left(\frac{T}{T_0}\right)^{\frac{3}{2}}}{\phi\left(\eta_0\right)\sqrt{np}} \tag{22.54}$$

$$\phi\left(\eta_0\right) = ln\left(1+\eta_0\right) - \frac{\eta_0}{1+\eta_0} \tag{22.55}$$

$$\eta_0 = \frac{c_2}{N_c F_{-\frac{1}{2}}\left(\frac{n}{N_c}\right) + N_v F_{-\frac{1}{2}}\left(\frac{p}{N_v}\right)} \left(\frac{T}{T_0}\right)^2 \tag{22.56}$$

where N_c, N_v are the densities of state for the electrons and holes, respectively, and $F_{-\frac{1}{2}}$ is the derivative of the usual Fermi-Dirac integral of order 1/2. The other parameters for this model (along with their default values) are defined in the following table:

symbol	parameter name	electron value	hole value	units
c_1	c1	1.56E15	1.56E15	$m^{-1}V^{-1}s^{-1}$
c_2	c2	7.63E25	7.63E25	m^{-3}

Note that the parameter names are abbreviated here for the sake of convenience. When the Brooks-Herring model is activated, an el_bro_her_ (for electrons) or hole_bro_her_ (for holes) prefix must be added to the name shown in the table.

Parameters For Philips Unified Model

This model combines dopant and carrier scattering effects using the following relationship[128]:

$$\frac{1}{\mu} = \frac{1}{\mu_L} + \frac{1}{\mu_{DAeh}} \tag{22.57}$$

where $\mu_L = \mu_{max}\left(\frac{T}{T_0}\right)^{-\theta}$ describes the effects from phonon scattering and μ_{DAeh} includes all other bulk mechanisms (free carriers, ionized dopants, etc...):

$$\mu_{DAeh} = \mu_N \left(\frac{N_{sc}}{N_{sc,eff}}\right)\left(\frac{N_{ref}}{N_{sc}}\right)^{\alpha} + \mu_c \left(\frac{n+p}{N_{sc,eff}}\right) \tag{22.58}$$

with:

$$\mu_N = \frac{\mu_{max}^2}{\mu_{max} - \mu_{min}}\left(\frac{T}{T_0}\right)^{3\alpha - 1.5} \tag{22.59}$$

$$\mu_c = \frac{\mu_{max}\mu_{min}}{\mu_{max} - \mu_{min}}\left(\frac{T}{T_0}\right)^{0.5} \tag{22.60}$$

The scattering center density is given by:

$$N_{sc} = \begin{cases} N_D^* + N_A^* + p & \text{for electrons} \\ N_D^* + N_A^* + n & \text{for holes} \end{cases} \tag{22.61}$$

while the effective scattering center density follows:

$$N_{sc,eff} = \begin{cases} N_D^* + G\left(P_e\right) N_A^* + \frac{p}{F(P_e)} & \text{for electrons} \\ N_D^* + G\left(P_h\right) N_A^* + \frac{n}{F(P_h)} & \text{for holes} \end{cases} \tag{22.62}$$

In the above equations, N_A^* and N_D^* are modified dopant concentrations which account for clustering effects at very high concentrations[1]:

$$N_X^* = N_X\left[1 + \frac{1}{c_X}\left(\frac{N_X}{N_{X,ref}}\right)^2\right] \text{ for X=A,D} \tag{22.63}$$

while screening effects are represented by the analytic functions F and G:

[1]The original reference indicates that the ionized dopant concentration should be used here; for the sake of simplicity, Crosslight uses the total dopant concentration as of the 2014 version. This approximation should hold for shallow dopants at room temperature.

$$F\left(P_i\right)=\frac{0.764P_i^{0.6478}+2.2999+6.5505\frac{m_i^*}{m_j^*}}{P_i^{0.6478}+2.3670-0.8552\frac{m_i^*}{m_j^*}} \tag{22.64}$$

$$G\left(P_i\right)=1-\frac{a_g}{\left[b_g+P_i\left(\frac{m_0}{m_i^*}\frac{T}{T_0}\right)^{\alpha_g}\right]^{\beta_g}}+\frac{c_g}{\left[P_i\left(\frac{m_0}{m_i^*}\frac{T}{T_0}\right)^{\alpha_g'}\right]^{\gamma_g}} \tag{22.65}$$

where $i=e,h$ for either electrons or holes, $\frac{m_i^*}{m_0}$ is a fitting parameter related to the relative effective mass and $\frac{m_j^*}{m_0}$ is the matching fitting parameter for the other carrier. The argument P_i of these functions is a screening parameter defined as a weighed harmonic mean of the Brooks-Herring and Conwell-Weisskopf approaches:

$$P_i=\left[\frac{f_{CW}}{3.97\times10^{13}N_{sc}^{-\frac{2}{3}}}+\frac{f_{BH}}{\frac{1.36\times10^{26}}{n+p}\frac{m_i^*}{m_0}}\right]^{-1}\left(\frac{T}{T_0}\right)^2 \tag{22.66}$$

The other parameters for this model (along with their default values) are defined in the following table:

symbol	parameter name	electron value	hole value	units
μ_{max}	umax	0.1417	0.04705	$m^2/(Vs)$
μ_{min}	umin	0.00522	0.00449	$m^2/(Vs)$
θ	theta	2.285	2.247	
N_{ref}	e_ref	9.68E22	2.23E23	m^{-3}
α	alpha	0.68	0.719	
$N_{D,ref}$	d_ref	4E26	4e26	m^{-3}
$N_{A,ref}$	a_ref	7.2E26	7.2e26	m^{-3}
c_D	c_d	0.21	0.21	
c_A	c_A	0.5	0.5	
m_e^*	me	1.0	1.0	
m_h^*	mh	1.258	1.258	
m_0	m0	1.0	1.0	
f_{CW}	fcw	2.459	2.459	
f_{BH}	fbh	3.828	3.828	
a_g	ag	0.89233	0.89233	
b_g	bg	0.41372	0.41372	
c_g	cg	0.005978	0.005978	
α_g	alpha_g	0.28227	0.28227	
α_g'	alpha_g2	0.72169	0.72169	
β_g	beta_g	0.19778	0.19778	
γ_g	gamma_g	1.80618	1.80618	

Note that the parameter names are abbreviated here for the sake of convenience. When the Philips unified model is activated, an el_phili_ (for electrons) or hole_phili_ (for holes) prefix must be added to the name shown in the table.

Parameters For Surface Degradation Models

These models are mainly used in MOSFET applications where high transverse electric fields cause the carriers to strongly interact with the semiconductor/oxide interface. These effects are highly localized and throughout this section, a scaling parameter $D = e^{-\frac{\text{distance}}{l_{crit}}}$ is used to turn off the mobility degradation away from the interface. Surface effects also cause anisotropic mobility and throughout this section, $F_\perp$ is defined as the transverse electric field normal to the interface.

As such, it is not recommended to combine the surface degradation models of this section with the **mobility_xy** statement. Crosslight's recommendation is to use the surface degradation models for MOSFET/transistor applications (especially in silicon) while the **mobility_xy** model is better suited to scale the mobility in nanostructured materials such as superlattice blocking layers or VCSEL DBR mirrors.

University of Bologna Inversion Layer

This model combines the scattering effects from Coulombic effects (μ_{cb}), surface acoustic phonons (μ_{ac}) and surface roughness (μ_{rough}) using the following relationship[129]:

$$\frac{1}{\mu_{surf}} = \frac{1}{\mu_{cb}} + \frac{D}{\mu_{ac}} + \frac{D}{\mu_{rough}} \tag{22.67}$$

The Coulomb term and screening effects are given by:

$$\mu_{cb} = \mu_{bulk}\left[D\left(1 + f_{sc}^{\tau}\right)^{\frac{1}{\tau}} + (1 - D)\right] \tag{22.68}$$

$$f_{sc} = \left(\frac{N_1}{N_A + N_D}\right)^{\eta} \frac{N_{min}}{N_A + N_D} \tag{22.69}$$

where N_{min} is the minority carrier concentration.

The surface scattering terms are defined as:

$$\mu_{ac} = c\left(\frac{T}{T_0}\right)^{-\gamma_c}\left(\frac{N_A + N_D}{N_2}\right)^{a} F_\perp^{-\delta} \tag{22.70}$$

$$\mu_{rough} = d\left(\frac{T}{T_0}\right)^{\gamma_d}\left(\frac{N_A + N_D + N_3}{N_4}\right)^{b} F_\perp^{-\gamma} \tag{22.71}$$

The other parameters for this model (along with their default values) are defined in the following table:

symbol	parameter name	electron value	hole value	units
N_1	n1	2.34E22	2.02E22	m^{-3}
N_2	n2	4E2	7.8E21	m^{-3}
N_3	n3	1E23	2E21	m^{-3}
N_4	n4	2.4E24	6.6E23	m^{-3}
c	c	1.8	0.5726	$m^2/(Vs)$
γ_c	gamac	1.6	1.3	
d	d	5.80E15	7.82E11	$m^2/(Vs)$
γ_d	gamad	0	1.4	
τ	tau	1.0	3.0	
η	eta	0.3	0.5	
a	a	0.026	-0.02	
b	b	0.11	0.08	
l_{crit}	lcrit	1E-8	1E-8	m
δ	delta	0.29	0.3	
λ	lambda	2.64	2.24	

Note that the parameter names are abbreviated here for the sake of convenience. When the University of Bologna Inversion Layer model is activated, an el_uni_boloinve_ (for electrons) or hole_uni_boloinve_ (for holes) prefix must be added to the name shown in the table.

Enhanced Lombardi

This model is an enhanced version of the Lombardi model found in **mobility_xy**. It combines the scattering from surface acoustic phonons (μ_{ac}) and surface roughness (μ_{rough}) using the following relationship[130]:

$$\frac{1}{\mu_{surf}} = \frac{1}{\mu_{bulk}} + \frac{D}{\mu_{ac}} + \frac{D}{\mu_{rough}} \tag{22.72}$$

The acoustic phonon contribution takes the form:

$$\mu_{ac} = \frac{B}{F_\perp} + \frac{C\left(\frac{N_i}{N_0}\right)^\lambda}{F_\perp^{\frac{1}{3}}\left(\frac{T}{T_0}\right)^k} \tag{22.73}$$

while the surface roughness term is defined by:

$$\frac{1}{\mu_{rough}} = \frac{1}{\delta}\left(\frac{F_\perp}{F_{ref}}\right)^{A^*} + \frac{F_\perp^3}{\eta} \tag{22.74}$$

$$A^* = A + \alpha_\perp (n+p)\left(\frac{N_{ref}}{N_i + N_1}\right)^v \tag{22.75}$$

with a reference field of F_{ref}=1 V/cm and a reference doping density of F_{ref}=1 cm^{-3}.

The other parameters for this model (along with their default values) are defined in the following table:

symbol	parameter name	electron value	hole value	units
B	B	4.75E5	9.925E4	m/s
C	C	0.26921	1.36788	$m^{\frac{5}{3}}s^{-1}V^{-\frac{2}{3}}$
N_0	n0	1E6	1E6	m^{-3}
N_1	n1	1E6	1E6	m^{-3}
λ	lambda	0.125	0.0317	
k	k	1.0	1.0	
δ	delta	5.82E10	2.0546E10	$m^2/(Vs)$
A	A	2.0	2.0	
$\alpha_\perp$	alpha	0.0	0.0	m^3
v	v	1.0	1.0	
η	eta	5.82E32	5.82E32	$m^{-1}s^{-1}V^2$
l_{crit}	lcrit	1E-8	1E-8	m

Note that the parameter names are abbreviated here for the sake of convenience. When the Enhanced Lombardi model is activated, an el_lombard_ (for electrons) or hole_lombard_ (for holes) prefix must be added to the name shown in the table.

Examples

```
low_field_mobility_model  mater=1 &&
   el_philips_unified_model = yes el_phili_umax=0.1417
```

This statement activates the unified Philips model for electrons in material #1 and the parameter umax in the function is specified as 0.1417 $m^2/(Vs)$.

22.439 lplot_xy

parameter	data type	values [defaults]
data_file	char	[vttek]
variable	char	(see list)
n_variables	charxn	
integration	char	[no]
x_from_label	char	[void]
y_from_label	char	[void]
x_to_label	char	[void]
y_to_label	char	[void]
math_oper	char	[void]
xy_from	realx2	
xy_to	realx2	
xrange	realx2	
yrange	realx2	
z	real	
integration_start	real	[0.0](um)
integration_length	real	[1.e5](um)
var_num	intg	[1]
mode_index	intg	[1]
trap_index	intg	[1]

lplot_xy is a post-processor statement used to plot data along a 1D cut line. It is similar to **plot_1d** and **lplot_xyz**. Depending on the 2D/3D nature of the simulation, the following rules should guide the choice of plotting command:

- 2D simulations: use **plot_1d**
- 3D cylindrical simulations with one mesh plane: use **plot_1d**
- xy cut of a 3D simulation: use **lplot_xy**
- z cut of a 3D simulation: use **lplot_xyz**

Parameters

Most parameters for this function are similar to the ones for **plot_1d** and are omitted for the sake of brevity. The following parameters are specific to **lplot_xy**:

- **z**, in conjunction with **xy__from** and **xy__to**, defines the beginning and ending points of the 1D cut line. Note that **xrange** and **yrange** also provide an alternate way to define the extent of the cut line in the mesh plane.

Unlike **plot__1d**, parameters related to plotting of the QW states are not available in this function. Use **lplot__xy__qw__states** instead.

Examples

```
lplot_xy variable=wave_intensity xy_from=(0.5 1.1) &&
  xy_to=(0.5 1.9) z=200.
```

22.440 lplot__xy__qw__states

parameter	data type	values [defaults]
data__file	char	[void]
xrange	realx2	
yrange	realx2	
qw__wave__ht	real	[0.1] (eV)
length__below__qw	real	[-9999.]
length__above__qw	real	[-9999.]
cond__subband	intg	[1] 2
val__subband	intg	[1] 2 3
zseg__num	intg	[1]
complex	intg	[1]

lplot__xy__qw__states is a variation of **lplot__xy**. However, instead of plotting a 1D cut line from a 3D simulation, it plots the QW states on the band diagram.

Parameters

- **data__file** is the name of a text file in which a copy of the plot data will be saved.
- **xrange** and **yrange** can be used to define the plot region.
- **qw__wave__ht** is the height of the quantum wave amplitude as plotted on the band diagram. However, the simulation must first have been run using the **self__consistent** statement. It is also necessary to output the QW wave data using **more__output**.

- **length_below_qw** and **length_above_qw** extend the plotting region beyond the QW. This can make it easier to observe the spatial extent of wave functions.
- **cond_subband** is used when **qw_wave**= 1 to indicate which conduction band is being plotted. The main condition band valley (Γ) is equal to 1 while 2 represents the side valley (L, X).
- **val_subband** is used when **qw_wave**= 1 to indicate which valence band is being plotted. Values of 1, 2 and 3 are used to represent the HH, LH and CH bands, respectively.
- **zseg_num** defines the z-segment number of the region being plotted. This value should match what is defined in the corresponding **z_structure** statement.
- **complex** is the number of the complex MQW region being plotted.

22.441 lplot_xyz

parameter	data type	values [defaults]
data_file	char	[vttek]
variable	char	(see list)
n_variables	charxn	
integration	char	[no]
math_oper	char	[void]
xy_point	realx2	
z_from	real	
z_to	real	
xrange	realx2	
yrange	realx2	
integration_start	real	[0.0](um)
integration_length	real	[1.e5](um)
qw_wave_ht	real	[0.1] (eV)
var_num	intg	[1]
mode_index	intg	[1]
trap_index	intg	[1]
qw_wave	intg	[0] 1
cond_subband	intg	[1] 2
val_subband	intg	[1] 2 3

lplot_xyz is a post-processor statement used to plot data along a 1D cut line. It is similar to **lplot_xy** and **plot_1d**. Depending on the 2D/3D nature of the simulation, the following rules should guide the choice of plotting command:

- 2D simulations: use **plot_1d**
- 3D cylindrical simulations with one mesh plane: use **plot_1d**
- xy cut of a 3D simulation: use **lplot_xy**
- z cut of a 3D simulation: use **lplot_xyz**

Parameters

Most parameters for this function are similar to the ones for **plot_1d** and are omitted for the sake of brevity.

- **xy_point** defines the position of the cut line in the xy plane. If the cut line is not parallel to the z axis, **xrange** and **yrange** may be used to define the extent of the cut line in the mesh plane.
- **z_from** and **z_to** define the extent of the 1D cut line along the z direction.

Examples

```
lplot_xyz variable=elec_conc xy_point=(2.5 0.4)  &&
  z_from=0.   z_to=50
```

22.442 makebend_dome

parameter	data type	values [defaults]
xy_center_1st_plane	realx2	[0. 0.] (μm)
height_1st_plane	real	[0.1] (μm)
diameter_1st_plane	real	[1.] (μm)
xy_center_last_plane	realx2	[0. 0.] (μm)
height_last_plane	real	[0.1] (μm)
diameter_last_plane	real	[1.] (μm)
zseg_num	intg	[1]

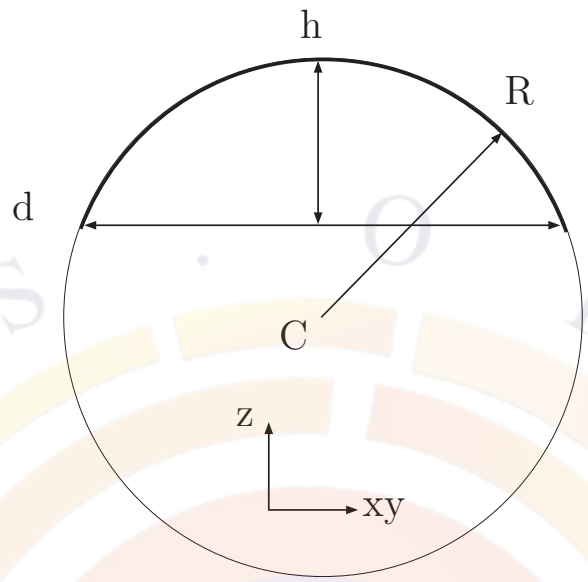


Figure 22.16: 2D cut of spherical dome bending

This statement is used to bend xy mesh planes in the z direction to form a spherical dome. The curvature of the first and last mesh planes in the segment are specified and the bending is interpolated for intermediate layers.

Parameters

The parameters for this statement can be more easily understood by examining Fig. 22.16. The spherical dome is shown by a 2D cut through the maximum bending area and the highlighted section of the circle represents the edge of the bent plane.

- **xy_center_1st_plane** are the in-plane coordinates of the center of curvature for the first plane in the segment.
- **height_1st_plane** is the maximum bending height on the first plane. A value of zero means no bending in the z direction.
- **diameter_1st_plane** is the diameter of a circle in the x-y plane formed by intersecting the dome and original unbent mesh plane. It is an indirect way of specifying the solid angle represented by the bent plane.
- **xy_center_last_plane** is the same as **xy_center_1st_plane** for the last mesh plane in the segment.

- **height_last_plane** is the same as **height_1st_plane** for the last mesh plane in the segment.
- **diameter_last_plane** is the same as **diameter_1st_plane** for the last mesh plane in the segment.
- **zseg_num** is the segment in which the bending is applied.

Examples

```
makebend_dome zseg_num=1 &&
  xy_center_1st_plane=(0.5 0.5) &&
  height_1st_plane = 0.0 &&
  diameter_1st_plane = 1.0 &&
  xy_center_last_plane=(0.5 0.5) &&
  height_last_plane = 0.2 &&
  diameter_last_plane = 1.0
```

22.443 makebend_rectangle_based_pyramid

parameter	data type	values [defaults]
xy_center_1st_plane	realx2	[0. 0.] (μm)
height_1st_plane	real	[0.1] (μm)
x_width_1st_plane	real	[1.] (μm)
y_width_1st_plane	real	[1.] (μm)
xy_center_last_plane	realx2	[0. 0.] (μm)
height_last_plane	real	[0.1] (μm)
x_width_last_plane	real	[1.] (μm)
y_width_last_plane	real	[1.] (μm)
zseg_num	intg	[1]

This statement is used to bend xy mesh planes in the z direction to form pyramids. The bending is specified for the first and last mesh planes in a segment and interpolated for intermediate planes.

Parameters

The parameters for this statement can be more easily understood by examining Fig. 22.17. The pyramid is shown in a projection view with h being the height of

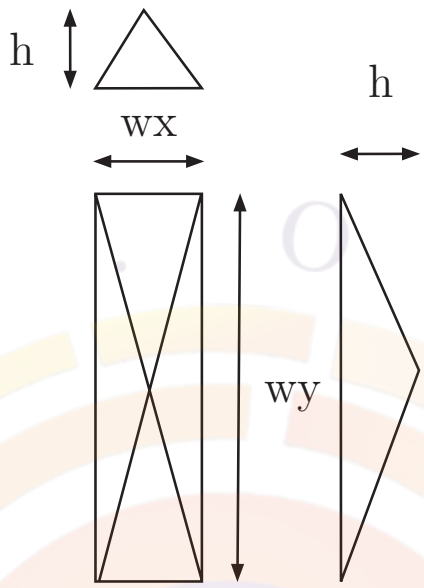


Figure 22.17: Projection view of pyramidal bending

the pyramid in the z direction and wx,wy are the widths of the pyramid base in the x and y direction.

- **xy_center_1st_plane** are the in-plane coordinates of the center of curvature for the first plane in the segment.
- **height_1st_plane** is the maximum bending height on the first plane. A value of zero means no bending in the z direction.
- **x_width_1st_plane** and **y_width_1st_plane** are the widths of the pyramid base in the x and y direction.
- **xy_center_last_plane** is the same as **xy_center_1st_plane** for the last mesh plane in the segment.
- **height_last_plane** is the same as **height_1st_plane** for the last mesh plane in the segment.
- **x_width_last_plane** and **y_width_last_plane** are the same as **x_width_1st_plane** and **y_width_1st_plane** for the last mesh plane in the segment.
- **zseg_num** is the segment in which the bending is applied.

Examples

```
makebend_rectangle_based_pyramid zseg_num=1 &&
```

```
xy_center_1st_plane=(0.5 0.5) &&
height_1st_plane = 0.0 &&
xy_center_last_plane=(0.5 0.5) &&
height_last_plane = 0.2 &&
```

22.444 makebend_tilt

parameter	data type	values [defaults]
fixed_xy_1st_plane	realx2	[0. 0.] (μm)
theta_1st_plane	real	[0.] (degrees)
phi_1st_plane	real	[0.] (degrees)
fixed_xy_last_plane	realx2	[0. 0.] (μm)
theta_last_plane	real	[0.] (degrees)
phi_last_plane	real	[0.] (degrees)
zseg_num	intg	[1]

This statement is used to tilt xy mesh planes in a particular direction. The direction of the plane normal vector is specified for the first and last planes in a segment and interpolated for intermediate planes.

Parameters

- **fixed_xy_1st_plane** is a point of the mesh plane that is on the tilt axis: it is not affected by the transformation.
- **theta_1st_plane** and **phi_1st_plane** are the angles describing the direction of the plane normal vector in the spherical coordinate system.
- **fixed_xy_last_plane** is the same as **fixed_xy_1st_plane** for the last mesh plane in the segment.
- **theta_last_plane** and **phi_last_plane** are the same as **theta_1st_plane** and **phi_1st_plane** for the last mesh plane in the segment.
- **zseg_num** is the segment in which the transformation is applied.

Examples

```
makebend_tilt zseg_num=1 &&
```

```
xy_center_1st_plane=(0.5 0.5) &&
phi_1st_plane = 0.0 &&
xy_center_last_plane=(0.5 0.5) &&
phi_1st_plane = 45.0
```

22.445 mass_density

parameter	data type	values [defaults]
(see) material_par		

The material statement **mass_density** is used to define the density (in kg/m^3) of the semiconductor material.

The use of this parameter and related examples are given under **material_par** in section 22.456.

22.446 mass_gamma_bar

parameter	data type	values [defaults]
(see) material_par		

The material statement **mass_gamma_bar** is an active layer macro statement used to define the relative effective mass of the Gamma band in the quantum barrier.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.447 mass_gamma_bulk

parameter	data type	values [defaults]
(see) material_par		

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

It is the conduction bane effective mass in direction along the c-axis. This statement is applicable for wurtzite structure bulk region [1].

[1] S. L. Chuang, “Optical Gain of Strained Wurtzite GaN Quantum Well Lasers", IEEE J. Quantum Electron., VOL. 32, NO. 10, OCTOBER 1996, p. 1791

22.448 mass_gamma_well

parameter	data type	values [defaults]
(see) material_par		

The material statement **mass_gamma_well** is an active layer macro statement used to define the relative effective mass of the Gamma band in the quantum well.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.449 mass_l_bar

parameter	data type	values [defaults]
(see) material_par		

The material statement **mass_l_bar** is an active layer macro statement used to define the relative effective mass of the L-band in the quantum barrier.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.450 mass_l_well

parameter	data type	values [defaults]
(see) material_par		

The material statement **mass_l_well** is an active layer macro statement used to define the relative effective mass of the L-band in the quantum well.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.451 mater_var

parameter	data type	values [defaults]
name	char	
variation	char	linear_x, linear_y
n_values	realxn	
rotation_angle	real	(degrees)
reference_point	realx2	(μm)
data_num	intg	

mater_var works in conjunction with the **load_macro** statement to define the spatial variation of a user-defined variable. The local value of this variable is passed as an argument to the various functions inside the macro to define material parameters.

In many ways, this statement duplicates some of the functionality common to all material parameter statements (c.f. Sec. 22.456). However, it operates at a higher level and is usually automatically generated by Crosslight helper tools like the layer.exe program.

Parameters

- **name** is a unique identifier used to identify the spatial variation. It should be the same as **var_name** in **load_macro** in order to correctly link the variation to the right symbol/macro variable.
- **variation** defines how the variable varies with distance:
  - *linear_x* means that the physical quantity linearly depends on the position in the x-direction. This must be used in conjunction with the parameters **data_num** and **n_values**.
  - *linear_y* is the same as *linear_x* except it is in the y-direction.

– *linear_xy* is a linear variation in an arbitrary direction specified by **rotation_angle** and **reference_point**.

– *function* means a mathematical function of the form:

```
function(variable1, variable2, ...)
......
end_function
```

All lines in a function except the last one must end with a semicolon (;) to suppress the line output. The output of the last line is the value returned by the function. The rules for mathematical functions are further explained in Appendix B.

– *table* is used for tabulated data. Multi-dimensional data may be used but a column-wise variation with the left-most index varying the quickest is needed as shown below:

```
table(x,y)
x(1)  y(1)  matrix(1,1)
x(2)  y(1)  matrix(2,1)
x(3)  y(1)  matrix(3,1)

x(1)  y(2)  matrix(1,2)
x(2)  y(2)  matrix(2,2)
x(3)  y(2)  matrix(3,2)
............
end_table
```

- **rotation_angle** is effective when **variation**=*linear_xy*. It is the angle from the y-axis.

- **reference_point** is (x,y) coordinate of the reference point of the rotated grading region. It is effective when **variation**=*linear_xy*. In listing of coordinate/composition in n_values above, the coordinate is the distance from this reference point for a rotated grading region.

- **n_values** specifies data points used for the linear interpolation when **variation**=*linear_x*. The format is `n\_values=(x1,f1,x2,f2,x3,f3, ...)` where x_i and f_i are the x-coordinate and function value of the ith point. A similar format is used with the other linear variation formats.

- **data_num** is the number of data values in **n_values**. It is equal to twice the number data points since coordinates must also be included in **n_values**.

Examples

```
mater_var name=In variation=linear_y 4_values=(0. 0.14, 0.3, 0.2) &&
 data_num=4
```

22.452 material

parameter	data type	values [defaults]
type	char	[semicond]
el_vel_model	char	constant, 2.piece,[beta],n.gaas
hole_vel_model	char	constant, 2.piece,[beta],n.gaas
trapleveli_model	char	gaussian, expo_tail
trapleveli_tail_side	char	conduction, valence
trapleveli_charge_type	char	donor, acceptor
band_valleys	realx2	[1 1]
mater	intg	[1]
group_index	real	[3.56]

material is a passive macro statement that defines various model parameters for a specific material.

Parameters

- **type** is the type of material. It takes one the following:
  - *semicond* refers to a normal semiconductor like Si or InP.
  - *insulator* is a special material type where the current is exactly zero.
  - *resistor* is a special material type used for metals. See Sec. 5.3.
  - *wurtzite* refers to semiconductors like GaN and ZnO which have the wurtzite crystal structure.
  - *organic* refers to organic semiconductors.
  - *super_struc_type* is a complicated layer structure which is usually simulated separately in a smaller sub-project.
- **mater** is the index of the material being described. This is omitted when the statement is used in a macro file.
- **el_vel_model** and **hole_vel_model** are the field-dependent mobility models from Sec. 5.1.6. Each parameter can take on the following:

- 1. *constant*
- 2. *2.piece*
- 3. *beta*. This is the default for most devices.
- 4. *n.gaas*. This model shows the negative differential resistance typical of GaAs and applies to electrons only.
- 5. *poole_frenkel*. This model is used for organic semiconductors.
- 6. *modified_transf_elec*. This corresponds to a mixture of the "beta" or Canali model and a modified transferred-electron model: it applies to electrons only. See **beta_mte** for details.

A user-defined mobility model can also be used.

- **band_valleys** is the number of equivalent conduction band valleys and valence band peaks. This is needed to compute the density of states from the effective masses.
- **group_index** is the group index of the 2D cross section being simulated. The group index is used to measure the velocity of the photon energy and it appears in the photon rate equation and in stimulated emission terms.
- **trapleveli_model**, **trapleveli_tail_side** and **trapleveli_charge_type** are a set of parameters defining traps that are always a part of the material being defined in a particular macro. This provides an alternate means of defining traps in the software besides the use of the **doping** statement. Refer to Sec. 5.5 for the rules governing traps when both sets of statements are used.

  The significance of these parameters and the values allowed is the same as for the similarly-named parameters in the **doping** statement. Note that "i" is an identifying label (i=1..9) so different traps species can coexist in the same material.

Examples

```
material band_valleys=(6 1) &&
  el_vel_model=beta hole_vel_model=beta
```

This is used for Si.

```
material type=semicond band_valleys=(1 1) &&
  el_vel_model=n.gaas hole_vel_model=beta
```

This is used for GaAs.

22.453 material_3d

parameter	data type	values [defaults]
global	char	[yes], no
intern_spon_par	char	[yes], no
spon_par	real	[1.7]
dkdn_real	real	[0.] (m^2)
dkdn_imag	real	[0.] (m^2)
dkdn_nref	real	[2.e24] (m^{-3})
dndn_type2	real	[-1.e-28] (m^3)
dgdn_type2	real	[1.e-22] (m^2)
dgdw_type2	real	[0.0]
dkdj_real	real	[0.] (m/A)
dkdj_imag	real	[0.] (m/A)
dkdj_jref	real	[0.] (A/m^2)
sec_num	intg	[1]

material_3d is primarily used in PICS3D in conjunction with **passive_3d** to describe how the real and imaginary parts of the propagation constants vary in unmeshed passive regions. Other parameters relating to the Green's function analysis are also set here so this command may also be used to override internal defaults for that model.

Parameters

- **global** determines whether this statement applies to all optical sections. If not, only the section defined in **sec_num** is affected.
- **dkdn_real** and **dkdn_imag** are the derivatives of the real and imaginary parts of the propagation constant with respect to the carrier density; **dkdn_nref** is the reference carrier density. **dkdj_real**, **dkdj_imag** and **dkdj_jref** implement a similar model using a reference current. These values are used in the longitudinal transfer matrix code.
- **dndn_type2**, **dgdn_type2** and **dgdw_type2** are derivatives of the refractive index and optical gain with respect to the carrier density (n) and frequency (w). These values are used in the small-signal and noise analysis.
- **intern_spon_par** determines whether the internally calculated value of the population inversion factor n_{sp} (as defined in Eq. D.124) is used in the Green's

function model. If **global**=*no*, this setting must be the same in all sections of the device.

If **intern_spon_par**=*no*, a hard-coded value defined in **spon_par** is used in all integrals that depend on gn_{sp}; typical values for AlGaAs and InGaAsP lasers are 2.6 and 1.6, respectively. If **intern_spon_par**=*yes*, n_{sp} is obtained from the gain value at each bias step and Eq. D.124.

From the round-trip gain formulation of Green's function theory, we know that the imaginary component of the optical frequency represents the contribution of the spontaneous emission to the propagation so that changes in n_{sp} will be reflected in that value. Such changes will thus affect multiple aspects of the model including the emission spectra, noise and linewidth analysis, the shape of the L-I curve near threshold, the photon number in each mode, etc...

We note that this term is a key component of the Green's function model and is not always fully understood: in the model, the optical gain, the photon density of states and the amount of spontaneous emission coupling into the longitudinal modes are all interconnected; the latter term includes both lateral coupling of the emission due to the waveguide shape/index profile and longitudinal coupling due to shape of the longitudinal mode solutions. While a quantum mechanical relationship exists based on material gain properties (c.f. Chap D), we must also consider how optical losses in the cavity affect the population inversion in the device under operating conditions.

Additionally, if **intern_spon_par**=*no* then the integrated spontaneous emission rate in each mode is also replaced by the averaged value over each mode; this is done for historical reasons and to support regions defined using **passive_3d** statements. As can be seen in Sec. E, this modification can have strong effects on the shape of the emission spectrum and the definition of the side mode suppression ratio. For this reason, the default setting of **intern_spon_par**=*yes* is ***strongly*** recommended: numerical testing shows the spectrum shape does not match experiment if **intern_spon_par**=*no*.

22.454 material_label_define

parameter	data type	values [defaults]
label	char	
mater	intg	[1]

material_label_define is used for import/export purposes with CSUPREM and assigns a specific user-defined label to a material number.

Parameters

- **label** is the user-defined label.
- **mater** is the material number associated with that label.

22.455 material_lib

parameter	data type	values [defaults]
name	char	
var_namei(i=1..9)	char	
var_symboli(i=1..9)	char	
type	char	[basic], complex
vari(i=1..9)	real	
mater	intg	[1]

material_lib is the equivalent of the **load_macro** statement for the "library" material system described in Sec. 3.5. It associates a certain library name and its associated parameter with a material number. This statement is usually automatically generated by layer.exe.

See **complex_var_symbol** for further information.

Parameters

- **name** is the name of the library.
- **var_symboli**, with i=1..9, is the variable name (.e.g. "x" for composition) of variable #i. This symbol must be associated with the value of the variable in **vari** or otherwise link with a grading function name in **var_namei**.
- **type** determines whether this library will only use the passive part of the underlying macro (*simple*) or also invoke the complex MQW active macro (*complex*). In general, this parameter is automatically set by the layer.exe program for those layers tagged as being part of a quantum-confined region.
- **mater** is the material number being linked to this material library.

Examples

```
material_lib name=AlGaAs mater=   1 &&
  var_symbol1=x  var1=  0.7100E+00
```

22.456 material_par

parameter	data type	values [defaults]
variation	char	linear_x,linear_y,function,[constant]
var_namej (j=1..9)	char	[void]
varj(j=1..9)	real	
var_symbolj(j=1..9)	char	void
mater_label	char	void
value	real	
n_values	realxn	
mater	intg	[1]
data_num	intg	[0]
var_num	intg	[1]

material_par is not a statement that can be processed by the software. It is simply a stand-in name for all the material macro statements that use the same set of parameters. For example, the parameters of the table above apply equally to **band_gap** and **hole_mass**. The parameters described here are also collectively referred to as *group1* parameters which are common to most physical quantities of the simulation program.

The statements in *group1* are most often found in material macro files but can also be used in other input files to manually define parameters or override default macro values.

Parameters

- **variation** defines how the physical quantity varies with position. There are several options for this parameter:
  - *constant* means that a fixed value is used everywhere in the material to define a physical quantity.
  - *linear_x* means that the physical quantity linearly depends on the position in the x-direction. This must be used in conjunction with the parameters **data_num** and **n_values**.

– *linear_y* is the same as *linear_x* except it is in the y-direction.

– *function* means a mathematical function of the form:

```
function(variable1, variable2, ...)
......
end_function
```

All lines in a function except the last one must end with a semicolon (;) to suppress the line output. The output of the last line is the value returned by the function. The rules for mathematical functions are further explained in Appendix B.

– *table* is used for tabulated data. Multi-dimensional data may be used but a column-wise variation with the left-most index varying the quickest is needed as shown below:

```
table(x,y)
x(1)  y(1)  matrix(1,1)
x(2)  y(1)  matrix(2,1)
x(3)  y(1)  matrix(3,1)

x(1)  y(2)  matrix(1,2)
x(2)  y(2)  matrix(2,2)
x(3)  y(2)  matrix(3,2)
............
end_table
```

- **var_name1-9** is used when **variation**=*function* and the user-defined variable varies in space according to the matching definition in **mater_var**. The local mesh value of the variable is passed to the macro functions as an argument. This parameter must be paired with **var_symbol1-9** to define the symbol of the function argument in free-style macros.

  This is often used to define composition gradings.

- **var1-9** is used when **variation**=*function* and the user-defined variable is a constant. This constant is passed to the macro functions as an argument. This parameter must be paired with **var_symbol1-9** to define the symbol of the function argument in free-style macros.

- **var_symbol1-9** is the symbolic variable name used as function argument in the macro. If symbols are defined by the user, then they must exactly match the function arguments in the macro. If they are omitted, then the software assumes the older fixed-form is being used for the macro and the order of

the parameters in **var1-9** or **var_name1-9** must exactly match that of the function arguments in the macro.

- **mater_label** is a user-defined label that can be used to refer to this material in other commands.
- **value** is the value of the statement or physical quantity if **variation**=*constant*.
- **n_values** specifies data points used for the linear interpolation when **variation**=*linear_x*. The format is `n\_values=(x1,f1,x2,f2,x3,f3, ...)` where x_i and f_i are the x-coordinate and function value of the ith point. A similar format is used when **variation**=*linear_y*.
- **mater** is the material number of the material being described. This is omitted within the scope of a macro file but must be specified when overriding default macro values elsewhere in the simulation.
- **data_num** is the number of data values in **n_values**. It is equal to twice the number data points since coordinates must also be included in **n_values**.
- **var_num** is the number of function arguments used when **variation**=*function*. it is obsolete for newer free-style macros.

Examples

```
band_gap  value=1.425 mater=2

dielectric_constant variation=linear_y data_num=6  &&
 6_values=(0. 10.5, 0.5 11.4, 0.9 12.)

electron_mass variation=function var1=0.3
function(x)
0.4&+0.2&*x
end_function
```

22.457 max_electron_mob

See **electron_mobility**

22.458 max_hole_mob

See **hole_mobility**.

22.459 mesh_output

parameter	data type	values [defaults]
mesh_outf	char	
order	char	[yes],no

The mesh generation statement **mesh_output** defines the mesh output data files. This statement is used after the placement of mesh lines on the edges of the polygons.

Parameters

- **mesh_outf** is the mesh output file.
- **order** specifies the format of the mesh output. If the mesh output is to be used by the solver, *yes* should be chosen; that is, ordering of the mesh is required to interface with the solver. If the mesh is to be further manipulated after this statement, a *no* value should be chosen.

Examples

```
mesh_output mesh_outf=bulk2d.msh  order=yes
```

22.460 microcavity_model

parameter	data type	values [defaults]
fdfd_vectorial	char	[yes],no
set_wavelength	real	(μm)

microcavity_model solves the wave equation directly (i.e. without doing a separation of variables) using an eigenvalue method.

In order to force the outgoing wave (which normally extends to infinity) to decay to zero within a fixed simulation domain, Perfectly Matched Layer boundaries (PML) are applied to the output facet using the **microcavity_exit** statement. The output power vs. bias is then obtained by integrating the power loss in the PML region: this value is equal to the power which would have travelled to infinity if a free space region had been used instead of the PML.

As of v.2016, this model is available for Fabry-Perot-like 2D edge emitters and VCSELs, with a particular focus on surface relief applications. As of v.2017, a new vectorial mode solver is available for VCSELs and is used as the new default; edge emitting devices should manually enable the previous scalar mode solver.

Parameters

- **fdfd_vectorial** enables a finite difference frequency domain (FDFD) vectorial mode solver for the microcavity model. The FDFD mesh is rectangular and uniformly-spaced in any given direction. The spacing of this mesh is obtained from the smallest of the following criteria:
  - the minimum mesh spacing in a given direction
  - the Courant stability condition typically used in the related field of finite difference time domain (FDTD) simulations ($\frac{\lambda}{10n}$)
  - an internally-set minimum of 5 points in the radial direction and 100 points in the vertical/propagation direction

  Material properties and mode profiles are interpolated to/from the irregular FEM mesh used for the drift-diffusion model and the FDFD grid.
- **set_wavelength** sets the wavelength used to solve the wave equation.

22.461 microcavity_exit

parameter	data type	values [defaults]
below_y	real	(μm)
above_y	real	(μm)
left_x	real	(μm)
right_x	real	(μm)
power_refl	real	[0.0]

---

microcavity_exit sets up the position of PML boundary conditions used for the **microcavity_model** command.

Parameters

- **below_y** sets the position of the bottom horizontal boundary. If omitted, no PML boundary is set on that side of the device.
- **above_y** sets the position of the top horizontal boundary. If omitted, no PML boundary is set on that side of the device.
- **left_x** sets the position of the left vertical boundary. If omitted, no PML boundary is set on that side of the device.
- **right_x** sets the position of the right vertical boundary. If omitted, no PML boundary is set on that side of the device.
- **power_refl** is a power reflection coefficient applied to the wave before hitting the PML boundary.

22.462 min_electron_mob

See **electron_mobility**

22.463 min_hole_mob

See **hole_mobility**.

22.464 minispice

parameter	data type	values [defaults]
circuit_file	char	
spice_device_to_tcadmesh	char	
spice_source_to_tcadbias	char	
scan_variable	char	
contactj_to_spice_node (j=1..109)	char	
spice_source_node_scan	char	
z_dim	real	[10] (μm)
spice_node_scan	intg	

minispice is new to the 2013 version of Crosslight and enables a full-fledged mixed-mode (SPICE + TCAD) simulation. It replaces and enhaces the **external_cir** statement used to define external circuits in previous versions.

This statement may be used in one of two ways: to define a sub-circuit attached to a particular contact or to substitute one or more SPICE devices in a larger circuit with TCAD devices. The first mode replicates the functionality of **external_cir** and may be used multiple times to attach different sub-circuits to different electrodes.

The second approach is more typical of mixed-mode simulation and may be used to model a larger integrated circuit. In this mode, the TCAD device is defined inside the SPICE circuit file using a placeholder element, with the Drift-Diffusion model replacing the usual SPICE compact model of the device during the full mixed-mode simulation. This second approach is more powerful and should be used for all new simulations.

Note that if multiple devices in the same circuit must be replaced by TCAD devices, then **more_tcadmesh** must be used to define the extra placeholder elements in the circuit file.

As of the 2014 version, transient simulations and DC parameter sweeps are supported in mixed-mode simulations. Small-signal AC modeling is new to the 2015 version of the software but still under development. See **ac_voltage** for examples of the currently-available functionality.

Parameters

- **circuit_file** is the name of the SPICE layout file (.cir) defining circuit (or sub-circuit) for this command. For those users not familiar with SPICE layout rules, a convenient manual may be found at `http://www.freeda.org/doc/SPICE/spice.pdf`.

  If the sub-circuit mode of **minispice** is used, then multiple circuit files may be defined (one for each contact) by repeatedly using this command. In that case, node numbers may be repeated in each layout file as they are independent from each other.

  Please note that following normal SPICE convention, node zero (0) of the layout should be the circuit's electrical ground; however, this convention may be ignored in the sub-circuit mode of **minispice**.

- **spice_device_to_tcadmesh**, is the name of a circuit entity in the above layout file which is replaced by the TCAD meshed device. As an example, a simple diode or MOSFET model from a standard library can be used as a placeholder to draw the circuit layout with a third-party GUI tool. The actual mixed-mode simulation will then be performed using the TCAD device's

own geometry, doping profiles, trap settings, etc... rather than the model parameters from the SPICE library.

This parameter is not used in the sub-circuit mode of **minispice**.

- **contactj_to_spice_node** associates the mesh boundary #j (j=1..109) (i.e. the contact boundary for the TCAD device) with a given SPICE node label in the circuit layout file. If the sub-circuit method is used, then only one such statement may be used in each **minispice** command, in which case it associates a SPICE sub-circuit with a particular electrode.

  If **spice_device_to_tcadmesh** is used though, then each mesh electrode/contact number that connects with a SPICE element must be associated with its equivalent SPICE node label.

- **spice_node_scan** is used to define where the scan bias control variable (e.g. *voltage_1*) is applied in the SPICE circuit or sub-circuit. An illustration of this can be found in the examples below.

  Note that the control variables always follow the usual rules of the **scan** statement: at equilibrium, the voltage and net current are all zero. Afterwards, the bias voltage stays constant and always inherits the value of the previous scan unless it is explicitly scanned or a current boundary is used.

  Please note that although the control variables from the **scan** statement are assigned to a particular SPICE node, the software will usually print the bias values for the internal bias point (electrode) in the simulation log file.

- **scan_variable** may be used to associate the electrode number of the scan control variable with the node number of **spice_node_scan**. For example, specifying **scan_variable**=*voltage_1* means that the voltage applied at **spice_node_scan** is V_1.

  However, this approach should be considered obsolete and **contactj_to_spice_node** is the preferred method of assigning a sub-circuit to a particular electrode.

- As an alternative to directly driving the SPICE sub-circuit with APSYS bias controls, **spice_source_to_tcadbias** may be used to drive an existing voltage or current source in circuit layout file using a transient simulation or a DC parameter sweep; in the **scan** statement, the corresponding variables for this are *time* and *virtual_time*, respectively. In this case, the SPICE source connected at the node label defined in **spice_source_node_scan** is scanned.

- **zdim** is the omitted dimension in the z-direction used in 2D simulations to convert units (e.g. Amperes to A/m). It is not used in 3D simulations.

Sub-Circuit Example: Series Resistor

Consider a simple diode with a parasitic 100 Ohm resistance connected in series (res.cir):

```
#comment line
r 0 1 100
```

The simulation file for the diode invokes this external circuit using the following commands:

```
minispice circuit_file=resistor.cir &&
contact2_to_spice_node=0 spice_node_scan=1

....

scan var=voltage_1 value_to=-0.7 &&
  init_step=0.01 min_step=1.e-5 max_step=0.02
```

Here the sub-circuit (resistor) is connected to electrode #2 of the device. Inside the sub-circuit, SPICE node 0 is connected to the device electrode while SPICE node 1 is used to apply the bias voltage *voltage_2*. Since the scan statement applies to electrode #1, a value of -0.7V is applied directly to the device since there is no sub-circuit on that side of the device. On the "+" side of the device, *voltage_2* stays at 0V throughout the simulation since it is never scanned; the internal voltage of electrode #2 is left floating and depends on the current flow in the resistor.

Sub-Circuit Example: Multiple Series Resistors

```
minispice circuit_file=res1.cir &&
contact2_to_spice_node=0 spice_node_scan=1

minispice circuit_file=res2.cir &&
contact1_to_spice_node=0  spice_node_scan=1

.....

scan var=voltage_1 init_step=0.01 value_to=-3. min_step=1.e-5
```

Here two sub-circuits are defined and are attached to electrodes #2 and #1, respectively, where each of the two circuit files is similar to the one in the previous section. As can be seen in Fig. 22.18 within each layout file, the electrode is connected to

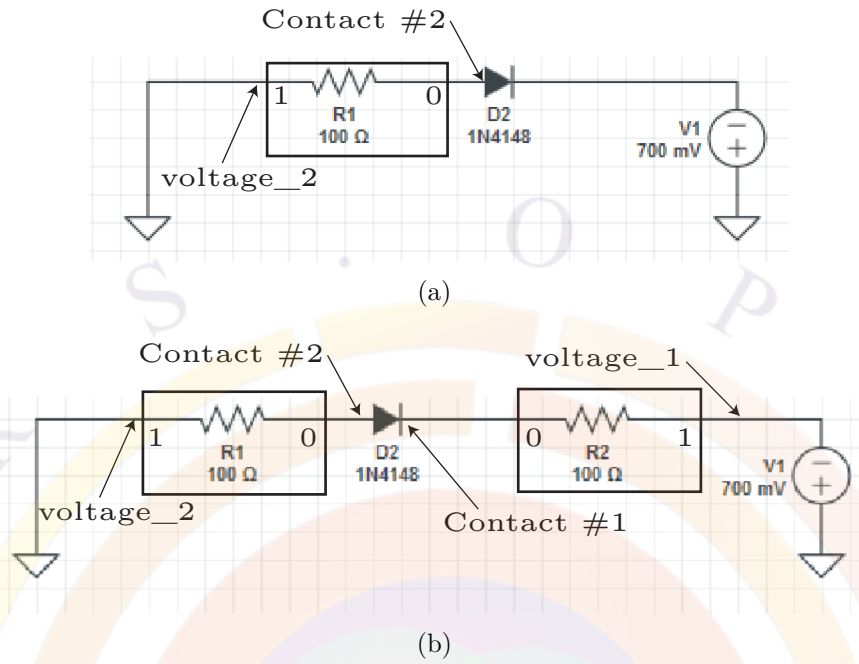


Figure 22.18: Sub-circuits attached to one (a) or two (b) contacts. SPICE node numbers in each sub-circuit are reused

the local SPICE node #0 and the scan bias variables (*voltage_1* and *voltage_2*) are applied to local SPICE node #1. As before, scan variable *voltage_2* stays at 0V throughout the simulation; however, the voltage of both electrodes is now floating.

Note that different resistor values may be used in each circuit file and the sub-circuits may also have completely different elements.

Sub-Circuit Example: Parallel Resistor

This is a trick question[2]: it is not possible to use a sub-circuit approach to represent a resistor or capacitor connected in parallel with a TCAD device. By ***definition***, a sub-circuit is only attached to one electrode and a parallel resistor would connect two different electrodes. Resistors and other SPICE devices may be connected in parallel but only within the same sub-circuit.

To model this kind of situation, a complete layout file for the whole circuit must be used along with **spice_device_to_tcadmesh**. The next example describes a more complex version of the same concept.

[2]Previous version of the manual was in error

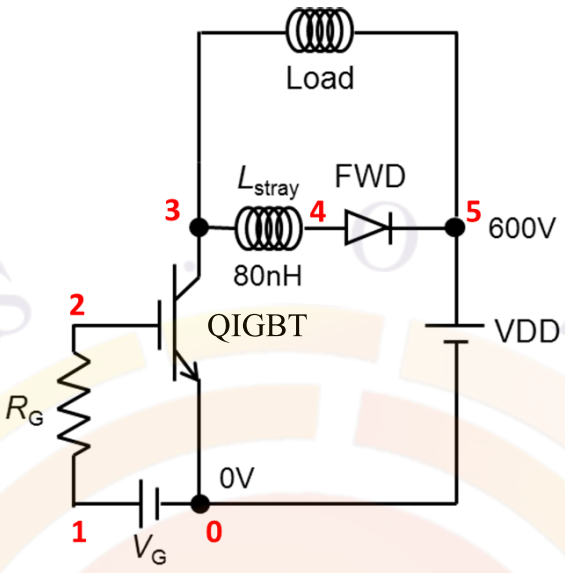


Figure 22.19: Circuit layout for a IGBT switching test circuit

Integrated Circuit Example: External Bias

Let us consider the circuit shown in Fig. 22.19 and defined in the file "IGBT_switching.cir" below:

```
# A switching test circuit
VG 1  0  pulse(0 15 1e-6 1e-9 1e-9 0.5e-6 1e-6)
RG 1  2  40
Zigbt  3  2  0  IGBT
Lstray 3  4  0.02u
Dfwd 4  5  FWD 1e-5
Lload 3  5 0.5u
VDD 5  0  100
.MODEL FWD D(AF=1 BV=1200 CJO=0. EG=1.11
+ FC=0. IBV=1.E-10 IS=1.E-14
+ KF=0 M=0.5 N=1 RS=1e-10 TT=1e-6
+ VJ=1.0 XTI=3.00E+00)
.TRAN 5n 2.5u
.PROBE
.END
```

In this setup, we replace the IGBT device with our meshed TCAD device. We also note that this circuit contains an external transient source term (Vg) in addition to a DC bias (VDD). By using the **minispice** and **scan** statements below, we can

control both of these sources in the simulator: the normal scan variables (*voltage_1*) are not used.

```
minispice circuit_file=IGBT_switching.cir z_dim=1e6 &&
spice_device_to_tcadmesh=zigbt &&
contact1_to_spice_node=0 &&
contact2_to_spice_node=2 &&
contact3_to_spice_node=3

....

scan var=virtual_time value_to=1 init_step=0.01 min_step=1e-5 &&
max_step=0.1

...

scan var=time value_to=1.0e-6 init_step=1e-9 min_step=1e-14 &&
max_step=1e-7
```

During the simulation, *virtual_time* controls all of the DC sources simultaneously: it acts a scaling factor for the value defined in the SPICE layout with a initial value of zero to match the equilibrium conditions required for TCAD modeling. After this initial biasing, the transient pulse source Vg is activated by increasing the wall clock of the simulation (*time*).

22.465 mmb_gaintable

parameter	data type	values [defaults]
directory	char	[void]
use_gain_spec	char	[yes]
use_index_spec	char	[yes]
use_spon_spec	char	[yes]
active_regnum	intg	[1]

The statement **mmb_gaintable** is used to activate an interface to import the microscopic many-body material data base of Univ. of Arizona (or Non-linear Control Strategy Inc.). The data base consists of quantum well optical gain, refractive index and spontaneous emission computed at a limited number of temperatures and carrier densities. The assumption in the data base is that there is equal amount of electrons and holes in any quantum wells. This statement simply imports the gain and

index data from the mmb-data base and interpolates between data files to obtain a continuous gain spectrum useful for simulation. To make the simulation compatible, we have to average the amount of electrons and holes in the well to come up with a single value of density to compare with the data files in the mmb-data base.

- **directory** is the name of the directory where the data files of mmb-data is stored. This directory must be located in the current directory of the input files of the simulation.

- **use_gain_spec** instructs the simulator to import the gain spectrum data from the mmb-data.

- **use_index_spec** instructs the simulator to import the refractive index spectrum data from the mmb-data.

- **use_index_spec** instructs the simulator to import the spontaneous emission spectrum data from the mmb-data.

- **active_regnum** is the active region number this statement takes effect.

Example(s)

```
mmb_gaintable directory=mmb_data use_gain_spec=yes use_index_spec=yes
```

22.466 mobility_xy

parameter	data type	values [defaults]
dir	char	[y],x
elec_field_model	char	[void],intel1,intel2,lombardi
hole_field_model	char	[void],intel1,intel2,lombardi
use_elec_hv_expo_factor	char	[no]
use_hole_hv_expo_factor	char	[no]
mater_label	char	
factor_elec	real	[1.]
factor_hole	real	[1.]
elec_intel1_beta	real	[0.5]
elec_intel1_e_crit	real	[4.2e6] (V/m)
elec_intel2_alpha	real	[1.02]
elec_intel2_e_univ	real	[5.71e7] (V/m)
hole_intel1_beta	real	[0.5]
hole_intel1_e_crit	real	[3.e6] (V/m)
hole_intel2_alpha	real	[0.95]
hole_intel2_e_univ	real	[2.57e7] (V/m)
elec_lombardi_b	real	[4.75e7] (cgs unit)
elec_lombardi_alpha	real	[1.74e5] (cgs)
elec_lombardi_beta	real	[0.125]
elec_lombardi_delta	real	[5.82e14] (cgs)
hole_lombardi_b	real	[9.93e7] (cgs)
hole_lombardi_alpha	real	[8.84e5] (cgs)
hole_lombardi_beta	real	[0.0317]
hole_lombardi_delta	real	[2.05e14] (cgs)
elec_hv_critical_field	real	[5.e7] (V/m)
elec_hv_expo_field	real	[1.e7] (V/m)
elec_hv_minfac	real	[0.5]
hole_hv_critical_field	real	[5.e7] (V/m)
hole_hv_expo_field	real	[1.e7] (V/m)
hole_hv_minfac	real	[0.5]
mater	intg	[1]

This statement defines an anisotropic mobility on the x-y plane. This may be useful for an effective medium approximation of multiple layers (such as DBR mirrors in a VCSEL) or for vertical field-dependent mobility in a MOSFET channel.

For mobility reduction due to transverse field under the MOSFET channel, we have implemented the following mobility models:

- *intel1* is based on the PISCES-2ET manual[131] and defines a factor to reduce the mobility under the channel:

$$r_{perp} = (1 + E_{\perp}/E_{crit})^{-\beta} \tag{22.76}$$

  where $E_{\perp}$ is the field perpendicular to the SiO2/Si interface.

- *intel2* is the same as above with the following formula for the reduction factor:

$$r_{perp} = [1 + (E_{\perp}/E_{univ})^{\alpha}]^{-1} \tag{22.77}$$

- *lombardi* sums up the contributions to mobility from different sources as follows[132]:

$$\frac{1}{\mu} = \frac{1}{\mu_{ac}} + \frac{1}{\mu_{srf}} + \frac{1}{\mu_0} \tag{22.78}$$

  where μ_0 is the mobility due to longitudinal field/hot-carrier effect.

  The other terms are as follows:

$$\mu_{ac} = \frac{B}{E_{\perp}} + \frac{\alpha N^{\beta}}{T_L E_{\perp}^{\frac{1}{3}}} \tag{22.79}$$

$$\mu_{srf} = \frac{\delta}{E_{\perp}^2} \tag{22.80}$$

Optionally, an additional mobility reduction factor of the following form is used:

$$f_{hv} = \begin{cases} 1 & \text{if } E_{\perp} \le E_{cr} \\ f_{min} + (1 - f_{min}) e^{-(E_{\perp} - E_{cr})/E_{exp}} & \text{if } E_{\perp} > E_{cr} \end{cases} \tag{22.81}$$

This additional reduction is used to describe quasi-saturation effect in high voltage (HV) MOSFET where strong suppression of mobility due to high vertical field (or high Vg) has been reported.

Parameters

In the following description, we only define parameters for electrons; those for holes are completely similar.

- **dir** is the direction in which the mobility is modified by a factor.
- **elec_field_model** specifies the model used for electron mobility. If set to void, a constant factor is used to modify the default mobility.

- **factor_elec** is a constant scaling factor applied to the electron mobility.
- **elec_intel1_beta**, **elec_intel1_e_crit** are parameters for the *intel1* model described above.
- **elec_intel2_alpha**, **elec_intel2_e_univ** are parameters for the *intel2* model described above.
- **elec_lombardi_b**, **elec_lombardi_alpha**, **elec_lombardi_beta**, **elec_lombardi_delta** are parameters for the Lombardi model described above. Please note that to keep the same formulas as in the original paper, we use CGS units for these values.
- **use_elec_hv_expo_factor** enables the additional mobility reduction factor f_{hv}. **elec_hv_critical_field**,**elec_hv_expo_field** and **elec_hv_minfac** are the E_{cr}, E_{exp} and f_{min} values for this model.
- **mater** is the number of the material affected by this statement. If a label has previously been defined, **mater_label** may be used instead.

Examples

```
mobility_xy dir=y factor_elec=0.6 mater=3
```

The above statement decreases the electron mobility in y-direction by a factor of 0.6 in material number 3.

22.467 mode_srch

parameter	data type	values [defaults]
adjust_range	char	[no], yes
force_yrange	char	[no], yes
omega_xrange	real	[8.]
omega_yrange	real	[4.]
omega_ymin	real	[1.e3]
wavel_xrange	real	(m)
sample_x	intg	[391]
sample_y	intg	[101]
cavity_num	intg	[1]

mode_srch sets parameters relating to the longitudinal mode search in PICS3D.

Parameters

- **adjust_range** turns on/off the option to search for the position of the dominant mode; when turned on, the program will attempt to put the dominant mode in the middle of the frequency range. This search is done prior to the onset of the coupling; once photon coupling is turned on, the same set of longitudinal modes will be used throughout the simulation.

- **omega_xrange** defines a range of optical frequencies ($\Re\omega$) for the longitudinal mode search and is expressed as a multiple of the Fabry-Perot mode spacing for this cavity ($\frac{2\pi c}{nL}$). This range is defined around the frequency corresponding to the reference wavelength defined in the **longitudinal** statement.

- **omega_yrange** is similar to **omega_xrange** but defines the maximum value for the imaginary part of the frequency, as defined in Sec. 16.4. This parameter must be limited since with enough additional spontaneous emission, any wavelength can be made to satisfy the round-trip gain equation: we must limit the analysis to mode of importance with sufficient cavity feedback.

- **wavel_xrange** (in unit of meter) is the wavelength search range for the longitudinal modes. If this is defined, it will override the search range define by **omega_xrange** but the latter should still be used to allocate sufficient storage for the mode search.

- **force_yrange** is used to force the y-range ($\Im\omega$) for mode search on the complex omega plane. If this is not forced, the range used in the mode search is determined by a quick search on both ends of the x-range. This option may be used to ensure all modes are found within a certain y-range.

- **omega_ymin** is used so that $\Im\omega$ never dips below a certain minimum amount. Since this quantity is related to the amount of additional power provided by the spontaneous emission, it has a tendency to go to zero as the mode starts to lase and the photon number goes up. However, a certain amount of spontaneous emission is always present, even above threshold.

- **sample_x** and **sample_y** is the number of sample points in the complex frequency plane that are used for the mode search.

- **cavity_num** can be used to specify different longitudinal settings when working with multiple optical cavities. See also **begin_cavity**.

Examples

```
mode_srch omega_xrange = 20
```

22.468 modify_bias_output

parameter	data type	values [defaults]
radiative_qw_only	char	[yes],no

modify_bias_output changes the way certain bias-dependent variables are computed.

Parameters

- When **radiative_qw_only**=*yes*, only radiative emission from the QW regions contributes to the internal quantum efficiency (IQE) calculations.

22.469 modify_bulk_macro

parameter	data type	values [defaults]
variable	char	
mater_label	char	
add_to_value	real	[0.0]
scale_value	real	[1.0]
mater	intg	[1]

modify_bulk_macro provides a convenient way of modifying an existing or previously defined bulk macro variable.

Parameters

- **variable** is the macro variable being altered by this statement.
- **add_to_value** adds a certain value to the specified variable.
- **scale_value** multiplies the specified variable by a certain value.
- **mater** is the material number where the variable should be modified. If a label has previously been defined as an alias, **mater_label** may be used instead.

Examples

```
modify_bulk_macro variable=affinity add_to_value=0.2
```

The above command would shift the affinity, and thus the band alignment, by 0.2 eV.

22.470 modify_gain

parameter	data type	values [defaults]
positive_only	char	yes,[no]
add_bulk_absorption_to_qw	char	yes,[no]
add_bulk_abs_spectrum	char	yes,[no]
enforce_index_spectrum	char	[no],yes
bulk_abs_eg	real	[1.] (eV)
bulk_abs_wavelength	real	(μm)
scale_bulk_abs	real	[1.]

modify_gain is used to modify the gain spectrum that is computed by the software.

Parameters

- **positive_only** truncates the negative part of the gain curve.
- **add_bulk_absorption_to_qw** can be used to add an imported gain or absorption spectrum to the gain curve. This can important in QW or QDOT solar cell applications which require accurate broad-band absorption, including the inter-valley transitions. Since the internal gain model in Crosslight is only valid for interband transitions near the Γ point, experimental absorption data can be imported to bridge the gap between the two spectral domains.

  When using this parameter, it is assumed that spectrum data has been imported using the **index_spectrum** statement or that **absorption** has been set to a non-zero value.
- **add_bulk_abs_spectrum** is used in conjunction with **add_bulk_absorption_to_qw**. When used, the bulk absorption is added to the full spectrum and not merely to the local values (at each mesh point).

- **enforce_index_spectrum** instructs the software to ignore the default internal gain spectrum calculations and completely use the imported data from **index_spectrum** for the gain curve of this active region. Note that for passive regions, merely using **index_spectrum** already forces the software to use the specified absorption data: this parameter is only necessary for cases where a region is declared as an active material for other reasons, such as enabling photon recycling models.

- **bulk_abs_eg** is the bandgap of the bulk material, used for cut-off of the absorption. **bulk_abs_wavelength** may be used to define this using the wavelength instead but only one of these parameters should be used.

- **scale_bulk_abs** may be used to artificially scale the bulk absorption spectrum before adding it to the QW. This can be used to account for uncertainties in the bulk effective mass.

Examples

```
modify_gain add_bulk_absorption_to_qw=yes
```

22.471 modify_layer_height

parameter	data type	values [defaults]
curve_data_file	char	
location	char	top, [middle], bottom
bottom_shaping	char	[uniform], gradual
top_shaping	char	[uniform], gradual
delta_y_from_bottom	real	(um)
delta_y_from_top	real	(um)
relative_yrange	realx2	[-1.e9 1.e9] (um)

modify_layer_height is used to modify the height of one or more layers when defining the device structure in the .layer file. This is shown schematically in Fig. 22.20.

A reference point (y_{ref}) must be defined where the layer height is left unchanged: this is done using the **location** parameter for the y-position and **layer_height_ref** for the x-position.

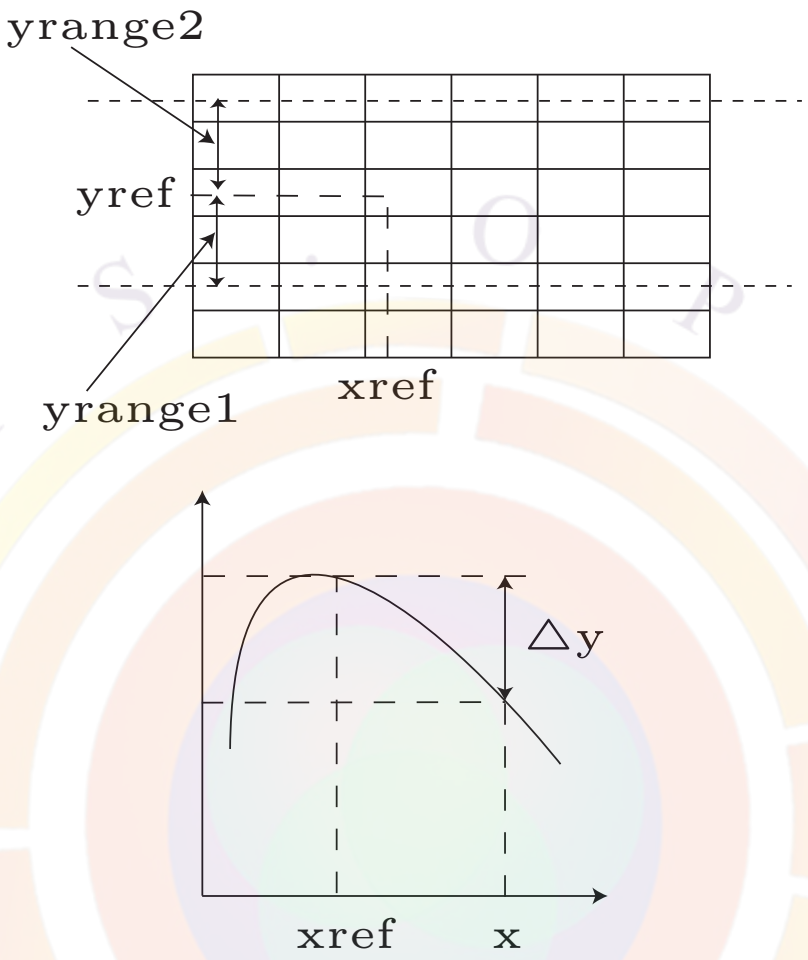


Figure 22.20: Visual representation of the parameters to **modify_layer_height**

Parameters

- **curve_data_file** is the name of the data file containing the profile/shape function used to alter the height of the layers. Note that only the shift away from the reference point is used to alter the profile: the absolute value of the reference curve is not used.
- **bottom_shaping** and **top_shaping** defines how the interfaces on each side of the reference point are altered. If set to *uniform*, the same y-shift is applied to all the interfaces; if set to *gradual*, the shift is linearly scaled so it becomes zero at the end of the range.
- **location**, **delta_y_from_bottom** and **delta_y_from_top** are used to define the reference y-position where the layer height is not modified. These parameters are used in the same way as those of **layer_position** and positions are relative to the last **layer** command issued.

- **relative_yrange** selects the interfaces that are affected by this command. It should be understood as an absolute position range with y_{ref} acting as the zero reference.

Examples

```
modify_layer_height curve_data_file=myshape_function_data.txt
```

This will modify all interfaces in the layer file according to the shape function.

```
modify_layer_height curve_data_file=myshape_function_data.txt &&
      relative_yrange=(-0.5 0.5) location=middle
```

This will set y reference to be middle of the preceding layer and modify the shape of all layer interfaces within $y_{ref} \pm 0.5\mu m$.

```
modify_layer_height curve_data_file=myshape_function_data.txt &&
      relative_yrange=(-0.5 0.5) location=middle
      bottom_shaping=gradual top_shaping=gradual
```

This is the same as the previous example except that the layer distortion is strongest at the reference point and goes to zero for interfaces close to the ends of the y-range.

22.472 modify_light_spectrum

parameter	data type	values [defaults]
light_filter	char	[no]
incident_power	real	[0.] (W/m^2) or W
wavelength	real	[1.] (μm)
filter_range	realx2	[-9999.0 -9999.0] (μm)

This statement is used to facilitate the modeling of the external quantum efficiency (EQE) of individual cells in a multi-junction solar cell. It allows the user to define a band-stop filter to prevent all absorption in a given cell (thereby making that cell current-limiting) and add a secondary monochromatic light source to probe the cell response.

modify_light_spectrum works in conjunction with the **light_power** statement and is often used as part of a series simulation to change the wavelength of the light probe.

Parameters

- **light_filter** activates a virtual band-stop filter on the incoming light spectrum.
- **incident_power** is the incident power of the secondary light source. It follows the same rules as in **light_power**.
- **wavelength** is the wavelength of the secondary light source.
- **filter_range** is the spectrum range of the virtual band-stop filter. By default, it blocks out the entire light spectrum.

Examples

```
modify_light_spectrum light_filter=yes filter_range=(0.3 0.8) &&
  incident_power=1.e5 wavelength=0.56
```

22.473 modify_opt_gen_rate

parameter	data type	values [defaults]
factor_file	char	[no] yes

modify_opt_gen_rate offers a way to modify the optical generation rate by a factor for different wavelengths.

- **factor_file** is the data file containing a 2-column data with first column being the wavelength in microns and the 2nd column being the factor to scale the optical generation rate profile.

```
modify_opt_gen_rate factor_file=my_scale_factor
```

22.474 modify_plot

parameter	data type	values [defaults]
surface_bottom	char	[yes]
show_data_points	char	[no]
watt_to_candela	char	[no]
qdot_wave_carrier_type	char	[electron,hole]
longitudinal_mode	intg	[1] 2 3
elec_conc_valley	intg	[1] 2
hole_conc_valley	intg	[1] 2 3
elec_conc_subb	intg	[1] 2 3 4
hole_conc_subb	intg	[1] 2 3 4
light_source	intg	[1] 2 3 4
pics3d_lateral_mode	intg	

modify_plot is a post-processor statement used to modify the appearance and content of subsequent plots.

Parameters

- **surface_bottom** affects surface plots to indicate whether a color bottom contour will be plotted to indicate different materials.
- **show_data_points** turns on/off plotting of individual data points in the graph.
- **longitudinal_mode** sets the index of longitudinal mode for the plotting statements related to 3D laser devices. This parameter is only used in PICS3D. Note that the lateral mode index is controlled separately, usually in the 2D plotting command itself.
- **elec_conc_valley** and **hole_conc_valley** are used to specify which band valley is used in the plotting of carrier concentrations.
- **elec_conc_subb** and **hole_conc_subb** are used to specify which subband is being plotted for the carrier concentrations.
- **light_source** can be used when multiple input light sources are defined in a simulation.
- **pics3d_lateral_mode** can be used in certain cases to sum up the power in all the longitudinal modes corresponding to a specific PIC3D lateral mode.

- **watt_to_candela** changes plots to display luminous intensity (SI unit of candela) instead of optical power (Watt). This is primarily used for LEDs emitting at optical wavelengths.
- **qdot_wave_carrier_type** is used when plotting the 3D (or cylindrical 2D) wave function from quantum dots and controls the carrier type that is plotted. The sub-band number may be accessed through the mode index parameter, similar to what is done to plot optical mode profiles.

Parameters

```
modify_plot surface_bottom=no
```

The above statement suppresses the plotting of material region at the bottom of a surface plot.

22.475 modify_qw

parameter	data type	values [defaults]
confine_left	char	[no] yes
use_bulkmass	char	[no] yes
kp_pot_sym	char	[yes] no
gain_num_integr	char	[no]
all_active_mater	char	[yes]
active_macro	char	[void]
tail_dos_n	char	[no]
tail_dos_p	char	[no]
kp88_cond_parab	char	[no]
use_av_pmass	char	[yes]
remove_above_line	char	[no]
lower_bound_label	char	[void]
upper_bound_label	char	[void]
complex_start_label	char	[void]
mb_pf_model	char	[no] yes
mater_label	char	
new_kp	char	
boundary_condition	char	[dirichlet], periodic
type2_detect	char	[no], yes
subhh_levelk(k=1..5)	real	[0.0]
sublh_levelk(k=1..5)	real	[0.0]

subhh_mxfack(k=1..5)	real	[1.0]
sublh_mxfack(k=1..5)	real	[1.0]
left_mesh	real	(um)
right_mesh	real	(um)
more_cond_energy	real	[0.](eV)
more_val_energy	real	[0.](eV)
dip_factor_pump	real	[1.]
tail_energy	real	[0.](eV)
tail_state_scale	real	[1.]
tail_bulk_ref	real	[0.02]eV
mb_damping_factor	real	[1.0]
set_kp8x8_angle	real	
max_kp8x8_angle	real	[90.]
scale_mb_screen_length	real	[1.]
max_mb_screen_length	real	[1.e16] (um)
mb_pf_expo_factor	real	[1.]
kp_search_offset	real	[0.1] (eV)
mater	intg	
kp_points	intg	[35]
kp_bands	intg	[2]
qlevel_once	intg	[0],1
kp_extra	intg	[10]
number_period	intg	[1]

modify_qw can be used in the gain preview (.gain) or main simulation (.sol) to modify quantum well model parameters.

Parameters

- **confine_left** is used to handle the special case of unconfined quantum states. In a self-consistent model of a quantum well, it may happen that the external bias is large such that right side (or the top side if well plane is horizontal) of the barrier is lowered to form another potential well which is made possible by the infinite potential wall at the right-most (or top-most) boundary. Solving the wave equation in such a coupled two-well system may result in many states being confined to the secondary right-well.

  These states are unrealistic and exist only because we impose an infinite potential wall by forcing the wave function to be zero at the right-most (or top-most)

boundary. Thus, we should filter out those unrealistic states while keeping the ones confined in the primary well on the left side (or lower side). The parameter **confine_left** is used to select the confined states in the primary well on the left. It is useful for quantum-MOS model when the gate bias is so large as to lower the oxide barrier potential to a level below that of the left barrier.

- **use_bulkmass** is to instruct the simulator to use bulk macro effective single-band mass when computing unconfined carrier density. We recall that the bulk macro provides effective mass for a single band model. For example, the heavy hole and light hole bands are combined to form an effective single valence band for convenience. One the other hand, the active layer macro for quantum well (or active region) computes the carrier density according to a multi-band model (with distinct HH and LH bands, for example).

  It is not always possible to reconcile the single-band and multi-band models. For example, if there is a split between the HH and LH levels as a result of strain, the equivalent single band model does not exist. In such a case, the simulator uses the multi-band model to override the single-band one for both confined and unconfined states within the active region.

  This may become a problem for the unconfined states since the model for the barrier region near a quantum well (in a self-consistent model, both well and barrier belong to the same active region) is different from that for the bulk region adjacent to it. We may see a discontinuity in background bulk (or unconfined) carrier density. Typically, if we plot the total carrier density, we may see the confined carrier density decays nicely to zero whereas there may be a discontinuity in the background bulk density. This behavior may destroy solution convergence causing frustration for users. The parameter **use_bulkmass** may be used to avoid such a situation. Please note that use of this parameter does not affect the discrete quantum states which are the most important in the active region.

- **kp_pot_sym** is used to indicate if symmetric potential is used to calculate the subband structure in the k.p theory.

- **gain_num_integr** is to indicate whether numerical integration is used to obtain the optical gain for the active region. By default, the free carrier optical gain is done by piece-wise analytical formulas. It is switched to numerical integration if many-body gain model is activated. One may choose to use numerical integration for the purpose of fair comparison with many-body gain. Also, in the limit of large **kp_points**, numerical integration produces the most accurate optical gain because direct integration is performed without any analytical fitting approximations.

- **all_active_mater** allows all active layers to use parameters in this statement, unless a specific material is defined by **active_macro** or **mater**.

- **active_macro** indicates that this statement affects all active layers with the above active macro name.
- **tail_dos_n** assigns the DOS tail (due to inhomogeneous broadening) to the conduction band. This works with the **tail_energy** parameter.
- **tail_dos_p** assigns the DOS tail (due to inhomogeneous broadening) to the valence band. This works with the **tail_energy** parameter.
- **kp88_cond_parab** sets parabolic conduction band for 8x8 valence mixing model. This parameter may be used to study the effect of non-parabolicity due to the conduction band.
- **use_av_pmass** instructs the simulator to use effective mass with average weighted by the envelop wave function for the purpose of computing optical gain. This parameter will have a significant effect if the barrier and well masses are substantially different.
- **remove_above_line** This parameter should be understood using a band diagram of quantum well with envelop function drawn at the confined energy levels. This parameter instructs the program to remove all quantum states which are centered above a line with connects the left-most and right-most points of the band diagram.
- **lower_bound_label**, if defined, is used by the program to set a lower boundary point in a 1-D potential distribution so that if the wave is localized outside of this lower limit, the state will be eliminated. This position label must be predefined in a .layer or .mater file. Note that this only applies to the full solution (.sol) since the .gain preview file does use the regular mesh.
- **upper_bound_label** is the same as **lower_bound_label** but sets an upper bound.
- **complex_start_label** must be defined if either or both of **lower_bound_label**, **upper_bound_label** are used. It is a position label for the bottom of the first quantum well of the MQW-complex being studied.
- **mb_pf_model** would turn on the Poole-Frenkel field dependence for the exciton model.
- **subhh_level_j,j=1..5** is the amount of modification made to the level of heavy hole subband j. "level" is measured downwards from the top of the corresponding bulk valence band.
- **sublh_level_j,j=1..5** is the amount of modification made to the level of light hole subband j. "level" is measured downwards from the top of the corresponding bulk valence band.

- **subhh_mxfac_j,j=1..5** is the factor multiplied to the in-plane heavy hole mass of subband j.

- **sublh_mxfac_j,j=1..5** is the factor multiplied to the in-plane light hole mass of subband j.

- **tail_energy** is the parameter E_{tail} describing an exponential tail of the joint density of states (JDOS) extending into the energy bandgap. The JDOS near the bandgap is written as JDOS $= \rho_r exp[(E - E_{edge})/E_{tail}]$, where E_{edge} is the energy of the band edge or the quantum subband edge, ρ_r is the reduced density of states at the band edge which is constant for a quantum well. This tail is associated with inhomogeneous broadening.

- **tail_bulk_ref** is for bulk active layer for which the reduced density of states is energy dependent. The exponential tail of the JDOS starts at a value (from the band edge) determined by this parameter.

- **tail_state_scale** may be used to scale the JDOS tail states described in the statement of **tail_energy** above.

- **left_mesh**, if defined, is the outer barrier thickness on the left side of a MQW region. This outer barrier is used to solve the Schrödinger equation and indicates the position of the zero-wave boundary condition; a convenient way to picture this is to imagine an infinite potential well encompassing the MQW region we are trying to solve.

  If this parameter is omitted then by default, a value of three times the quantum well width is used.

- **right_mesh** is the same as **left_mesh** but the defines the outer barrier thickness on the right-hand side of the MQW region.

- **boundary_condition** defines the boundary conditions used to solve the Schrödinger equation:

  - The default Dirichlet boundaries impose a zero-wave boundary condition as described in **left_mesh** above.
  - The periodic boundary conditions impose periodicity on the wave, with the total period length equal to the MQW thickness (including the outer barriers defined using **left_mesh** and **right_mesh**).

- **type2_detect** enables a model which automatically detects the position of the "well" and "barrier" layers in a MQW. This detection is necessary to work with type II superlattices and periodic boundaries but is disabled by default as it interferes with self-consistent quantum well regions and Dirichlet boundaries.

- **more_cond_energy** is used to extend the search of conduction subband energy levels up into the unbounded states. The densely populated energies in the extended unbound states can be used to treat optical transitions from bound-unbound states. Please note that the distribution of the unbound states depends on the boundary condition of the quantum mechanical wave equation. Only works when finite difference method is used for solving QW states (for non-symmetric QW, complex-MQW or self-consistent model).

- **more_val_energy** same as **more_cond_energy** except for the valence band.

- **dip_factor_pump** is the dipole factor affecting only the optical pumping wavelength.

- **kp_points** determines how many points in the k-axis will be included in the k.p valence mixing model. A smaller **kp_points** value results in a faster runtime but with a loss in precision.

- **kp_bands** is number of band valleys (such as HH, LH, SO) used in the k.p theory for zincblende material system. For wurtzite, see **modify_wurtizte**.

  The total order of the k.p matrix is twice the parameter value to take into account spin degeneracy:

  - 2: 4x4 k.p theory taking into account HH and LH bands.
  - 3: 6x6 k.p theory taking into account HH, LH and SO bands.
  - 4: 8x8 k.p theory taking into account the conduction, HH, LH and SO bands.

- **mb_damping_factor** is used to artificially scale the many-body effects contained in the gain/absorption spectrum. Please note that this parameter controls the magnitude and shape only. The bandgap renormalization can be scaled by **scale_mb_screen_length**.

- **set_kp8x8_angle** is the angle of k-vector with respect to the k_x axis in the k.p theory. If **new_kp**=*no*, this parameter only affects 8x8 k.p calculations. If this parameter is not specified, the Hamiltonian is averaged over a range from zero to **max_kp8x8_angle** before it is solved. Note that this is not the recommended averaging procedure and when using **gain_wavel** to average the gain, this angle is internally set. Users are thus advised to be careful when examining band dispersion results and transition dipole moments for 8x8 calculations: plotted values will not correspond to values from "proper" averaging.

  If **new_kp**=*yes*, the default setting of this value turns on the block-diagonalization procedure so that k.p results are independent of the in-plane angle, thereby

avoiding the averaging issue above. Setting a specific value instead forces the Hamiltonian to be solved at a specific in-plane angle: the axial approximation must also be disabled to use this feature.

- **scale_mb_screen_length** is used to artificially scale the screening length in the many-body theory and control the amount of bandgap renormalization.
- **max_mb_screen_length** is the maximum screening length used in the many-body theory gain model.
- **mb_pf_expo_factor** is a Poole-Frenkel field dependence factor used to account field dependence in the exciton model. This model assumes that external field alters the coulomb potential of the exciton in much the same way as in Poole-Frenkel style ionization. The formula we use is as follows:

$$\text{field factor} = exp(-qE_{shift}/kT) \tag{22.82}$$

where

$$E_{shift} = \left[\frac{Fq}{\pi\varepsilon_0\varepsilon}\right]^{(P_f/2)} \tag{22.83}$$

The **mb_pf_expo_factor** is the parameter P_f above.

- **max_kp8x8_angle** is the maximum angle used in averaging the k.p Hamiltonian if **set_kp8x8_angle** is not used. If **new_kp**=*yes*, this parameter may be used for all supported number of subbands; otherwise, only 8x8 calculations use this parameter.
- **mater** is the material number of the quantum well being modified. If a label has previously been defined for this material, **mater_label** may be used instead.
- **qlevel_once** forces the simulator to evaluate the quantum subbands only once in order to save on computation time. This parameter should not be used for self-consistent simulations since the multiple evaluations of the quantum subbands are the source of the self-consistency the user is trying to achieve.
- **new_kp** turns on a improved version of the zincblende k.p code. This version supports the following features for all supported subband settings:
  - block-diagonal mode with/without symmetric potential setting
  - explicit in-plane averaging of the in-plane angle (no-block diagonalization)

  Setting **new_kp**=*yes* is necessary to use non-common-atom interface effects in type II superlattices. See **nca_deltapot** for details.

- **kp_search_offset** controls some of the details of the eigenvalue search when **new_kp**=*yes*. The code searches for all eigenvalues $\leq E_0$ and after the first k_t iteration, this search offset is added to the highest found eigenvalue when starting a new search.

  This value also serves to detect type II band alignment situations when conduction band is coupling is considered; in that case, the search windows for eigenvalues may accidentally overlap and the number of search roots may need to be increased.

- **kp_extra** controls some of the details of the eigenvalue search when **new_kp**=*yes*. This is the number of extra search roots added to those found by a 1-band/parabolic solution of the Schrödinger equation. These extra search roots are necessary to improve the convergence of the "good" eigenvalues that are solutions of interest to the k.p Hamiltonian.

- **number_period** may be used if periodic boundary conditions are used to solve the Schrödinger equation and defines the total number of periods (N) in the superlattice. Note that this parameter does not repeat the MQW layer structure before the equation is solved: instead, multiple calls ($n = 0..N-1$) to the eigensolver are made, each with boundary conditions that impose a different phase shift to the wave function:

$$\Psi(z + Lp) = \Psi(z) * e^{iqLp}$$
$$q = \frac{2\pi n}{NLp}$$

  where Lp is the total period length, measured from the edges of the MQW solution domain (including *left_mesh* and *right_mesh*).

  Because only a single period is solved, the wave inside each period is normalized to $\frac{1}{N}$: this allows the gain curve to have the same maximum value for a single period and a superlattice. The number of periods parameter may be seen as merely providing a broadening of the transition energies by introducing additional near-degenerate solutions to the Schrödinger equation.

  The main advantage of this method (vs. simply repeating the MQW structure) is speed: eigensolvers tend to have a complexity greater than $O(N^2)$ so making repeated calls on a smaller problem makes the eigensolver solution time scale linearly. However, the user should keep in mind that the number of transitions will still scale as $O(N^2)$, no matter which method is used: this means that the gain calculations are always slower for a superlattice than for a single quantum well.

Examples

```
modify_qw sublh_level1=-0.01 sublh_mxfac1=1.5
```

To introduce an exponential tail in the joint density of states for optical transition in the active layer, you may use the following:

```
modify_qw tail_energy=0.06 tail_states_scale=1
```

22.476 modify_taper_height

parameter	data type	values [defaults]
bottom_data_from	char	
hline top_data_from	char	
hline bottom_data_to	char	
hline top_data_to	char	
hline region_ref_x	real	[0.] (um)
hline from_segment	intg	[1]

modify_taper_height modifies a taper line by defining a vertical offset in the mesh connection scheme. This can enhance the accuracy of a 3D simulation if, for example, the position of a QW region shift significantly and there is poor overlap between the two mesh planes during the 3D connection process. If this problem were allowed to persist, carriers could not freely flow along the longitudinal direction inside the QW and the connection would only be made indirectly, via other mesh points.

An alternate method of solving this kind of problem would be to define additional mesh planes and sample the longitudinal variation more precisely. However, this would increase the computational workload.

See **taper_between_segments** and Sec. 6.3 for a more complete description of the 3D modeling process and how tapers are defined in Crosslight.

Parameters

- **bottom_data_from** and **top_data_from** are data file names which define (x,y) pairs for the bottom and top of the taper region at the beginning of the taper. **bottom_data_to** and **top_data_to** are the matching parameters for the end of the taper.
- **from_segment** is the z-segment number immediately before the taper.

22.477 modify_vector_plot

parameter	data type	values [defaults]
vector_flag	intg	[2]

modify_vector_plot is a post-processor statement used to modify vector plots (such as current flow plots).

Parameters

- **vector_flag** flags different shapes of arrows used in the plot.

22.478 modify_wurtzite

parameter	data type	values [defaults]
fit_mass	char	[no],yes
mass_model	char	[average],small_k,large_k
simple_field_profile	char	[no],yes
barrier1_top_label	char	
mater_label	char	
c_axis_theta	real	[0.]
c_axis_phi	real	[0.]
psi_wurtz	real	[90.]
angle_ridge_c_axis_proj	real	
hex_lattice_a0	real	[3.189] Å
hex_lattice_c0	real	[5.185] Å
barrier1_top	real	
fit_mass_k_range	real	[0.08]
growth_plane_miller_index	intgx4	
mater	intg	
qw_plane_miller_index	intgx4	

modify_wurtzite is used to modify various parameters related to the wurtzite material system. Refer to Chap. 13 for additional theoretical background.

Parameters

- **fit_mass** indicates whether parabolic fitted masses are used for the valence band. If not, analytical mass models are used as explained in the **mass_model** parameter. The fitting range is defined by **fit_mass_k_range** which is a fraction of the Brillouin zone $\frac{2\pi}{a}$.

- **mass_model** indicates which analytical mass model is to be used. For a value of *small_k*, the zone center mass is used. **Caution: zone center masses can be negative in some material compositions, especially with tensile strain. This may cause numerical trouble because of parabolic band fitting approximations**. For a value of *large_k*, a large k-range is considered for the analytical model. *average* is an average of the above two cases.

- **simple_field_profile** turns on a simple 1D Poisson solver which can be combined with self-consistent quantum well calculations. This parameter is only used for gain preview calculations in the .gain file: in the full device simulation, the Poisson equation is solved over the full FEM mesh.

- **barrier1_top_label** and **barrier_top** are currently unused. They are intended to locate (using a position label or an absolute coordinate), the material corresponding to the first barrier and extract material properties for the simplified 1D Poisson solver in the gain preview mode.

- **c_axis_theta** and **c_axis_phi** are rotation angles θ and ϕ which define the orientation of the growth plane normal with respect to the original hexagonal coordinate system for wurtzite. By default, no rotation is defined which corresponds to the typical polar (c-plane) growth. Refer to Sec. 13.6 and Fig. 13.9 for more details.

- **psi_wurtz** is the in-plane rotation angle ψ which specifies the direction of the TE mode. This is needed to define the anisotropic refractive index that may arise in non-polar and semi-polar wurtzite orientations. **angle_ridge_c_axis_proj** is the complement of this angle and defines the waveguide propagation direction: only one of these these values needs to be defined. Refer to Sec. 13.6 and Fig. 13.9 for more details.

- **growth_plane_miller_index** can be used to specify the direction of the growth plane normal using the Bravais-Miller indices ($hkil$) instead of using θ and ϕ. The lattice constants **hex_lattice_a0** and **hex_lattice_c0** must be provided to calculate the vector direction. Note that the direction of the waveguide (ψ) must still be defined.

- **qw_plane_miller_index** is similar to **growth_plane_miller_index** but specifies the direction of the quantum confinement rather than the growth

plane. This is often used in GaN nanowires where confinement occurs in the side walls rather than in the growth direction.

- **mater** may be used to specify different wurtzite settings for different material numbers in the simulation. For example, the side walls of a GaN nanowire may each have different properties. If a label has previously been assigned to this material, **mater_label** may be used instead.

Examples

```
modify_wurtzite fit_mass=no mass_model=average
```

22.479 modu_bias

parameter	data type	values [defaults]
section1	real	[1.]
section2	real	[1.]
section3	real	[1.]
section4	real	[1.]
section5	real	[1.]
n_sections	realxn	
sec_num	intg	[-1]

modu_bias is used in the older AC model described in Sec. 18.6 to specify which section of the laser is modulated for the analytic AM and FM responses. In general, the fully coupled AC model in the post-processing stage should be preferred.

Parameters

- **section***i* (i=1,2,3,4,5) gives the relative modulation bias current at section i. This can be used to define push-pull (out-of-phase) modulation or simply to adjust the amplitude of the modulation.

- **n_sections** can be used to list certain sections that share the same modulation bias and will override the above **section***i* setting. The number of items in that list is given by **sec_num**.

Examples

```
modu_bias section1=1 section2=3

modu_bias sec_num=3 3_sections=(1 3 2)
```

22.480 monitor_emission

monitor_emission is used to plot the emitting light power in mesh points. It provides viewing the emission process for the "mesh_points" emission model. This statement is used at the post-processing stage after ray-tracing program is run.

This statement has no parameters.

22.481 more_dos_fermi_output

parameter	data type	values [defaults]
near_xyz	realx3	[0 0 0] (um)
conduction_band_valley	intg	
valence_band_valley	intg	

more_dos_fermi_output exports the density of states curve (DOS(E)) and quasi-Fermi level for a specific mesh point of the device.

See also **plot_more_dos_fermi** which is used to plot the resulting curve in the post-processing stage.

Parameters

- **near_xyz** specifies the coordinates of the mesh point for the data export.
- **conduction_band_valley** is the conduction band valley number for the export.
- **valence_band_valley** is the valence band valley number for the export.

22.482 more_output

parameter	data type	values [defaults]
ac_data	char	[no], yes
qw_states	char	[no], yes
ac_light_input	char	[no], yes
elec_mobility	char	[no], yes
hole_mobility	char	[no], yes
trap_information	char	[no], yes
minimum	char	[no], yes
rcled_information	char	[no], yes
impact_ionization	char	[no], yes
organic_exciton	char	[no], yes
space_charge	char	[no], yes
light_reflection	char	[no]
pics3d_ase	char	[no]
nonlocal_current	char	[no]
split_qw_states	char	[no]
pics3d_longitudinal	char	[yes]
qdot_conc	char	[no]
stress	char	[no]
heat_flux_density	char	[no]
polar_spectrum	char	[yes]
polar_vector	char	[no]
pics3d_noise	char	[no]
wave_phase	char	[no], yes
spectrum_print	char	[yes], no
mixmode_ac_data	char	[no], yes
mesh_detail	char	[no], yes
peltier_heat_related	char	[no],yes
pics3d_photon_bal_ac	char	[no], yes
elec_conc_subb	intg	[0]
hole_conc_subb	intg	[0]
cond_valley_prop	intg	[0]
val_valley_prop	intg	[0]
sum_space_charge_mater	intg	[0] 1 2..

The statement **more_output** is used to instruct the simulator to print additional data which is not normally included in the output files. This statement will affect

all **scan** statements that follow and should therefore be included at the beginning of the .sol file.

- **ac_data** generates the data necessary for small-signal AC analysis in the post-processing stage.
- **qw_states** enables the printing of quantum well subband wave functions. This may only be used if self-consistent MQW option is activated.
- **ac_light_input** is used in conjunction with **ac_data** to obtain the small-signal response of a photo-sensitive device in the post-processing stage.
- **elec_mobility** and **hole_mobility** print the carrier mobility distribution.
- **trap_information** enables the printing of trap information such as trapped electrons and trap levels.
- **minimum** may be used save on disk space by printing only the minimum amount of data sufficient to keep the main solver running. As a result, many quantities may not be available for plotting during the post-processing.
- **rcled_information** enables the printing of RCLED related information, such as optical cavity standing power distribution.
- **impact_ionization** enables the printing of data related to the impact ionization (II) model such as the II rate distribution.
- **organic_exciton** enables the printing of organic exciton diffusion profiles.
- **space_charge** enables the plotting of the space charge distribution.
- **light_reflection** turns on data related to light reflected by the transfer matrix model (TMM) used by the optical pumping code.
- **pics3d_ase** prints data related to the amplified spontaneous emission model (ASE) in PICS3D. This is primarily used in amplifiers (SOAs) and superluminescent diodes (SLEDs).
- **nonlocal_current** prints data for the non-local transport model used in **q_transport** and other similar commands.
- **split_qw_states** prints data for the split-state model in the **q_transport** statement.
- **pics3d_longitudinal** prints data that can be plotted along the optical longitudinal axis in PICS3D rather than the electrical z-axis imported from the mesh planes. This is necessary to use **plot_longitudinal** in the .plt file.

- **qdot_conc** enables plotting of the concentration inside embedded quantum dots.
- **stress** enables plotting the stress profiles which can be imported from CSUPREM or defined manually in APSYS.
- **heat_flux_density** prints data related to the heat flow for use in statements such as **plot_2d**.
- **polar_spectrum** enables plotting of the TE and TM spontaneous emission spectra.
- **polar_vector** enables plotting of the polarization vector in piezoelectric materials. It is related to the **polarization_charge** command.
- **wave_phase** is used in PICS3D to print the phase shift seen by the output light of an amplifier (SOA) or modulator (EAM).
- **pics3d_noise** enables plotting of noise spectrum terms in PICS3D such as relative intensity noise (RIN) and frequency modulation (FM) noise.
- **spectrum_print** may be used to disable optical gain and spontaneous emission spectrum data in order to speed up a simulation.
- **mixmode_ac_data** enables the use of the small-signal AC model for external circuit elements during the post-processing stage.
- **mesh_detail** enables printing of special detailed information about mesh points, such as the equivalent interface length at material boundaries or the local area around mesh points. Printing of this data may be useful to convert back and forth between 2D and 3D quantities on surfaces or to manipulate the raw data produced by **raw_output**.
- **peltier_heat_related** enables printing of key derivatives and gradients used in the calculations of the Thomson/Peltier heat source.
- **pics3d_photon_bal_ac** is only used when **ac_data**=*yes*. When enabled, this statement instructs PICS3D to export an AC Jacobian based on the photon balance model described in **init_wave** instead of the usual round-trip-gain (RTG) model.

  Note that if the photon balance model is turned on in **init_wave**, this parameter should not be used as the AC data will already in the photon balance format. This parameter should only be enabled in cases where the RTG model is used for DC simulations but the user wishes to use the photon balance model in the AC post-processing.

- **elec_conc_subb** and **hole_conc_subb** determine how many quantum well subbands are available when plotting carrier concentration in the post-processing.
- **cond_valley_prop** and **val_valley_prop** determine how many properties for each band valley (e.g. mobility or effective mass) are stored in the output file. See the various variations of **condj_valley_propk** and **valj_valley_propk** for details.
- **sum_space_charge_mater** enables printing of the integrated net space charge for a given material number. This parameter must match the material number defined when loading the macros and may usually be found in the .mater file.

Examples

```
more_output ac_data=yes qw_states=no
```

22.483 more_spectrum_output

parameter	data type	values [defaults]
variable	char	[gain]
mode	char	[te],tm
mater_label	char	
wavel_range	realx2	[.5 1.5]
diffuse_length	real	[0.2] (μm)
diffuse_center_depth	real	[0.2] (μm)
mater	intg	[1]
point_num	intg	[99]
tunnel_region	intg	
set_lateral_mode	intg	

more_spectrum_output can be used to generate additional spectrum output that would be otherwise unavailable for plotting. See also **plot_more_spectrum**.

Parameters

- **variable** is the spectral variable being exported. See the online Wizard in SimuCenter for a full list of supported variables.

- **mode** specifies whether the TE or TM spectrum data is being printed.

- **wavel_range** is the wavelength range over which the variable is exported. The number of data points in this range is given by **point_num**.

- **diffuse_length** and **diffuse_center_depth** are used in Optowizard to give a very rough approximation of the effects of diffusion current on the quantity being exported. In this program, the current transport equations are not solved so a makeshift diffusion constant (**diffuse_length**) is used to smooth out the results; the maximum value is assumed to be at a certain fixed position (**diffuse_center_depth**).

  Variables affected by this artificial broadening have a "diffuse" in their name.

- **mater** is the material number corresponding to the variable being exported. If a label has previously been defined for this material, **mater_label** may be used instead.

- **tunnel_region** defines the tunnel region number corresponding to the variable being exported. These regions are numbered according to the order in which the tunneling commands appear in the input file.

- **set_lateral_mode** fixes sets the lateral mode for the variable being saved (e.g. for modal gain).

22.484 more_sym_polygon

parameter	data type	values [defaults]
polygon_center_x	real	[1.0] (um)
polygon_center_y	real	[1.0] (um)

The statement **more_sym_polygon** creates another copy of a polygon defined previously with **sym_polygon_for_semicrafter**

- **polygon_center_x** and **polygon_center_y** are the (x,y) center coordinates of the new polygon.

22.485 more_tcadmesh

parameter	data type	values [defaults]
name	char	
contactj_to_spice_node (j=1..109)	char	

more_tcadmesh expands the **minispice** statement and allows for multiple SPICE entities in the same integrated circuit to be replaced by TCAD devices. **more_tcadmesh** should be repeated as many times as necessary when replacing multiple devices in the same circuit file, the first replacement being done by the **minispice** statement itself.

One important point to understand for multi-device simulation is that the Poisson and Drift-Diffusion equations for all the devices must be solved simultaneously. This means that all of the TCAD devices are part of the same simulation and a three-dimensional mesh is required: this can mean multiple 2D devices (1 mesh plane per device) or multiple 3D simulations for each device (each device represents a subset of the total number of mesh planes).

Either way, while the TCAD devices must all be solved at the same time, it does not necessarily mean that those devices are monolithically integrated. That is, even if multiple devices co-exist within the same sparse matrix in the Newton solver, this does not imply that there is current flow between "neighboring" devices. A helpful way of separating the devices is the **isolate_mesh_segment** command: this cuts off the FEM connection (current flow and potential) between devices.

Please note that Crosslight supports setting up each TCAD device in its own separate project folder. This allows for initial device-level simulation before modeling the integrated circuit. Statements such as **set_include** may be used to simplify the integration of multiple project files.

Parameters

- **name** is the name of the SPICE entity being replaced by a meshed TCAD device.
- **contactj_to_spice_node** associates a specific contact number ("j") with the given spice node.

Examples

```
minispice circuit_file=dio.cir  &&
```

```
spice_device_to_tcadmesh=Ztcad1  &&
contact1_to_spice_node=1 &&
contact2_to_spice_node=2
```

22.486 more_trap_output

parameter	data type	values [defaults]
tag	char	
x_label	char	
y_label	char	
zplane_label	char	
near_x	real	[0.] (μm)
near_y	real	[0.] (μm)
zplane_num	intg	[1]

This statement will output trap information near a specified point. This information is output only during normal printing of mesh-related data (e.g. band diagrams) and is not available for all bias steps.

Parameters

- **tag** is a user-defined label which links this command with **plot_more_trap**.

- **x_label**, **y_label** and **zplane_label** are position labels used to identify the coordinates of the data export. **near_x**, **near_y** and **zplane_num** serve the same purpose but rely on the x,y coordinates and mesh plane number (for 3D simulations)..

22.487 multimode

parameter	data type	values [defaults]
sort_modepeak	char	[void]
sort_modespan	char	[void]
initialize_bpm	char	[no] yes
mode_num	intg	[1]
boundary_type1	intgx4	
boundary_type2	intgx4	
pick_xmode	intg	[0]
pick_ymode	intg	[1]
ymode_num	intg	[1]
polar_num	intg	[1]

The statement **multimode** is used to control the multi-lateral mode solutions. In general, the wave equation has two types of boundary conditions if the device is symmetric and only the right half is simulated. The two types are symmetric and anti-symmetric wave functions with respective to the axis of symmetry (usually located on the left hand side). Please see the definition of boundary conditions in the statement **init_wave**. If the whole device is simulated, only one type of boundary condition is sufficient.

If there are n modes required and there are two types of boundary conditions, the first n/2 modes are solutions obtained with the first type of boundary condition and the last n/2 modes are those with the second type.

Using the effective index method, the simulator is capable of solving for the optical power of more than one lateral (x-mode) mode. However only one transverse (y-mode) mode is selected.

Please note that by default, the optical modes are numbered according to the modal index, starting from number one for the highest index. Sometimes, it is desirable to order the modes according to their geometries or shapes.

- **sort_modepeak** may take **void, +x, +y -x** or **-y**. If not **void**, it is used to instruct the program to sort the lateral optical modes according to the positions of the mode peak. For example, a parameter value of **+x** will result in modes being sorted such that mode peaks are located increasingly towards the +x direction with increasing mode number.
- **sort_modespan** may take **void, +x, +y -x** or **-y**. If not **void**, it is used to instruct the program to sort the lateral optical modes according to the span (or width) of the mode. For example, a parameter value of **+x** will result in

modes being sorted such that the mode half width in the x-direction increases with increasing mode number.

- **initialize_bpm** indicates whether a multiple mode solution is used to initialize a BPM calculation.

- **mode_num** is the number of lateral modes to be included in the simulation.

- **boundary_type1** is the first type of wave equation boundary. Its definition is identical to **boundary_type** in the the statement **init_wave**.

- **boundary_type2** is the first type of wave equation boundary. Its definition is identical to **boundary_type** in the the statement **init_wave**.

- If **pick_xmode** is zero, the program will solve all modes from mode number 1 to mode number **mode_num** (specified above) in the x-direction (lateral mode). If greater than zero, the program will solve only for mode number **pick_xmode**. For a device with multi-mode solution, it is recommended that all lateral modes be solved **pick_xmode=0**.

- **pick_ymode** must be a non-zero integer so that one mode in the transverse direction (y-direction) is selected for the simulator. Please note that the optical modes are numbered according to the modal index, starting from number one for the highest index.

- **ymode_num** is the number of y-modes.

- **polar_num** is number of polarizations (TE/TM) involved in the simulation.

Example(s)

```
multimode mode_num=6 boundary_type1=(2 1 1 1)  &&
  boundary_type2=(1 1 1 1)
```

In this example, six lateral modes are required. The device has a symmetric axis located on the left hand side (*i.e.* lx1, please see **init_wave**). The first three modes belong to symmetric (or even parity) modes of the first boundary type while the last three modes correspond to the second type with anti-symmetric (or odd parity) modes.

22.488 multimode_detect

parameter	data type	values [defaults]
detect_ratio	real	[1.]
mode_num	intg	[1]

The statement **multimode_detect** is used to specify the difference in power detection from different lateral modes. Usually, the laser power measurement equipment covers a limited angular range and higher order lateral modes will be detected with less amount of power.

- **detect_ratio** is a factor proportional to the power detected for a certain lateral mode.
- **mode_num** is the lateral mode index being affected.

Example(s)

```
multimode_detect front_back=0.7 mode_num=2
```

In this example, the 2nd order lateral mode detected is 70 percent of the fundamental mode which has a default detection factor of 1.

22.489 multimode_mirror

parameter	data type	values [defaults]
front_back	realx2	
mode_num	intg	[1]

The statement **multimode_mirror** is used to control the multi-lateral mode mirror reflectivity. Without this statement, mirror reflectivities are assumed to be the same as set by statement **init_wave**. It allows setting of different reflectivities for different lateral modes.

- **front_back** is used to define both front and back facet (or left and right) power reflectivity.
- **mode_num** is the lateral mode index being affected.

Example(s)

```
multimode_mirror front_back=(0.2 0.2)  mode_num=2
```

In this example, the 2nd order lateral mode is set to experience a mirror reflectivity of 20 percent while the fundamental mode is still defined by statement **init_wave**.

22.490 nca_deltapot

nca_deltapot is an active layer macro statement that defines "non common atom" (NCA) effects at the interface of a quantum well. As explained in the literature[133], the standard $k \cdot p$ theory used to derive the Hamiltonian for zincblende structures assumes the existence of certain symmetries and under certain conditions this approximation breaks down. For example in the case of InAs/GaSb superlattices, there is no shared atom between the two regions so when looking at the location of the atomic bonds, the [110] and [$\bar{1}$10] directions are no longer equivalent.

In order to continue using our usual $k \cdot p$ theory, a correction term must be added to the Hamiltonian. Unlike other $k \cdot p$ coupling terms which depend on the in-plane kinetic energy, this correction term breaks the symmetry of the potential profile and takes the following form:

$$V = H_{XY}^{AB} a_0 \delta(z)\Theta - H_{XY}^{BA} a_0 \delta(z - L)\Theta \tag{22.84}$$

where H_{XY} is the strength of the interface effect (in eV), a_0 is the lattice constant and $\delta(z)$ is the Dirac delta function which localizes the interface peaks on either side of the QW. Θ is the $k \cdot p$ matrix coupling the different hole basis functions and which is given in [133]: unlike the usual well/barrier potential terms, the NCA correction does not go into the diagonal of the Hamiltonian.

The value defined by **nca_deltapot** gives the strength of H_{XY}^{AB} on the left side of the QW with **nca_deltapot_right** giving the value on the right side; the minus sign of Eq. 22.84 is already built into the program for the right side. If either of these statements is omitted, the corresponding H_{XY} is set to zero; in the absence of both statements, the default (no NCA effects) $k \cdot p$ theory is recovered. Since the literature indicates that both parameters seem to have a complimentary effect, we expect that most users will only use **nca_deltapot** as a fitting parameter and leave $H_{XY}^{BA} = 0$ by omitting **nca_deltapot_right**.

Note that as of the 2014 version of APSYS, we follow [133] and assume atomically sharp interfaces in the correction term: the coupling matrix Θ reflects this. If there is sufficient user demand, graded interfaces may be added at a later date by modifying the coupling matrix and adding additional fitting parameters; please consult Crosslight if you need to include these effects in your models.

We also note that because of the shape of the coupling matrix Θ, the correction term is not compatible with $k \cdot p$ calculations using the block-diagonalization theory pioneered by S.L. Chuang[15, 33, 63]. As a result, plots of the subband dispersion relations and/or the transition dipole moments will only include NCA effects if the axial approximation is disabled in **set_active_reg** and a specific in-plane angle is specified in **modify_qw**. However, NCA effects will be included as part of an optical gain/absorption calculation if explicit averaging of the in-plane angle is used in lieu of block-diagonalization; see **gain_wavel**.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.491 nca_deltapot_right

See **nca_deltapot**.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.492 negf_model

parameter	data type	values [defaults]
add_to_dd_current	char	[yes],no
phonon_scat	char	yes,[no]
use_contact_imref	char	yes,[no]
model_type	char	[quantum_well], quantum_wire
charge_couple_model	char	[dd_conc_inject], add_negf_conc, mix
carrier	char	[electron], hole
division	real	[5.e-4] (μm)
phonon_energy	real	[0.062] (eV)
phonon_elast_rate	real	[1.08e-2] (eV^2)
phonon_inelast_rate	real	[0.0147] (eV^2)
max_tunnel_factor	real	[1.e10]
scale_current	real	[1.]
solve_schr_num	intg	[10]
subband_num	intg	[2]
tunnel_contact_from	intg	[1]
tunnel_contact_to	intg	[3]

negf_model is new to the 2013 version of Crosslight and implements a Non-Equilibrium Green's Function (NEGF) transport model[134]; currently, only electron transport is considered in this model. This command works in conjunction with **tunneling** to define the range over which the NEGF transport operates.

See also **negf_plot** which is used to visualize output from this model.

Parameters

- **add_to_dd_current** controls whether the NEGF current is added to the classical drift-diffusion current. There are two reasons for adding the NEGF current to the classical drift-diffusion current:

  1. NEGF treats only the coherent quantum ballistic transport which incoherent drift-diffusion does not include: for structures with a potential barrier, adding the NEGF contribution recovers the current that is missing from the classical model.
  2. The NEGF model in Crosslight is only applied to a subset of the whole device (both in terms of energy and geometry) in order to save on com-

putation time. Adding the NEGF model to the classical drift-diffusion allows for a proper model of the entire device.

The downside to this approach is that there will be double-counting for the part of the current above the barrier which is already included via thermionic emission in the classical drift-diffusion model. For devices without significant tunneling potential barriers, **add_to_dd_current**=*no* may therefore be more accurate.

However for most devices where non-equilibrium/ballistic transport occurs, **add_to_dd_current**=*yes* is recommended since it may be important to include the incoherent diffusion process which is not included in the NEGF. Please note that the DD current can easily be adjusted through the mobility and the saturation velocity settings.

- **charge_couple_model** determines how the NEGF charge is coupled into the Poisson equation. This carrier charge can either be the one from the standard drift-diffusion model, the one for the NEGF model or a mixture which progressively switches to the NEGF model at high current densities. This parameter may affect convergence.

- **phonon_scat** turns on the phonon scattering model in the NEGF calculations. The LO phonon energy ($\hbar\omega_0$) for the material is given by **phonon_energy** while the elastic and non-elastic collision rates are defined using **phonon_elast_rate** and **phonon_inelast_rate**, respectively. These rates correspond to the D_0 coefficient often found in the literature:

$$\Sigma^{in}(E_l) = D_0\left[(N+1)\cdot G^n(E_l+\hbar\omega_0) + N\cdot G^n(E_l-\hbar\omega_0)\right] \tag{22.85}$$

$$\Sigma^{out}(E_l) = D_0\left[(N+1)\cdot G^p(E_l+\hbar\omega_0) + N\cdot G^p(E_l-\hbar\omega_0)\right] \tag{22.86}$$

- **division** is the sampling interval along the transport direction used to solve the Schrödinger equations. This value should be less than the quantum mechanical wavelength or the NEGF current may be overestimated.

- **max_tunnel_factor** is used to limit the tunneling coefficient and avoid accidental numerical blow-ups in the calculations.

- **scale_current** is a fudge factor that may be used to artificially scale the NEGF current.

- **solve_schr_num** is the number of Schrödinger equations being solved. The computation time will rise linearly with this number.

- **subband_num** is the number of sub-bands (confined states) considered in the model. To save on computation time, it is assumed that only the first few sub-bands are populated enough to significantly contribute to the NEGF current.

- **model_type** determines whether the carriers are confined in 2D (quantum well) or 1D (quantum wire). The 1D model may be appropriate in certain MOSFET channel designs.

- **tunnel_contact_from** and **tunnel_contact_to** are the contact numbers which serve as the boundary conditions for the Green's function. At these points, the carrier populations are assumed to be at equilibrium and follow normal Fermi-Dirac statistics.

  use_contact_imref also controls this behavior by determining where the equilibrium boundary is located: either at the mesh points where the contact's ohmic/Schottky boundary is defined or at the start/end points of the NEGF region in the highly-doped source/drain region. Minute differences in the position of the Fermi level at these two points may lead to convergence issues so using the Fermi level/IMREF at the edges of the NEGF region is likely to lead to better results.

- **carrier** determines whether this command defines electron or hole transport. Note that at present, our NEGF model is still unipolar even if both carrier types are available.

Examples

```
$ this is for ballistic transport along the channel.
tunneling direction=x carrier=electron barrier_type=propagation_matrix &&
  x1_label=x1 y1_label=y1 x2_label=x2 y2_label=y2
negf_model division=2.e-4 solve_schr_num=10 add_to_dd_current=yes &&
  phonon_scat=no
```

22.493 negf_plot

parameter	data type	values [defaults]
variable	char	
log_conc	char	yes,[no]
view_xrot	real	[0.] (degrees)
view_yrot	real	[0.] (degrees)
valley	intg	[1]
subband	intg	[1]

negf_plot is a post-processing command used to plot data related to the Non-Equilibrium Green's Function (NEGF) model.

Parameters

- **variable** can show either:
  - *band_structure*: a surface plot of the conduction band used for NEGF transport
  - *elec_conc*: a surface plot of the electron concentration
  - *current_spectrum*: a plot of the NEGF current vs. energy
  - *elec_density_spectrum*: a plot of the carrier density vs. energy
  - *quantum_level*: is a 1D plot of the sub-band energy level vs. position
- **log_conc** determines whether carrier concentrations are shown on a linear or log scale.
- **view_xrot** and **view_yrot** are angles of rotation applied to 3D plots for viewing purposes.
- **valley** is the band valley number for which the NEGF calculations have been performed.
- **subband** is the confined sub-band number for which the NEGF calculations have been performed. It must be consistent with **subband_num** from **negf_model**.

22.494 new_inset_planes

This command is for internal use only.

22.495 newton_freeze_carrier

parameter	data type	values [defaults]
freeze_electron_mater	intg	
freeze_hole_mater	intg	
unfreeze_electron_mater	intg	
unfreeze_hole_mater	intg	
freeze_both_mater	intg	

The usage of **newton_freeze_carrier** is similar to that of **newton_par** and it defines parameters for the non-linear iterations in Newton's method. It should be used before the **scan** statement it intends to control.

Parameters

- **freeze_electron_mater** would freeze the electron profile for the material it specifies.
- **freeze_hole_mater** would freeze the hole profile for the material it specifies.
- **unfreeze_electron_mater** would unfreeze the electron profile for the material it specifies.
- **unfreeze_hole_mater** would unfreeze the hole profile for the material it specifies.
- **freeze_both_mater** would freeze both the electron and hole profile for the material it specifies.

Examples

```
newton_freeze_carrier freeze_hole_mater=3
```

The above command would freeze the hole distribution for material number 3. This can be set in the middle of a simulation when hole profile is not important.

22.496 newton_par

parameter	data type	values [defaults]

search_gainpeak	char	[full]
extrapolate	char	[yes], no
change_variable	char	yes, [no]
limit_elem	char	yes, [no]
update_tunneling	char	[yes], no
update_mqw	char	[yes], no
update_lateral_mode	char	[yes], no
negative_stimulate	char	[yes], no
recover_prev_mqw	char	yes, [no]
rescale_poisson	char	[yes], no
change_var_type	char	[both], n, p
update_rc_extraction	char	[yes],no
freeze_carrier	char	[void], electron, hole, both
update_index_change	char	[yes],no
update_macro	char	[yes],no
update_index_change	char	[yes], no
update_macro	char	[yes], no
turnoff_update_lightprof	char	[no], yes
update_rtg_phase	char	[yes], no
stall_restart	char	[yes], no
lateral_mode_perturbation	char	[no], yes
linear_matrix	char	[no], yes
damping_step	real	[2.0]
var_tol	real	[1.e-5]
res_tol	real	[1.e-5]
min_lit	real	[0.]
rate_damp	real	[1.e-5]
scale_eqn_n	real	[1]
scale_eqn_p	real	[1]
impact_damp	real	[1.e-9]
var_fac	real	[1.e3]
ratio_max	real	[-0.5]
huge_elem	real	[1.e10]
density_tol	real	[1.e16]
step_decrease	real	
step_increase	real	
back_refine_error_scale	real	[1.]
max_iter	intg	[50]
opt_iter	intg	[15]
stop_iter	intg	[15]
print_flag	intg	1, [2], 3

loopback	intg	[0]
converge_test	intg	[2]
gainsearch_num	intg	[30]
iter_flag	intg	[0] 1 2
iter_step	intg	[5]
mf_solver	intg	0, 1, 2, [3], 4
mf_md_flag	intg	[1] 0
mf_pivot_update	intg	[64]

The statement **newton_par** controls the non-linear Newton solver used in our software. The parameters of this statement may be used to improve the convergence rate of difficult problems.

For more details on the equations being solved and the numerical techniques involved, please see Chapters 5 and 6. Additional advice on troubleshooting convergence issues can be found in Chap. 4.

Note that the nonlinear variables are internally normalized to the order of 1-100.

Examples

- **search_gainpeak** is used to pick the wavelength of the lateral mode. Since many spectral quantities go into the non-linear equations, this can have an influence on convergence.
  - *full* means the peak of the modal gain, integrated over all mesh points. This is the default method except for PICS3D.
  - *one_point* means the peak of the material gain at a chosen mesh point. This is the fastest but least numerically stable method.
  - *long_mode* means the wavelength is determined by the longitudinal mode solver. This is the default method for PICS3D.
- **extrapolate** turns on/off the solution extrapolation used to improve the initial solution guess in the non-linear Newton solver. In most situations, this should be enabled. However, in low temperature simulations using carrier concentrations as solver variables, this should be disabled to avoid numerical overflow.
- **change_variable** is used to change the variables of the device simulator from quasi-Fermi levels to carrier densities. This can be helpful in carrier-blocking

structures or other low-current regimes where even large changes in the quasi-Fermi levels still produce very small carrier densities. For examples of such situations, see Chap. 4.

Note that **change_var_type** can be used to restrict the change of variables to only one type of carrier. However, this is not usually recommended in practice.

- **limit_elem** is used to set a upper limit to the off-diagonal elements in the Jacobian matrix. If set to *yes*, the parameter **huge_elem** will be used as the upper limit. When using carrier densities as the variable of solution, the off-diagonal elements may be so large that the computer can not handle it and may give an overflow error (printed as NaN). In such a case, it may help to set this parameter.

- **update_tunneling** can be used to update the tunneling model at every bias step.

- **update_mqw** can be used to update the quantum well subband calculation model at every bias step.

- **update_lateral_mode** is used to update the lateral mode eigensolver at every bias step. If lateral mode switches cause the main simulator to diverge, one may wish to switch this off so that a more stable (but less self-consistent) solution may be obtained.

- **negative_stimulate** controls whether or not the stimulated recombination term may become negative.

  To clarify the meaning of this parameter, it is important to note that carrier generation (G) and stimulated recombination (R_{stim}) are usually separate terms in the Crosslight model. The first term usually comes from an external light source and is used in solar cells or optically-pumped lasers; the second term comes the optical gain calculations and is used for internally-generated light propagating through a waveguide. These two terms often involve very different wavelength ranges and must thus be treated separately.

  If **negative_stimulate**=*yes*, stimulated recombination may give rise to a carrier generation term but in common laser designs, transparency occurs before the current crosses threshold so this case can often be neglected (i.e. $S \approx 0$ when gain is negative). However in structures with strongly non-uniform local gain profiles, neglecting this term may create kinks in the L-I curve: one part of the active region structure may provide enough gain to cross threshold and support a large photon density while another part of the active region is still absorbing.

  On the other hand, gain curves below transparency are also known to have very large negative values away from the bandgap: this can lead to numerical overruns during the wavelength search of the peak if **negative_stimulate**=*yes*.

Turning this effect off may thus be seen as a convenient way to avoid numerical difficulties.

In general, **negative_stimulate**=*yes* represents the correct physics and should be used in most cases. If users observe the numerical overruns described above, it is recommended to restrict the wavelength search range in **init_wave** instead of using the =*no* setting.

- **recover_prev_mqw** is used to modify the behavior of the self-consistent quantum well solver when a bias step in **scan** fails. When this parameter is enabled, the program solves and recovers the QW solution for the previous successful step to ensure that the previous convergence condition (including QW solution) can be reproduced. This recovery action involves more numerical solutions and takes more computation time. Otherwise the simulator does not recover previous QW solution and simply reduces the bias step in the hopes that this produces a convergent solution. In general recovery action increase convergence and accuracy of the simulation at the cost of extra computation time.

- **rescale_poisson** indicates that whether the Poisson's equation is to be scaled. This may affect converge.

- **update_rc_extraction** updates the resonant cavity extraction coefficient from the RCLED model at every bias step. This parameter can improve the smoothness of the L-I curve in these cases.

- **freeze_carrier** stops the update of the specified carrier distribution in all materials. Note that a related command can be used to stop the update for a specific material: **newton_freeze_carrier**.

- **update_index_change** turn on the updates to the refractive index (i.e. the change from its equilibrium value) at every bias step.

- **update_macro** turn on the updates to various material macro parameters at every bias step.

- **turnoff_update_lightprof** turns off the update of the light profile when using optical pumping.

- **update_rtg_phase** works similarly to the **ignore_rtg_phase** of the **longitudinal** statement. However, instead of completely ignoring the imaginary part of the round-trip gain equation (i.e. the wavelength of the mode) this parameter instructs PICS3D to stop solving this equation, thereby freezing the value of the mode wavelength.

- **stall_restart** controls the behavior of the software when a **scan** command loses convergence. If this parameter is enabled, the software will move on to

the next **scan** command in the input file (if any) instead of terminating the program.

IMPORTANT NOTE: This restart behavior can have unexpected side effects on device simulations. Unless otherwise specified, the software automatically uses the bias conditions at the end of the preceding scan as the initial conditions at the beginning of a **scan** command. If a scan is terminated early because of a stall, a stall restart will cause the initial conditions of the next scan to be different than what the user expected them to be.

- **lateral_mode_perturbation** instructs the software not to make repeated call to the eigenmode solver for the optical mode. Instead, the initial optical mode shape and refractive index perturbations are used to update the effective index.

- **linear_matrix** instructs the software to make a single linear solver step inside the Newton method. It is ***strongly*** recommended to limit the size of the bias step in the **scan** command so that the initial guess is close enough to converge to the specified tolerance using that single iteration of the Newton method.

- **damping_step** puts a limit on the size of the solution update between Newton iterations. Smaller values yield slower but more reliable convergence; larger values can converge quicker but may lead to solution oscillations in difficult problems.

  This option is similar (in principle) to many globally convergent Newton algorithms but a fixed limit is used instead of an automatically determined optimal update size.

- **var_tol** is the convergence criterion for the solver variables. Below this threshold, the solution can be considered "stable". Similarly, **res_tol** is the convergence criterion for the equation residues. Below this threshold, the equations can be considered as being "satisfied".

  The overall convergence criterion for a non-linear Newton step is a combination of these two criterion. Depending on the value of **converge_test**, the following conditions need to be met:

  - 1 Either **var_tol** or **res_tol** is satisfied.
  - 2 Both **var_tol** or **res_tol** are satisfied.
  - 3 **var_tol** is satisfied.
  - 4 **res_tol** is satisfied.

  The default value (2) ensures that the solution is both accurate and stable.

- **min_lit** is the minimum normalized light power, used to prevent the solution from accidentally approaching unphysical zero or negative values. It may be useful in transient simulations.

- **rate_damp** is the damping coefficient for the rate equation at steady state. The larger the coefficient, the more stable the solution to the rate equation, but the solution becomes less accurate. For transient simulation, no damping is used.

  It is not recommended to use a large **rate_damp** if a steady state simulation is to be connected to a transient simulation because the large difference between the damped and undamped rate equations may cause convergence problems for the transient step.

- **scale_eqn_n** and **scale_eqn_p** are the scaling factor for the electron and hole continuity equations, respectively.

- **impact_damp** is a small number used to control or damp the impact ionization term. The exponential impact ionization term tends to negative infinity if the local field is zero. This parameter is used to prevent such an infinity.

- **var_fac** is the variable factor used to normalize the carrier densities if **change_variable**=*yes*.

- **ratio_max** is the maximum ratio (or fraction) of change allowed for the carrier density during each Newton iteration. It is effective if **change_variable**=*yes*.

- **density_tol** is a reference value representing a small carrier concentration ($1/m^3$) in the solution when **change_variable**=*yes*. The program uses this value to decide if the carrier concentration is to be considered small numerically.

- **step_decrease** is a factor less then unity used to alter the bias step size if a bias step fails. Similarly, **step_increase** is a factor greater than unity used when the bias step succeeds.

  By default, neither valued is used and the solver heuristically determines the step size based on the suggested number of iterations in **opt_iter** and the actual number of iterations needed to achieve convergence.

- **back_refine_error_scale** is used to scale the error term in the iterative refinement method of the linear solver step. Although this may affect the convergence of the refinement method, this parameter should not normally be used by the end user.

- **max_iter** is the maximum number of nonlinear iterations. If the required number of iterations is greater than this value, the software will give up, automatically reduce the bias step and start the nonlinear iteration again.

- **opt_iter** is an estimate of the optimal number of nonlinear iterations. The software will always try to force convergence to occur within this number of iterations, increasing the bias step if the solution requires few iterations and decreasing it if more are required.

  The optimal value of this parameter can be difficult to set for even the most experienced user: there is always a tradeoff between a large bias steps which may fail and require many retries and small bias steps which converge reliably but take a long time to achieve the desired bias. The default value has produced good results for a large class of problems.

- **stop_iter** is used to examine the convergence trend and, if necessary, prematurely terminate the non-linear iteration before **max_iter** is reached. In many cases, it is preferable to terminate a large bias step which has little chance of reaching convergence and try again with a smaller step.

- **print_flag** controls how much information is printed during program execution.

- **loopback** is the number of loopback iteration used to update numerical equations (such as tunneling, optical wave, heat flow) decoupled from the main drift-diffusion solver after the normal updating procedure. Use of this parameter will increase the self-consistency but requires more computation time.

- **gainsearch_num** is the number of points used in search the lateral modal gain peak in a laser diode simulation. The simulator performs a uniform search of modal gain peak within the wavelength range specified by the **init_wave** statement. This initial search is followed a golden section search to refine the results; this is skipped if the input value is negative.

- **iter_step** is the number of GMRES iteration used if the CG method is specified in **mf_solver**.

- **mf_solver** determines which linear solver is used at every iteration of the non-linear Newton solver:

  - 0 Conventional sparse matrix solver: slow but reliable.
  - 1 Combined GMRES iterative method with multi-frontal LU decomposition.
  - 2 Multi-frontal LU decomposition with iterative refinement. Somewhat slower than GMRES but more reliable.
  - 3 Parallel multi-threaded multi-frontal solver. In newer version of the software, the number of threads is determined automatically based on the number of available processing cores. See also **parallel_linear_solver** for alternate solver accessible with this option.

  - 4 A parallel solver which can make use of graphical processing units (GPUs). This is built using the CUDA™ technology from NVidia®.
    Note that when using the GPU solver, **parallel_linear_solver** must not be used as it forces **mf_solver**=*3*.

- **mf_md_flag** takes 0 or 1, indicating different versions of minimum degree reordering method for the multi-frontal sparse solver.
- **mf_pivot_update** is the pivotal matrix size to update the F matrix from the C-matrix. For details, please see Ref: "An unsymmetric-pattern multifrontal method for sparse LU factorization", T. A. Davis and I. S. Duff, Tech. Report TR-94-038, University of Florida, November, 1994.

Examples

```
newton_par  damping_step=2 var_tol=1.e-5 res_tol=1.e-4  &&
  max_iter=40  opt_iter =12 stop_iter=10  print_flag=2  &&
```

To take the defaults,

```
newton_par  print_flag=3
```

22.497 no_auto_workfunction

parameter	data type	values [defaults]
mater_label	char	
mater	intg	[1]

Prior to version 2013 of the software, the **affinity** defined in a metal macro was used at heterojunctions so that combined with the default thermionic emission boundary a Schottky contact was effectively formed. Starting with the 2013 version, the work function of metals is automatically adjusted to match the Fermi level of the neighbouring highly-doped region and form an ohmic contact.

no_auto_workfunction is used to recover the previous behavior by disabling this automatic adjustment.

Parmaters

mater is the material number of the macro affected by this command. If a label has previously been defined for this material, **mater_label** may be used instead.

22.498 nonlocal_path

parameter	data type	values [defaults]
y_start_label	char	
y_end_label	char	
model	char	[use_mobility], use_tau
carrier_type	char	[electron], hole
xrange	realx2	(um)
yrange	realx2	(um)
mobility	real	[0.1] ($\frac{m^2}{Vs}$)
tau	real	[1.e-12](s)
mean_free_path	real	[0.01] (um)
zseg_num	intg	[1]
field_dep_model_id	intg	

nonlocal_path defines a non-local transport path between two arbitrary points. See also **nonlocal_transp_model** which controls additional parameters for the non-local transport path.

Parameters

- **y_start_label y_end_label** are y-position labels defining the non-local transport path. Absolute coordinates can also be used with **xrange** and **yrange**.
- **model** changes the way the current along the non-local path is computed.
  - *use_mobility* uses an equivalent mobility model based on the value of **mobility**.
  - *use_tau* uses an equivalent lifetime based on the value of **tau**.
- **carrier_type** determines the type of carrier which travels along the non-local path.
- **mean_free_path** scales the current depending on the path length, following a Drude-like approximation.
- **zseg_num** is the z-segment number in which the non-local path is located.
- **field_dep_model_id** is a label identifying a matching **nonlocal_transp_model** statement.

Examples

```
define_symbol symbol=%mob value=1.
define_symbol symbol=%mfp value=0.1

start_loop symbol=%i value_from=1 value_to=25
integer_func symbol=%k value_from=0 value_to=24

nonlocal_path y_start_label=active%k_end y_end_label=inj%i_start  &&
  model=use_tau tau=1.e-13
nonlocal_path y_start_label=inj%i_start y_end_label=inj%i_end  &&
  mobility=%mob mean_free_path=%mfp field_dep_model_id=1

....

end_loop

nonlocal_transp_model field_dep_file=field_dep_mob.txt  &&
 field_dep_model_id=1
```

22.499 nonlocal_transp_model

parameter	data type	values [defaults]
qcl_mobility_per_period	char	yes,[no]
qcl_mfp_per_period	char	yes,[no]
use_density_matrix_model	char	yes,[no]
field_dep_file	char	[field_dep_mob.txt]
field_dep_model_id	intg	

nonlocal_transp_model defines additional models for the transport properties of non-local transport paths.

Parameters

- **field_dep_file** is a user-specified text file containing the field dependence of the mobility for the non-local path.
- **qcl_mobility_per_period** determines if the equivalent mobility used in **nonlocal_path** is proportional to the number of QCL periods.

- **qcl_mfp_per_period** determines if the mean free path used in **nonlocal_path** is proportional to the number of QCL periods.
- **field_dep_model_id** is a label identifying a matching **nonlocal_path** statement.
- **use_density_matrix_model** uses a relationship between the field and the non-local transport velocity that is derived from density matrix theory[135].

Examples

See **nonlocal_path**.

22.500 nonlocal_transp_region

parameter	data type	values [defaults]
region_label	char	[void]
point1_near	realx2	(um)
point2_near	realx2	(um)
distance1	real	[0.01](um)
distance2	real	[0.01](um)
ref_interface_mater	intgx2	
zseg_num	intg	[1]

nonlocal_transp_region is used to define a region across which non-local transport (such as tunneling) can occur from one side of the region to another, relative to a material interface. This is useful when such a region is not rectangular or when we wish to define a region relative to a irregular material interface. It is preferred that the material interface follows a line that is approximately straight so that the direction of the non-local transport can be certain. Please refer to Fig. 22.21 for details.

Parameters

- **region_label** is a user-specified label identifying this region in other commands.
- **point1_near** and **point2_near** are two reference points near the reference material interface and which are used to define the extent/width of the non-local transport region.

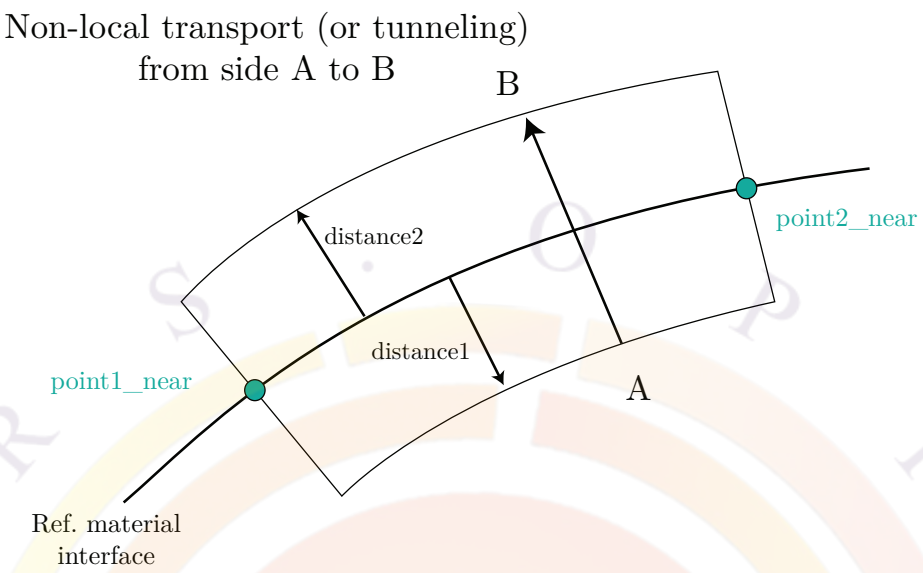


Figure 22.21: Schematic to explain the definition of a non-local transport region

The position does not need to be exact: the software will try to locate nearby mesh points on the specified interface.

- **distance1** and **distance2** are perpendicular distances away from the interface which determine the start and end points of the non-local transport region.

  Positive values are as defined in Fig. 22.21 but negative values may also be entered to put some end points on the opposite side of the interface.

- **ref_interface_mater** is a pair of material numbers (defined in .mater files) used to identify the a material interface. This interface is then used as a reference to define the non-local transport region.

- **zseg_num** is the z-segment number for 3D simulation.

Examples

```
nonlocal_transp_region region_label=%junc1 &&
   point1_near=(0. 11.16) point2_near=(6. 11.16) ref_interface_mater=(3 4) &&
   distance1=0.019 distance2=-0.001
```

The above command would define a region some distance from a reference interface between material 3 and 4.

22.501 norm_field

parameter	data type	values [defaults]
(see) material_par		

The material parameter **norm_field** is added to the list of parameters in the material library. It is used if “n.gaas" is chosen as the type of field-dependent mobility, as the normalizing field (F_{0n}) (see the User’s Manual). It allows the user to specify more accurate electron mobility as a function of field in III-V or II-VI materials, where the Gamma to L and Gamma to X bands cause the velocity vs. field relation to exhibit a negative slope.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.502 ohmic_junction

parameter	data type	values [defaults]
between_mater	intgx2	

The statement **ohmic_junction** may be used to set the current transport model across a heterojunction to an ohmic model: that is, the current is proportional to the carrier density times the gradient of quasi-Fermi levels. By default, thermionic emission model is used for all heterojunctions. We may call this model a Schottky type junction model because it is capable of rectifying the current by promoting it in the forward direction.

Parameters

- **between_mater** is used to specify two material numbers representing two materials between which the heterojunction is forced to behave like a ohmic junction. Like in the usual drift-diffusion model, the current is expressed as proportional to the local carrier density times the gradient of the quasi-Fermi levels.

Examples

```
ohmic_junction between_mater=[1 2]
```

The above statement forces the heterojunction between materials 1 and 2 to behave like an ohmic junction. The material numbers can be found in the .mater files.

22.503 oled_control

parameter	data type	values [defaults]
wavelength	real	(μm)

The statement **oled_control** activates the organic LED (OLED) model. Its use is similar to statement **led_control**.

- **wavelength** is the estimated peak emission wavelength in microns.

Example(s):

```
oled_control wavelength=0.51
```

22.504 optic_coating

parameter	data type	values [defaults]
spectrum_file	char	[void]
thickness	real	[0.01] (μm)
real_index	real	[3.2] (μm)
imag_index	real	[0.] (μm)
segment_from	real	[-1.e7] (μm)
segment_to	real	[1.e7] (μm)
layer_num	intg	[1]
segment_num	intg	[1]

optic_coating defines an optical coating that affects the optical pumping of the device but is not part of the FEM mesh solved by the software.

See **front_reflection** for an overview of the relevant model but take care not to combine the two statements.

Parameters

- **spectrum_file** defines a text file containing the n, k optical constants of the coating.
- **thickness** defines the optical coating thickness.
- **real_index** and **imag_index** define fixed n, k values for the coating. It is preferable to use **spectrum_file** if a broadband light spectrum is used (e.g. solar cells).
- **layer_num** is used to define a multi-layer coating. The layers are numbered from the starting point of the light path to the ending point. For example, if the light is incident from the top, then layer number 1 is the topmost layer.
- **segment_num** is used to define regions in the x-direction which have different coatings. This is analogous to defining columns in the layer file and should not be confused with the segment number used to define the z-mesh in 3D simulations.
- **segment_from** and **segment_to** work in conjunction with **segment_num** to define the x-range where a specific coating is applied. These parameters should not be confused with the z-segment range used to define the position of mesh planes in a 3D simulation.

Examples

```
optic_coating  thickness=0.147 real_index=2.05 imag_index=0.
```

22.505 optical_axis

parameter	data type	values [defaults]
ref_angle	real	[0.] (degrees)
aniso_index_factor	real	[1.]
aniso_index_polar	intg	[0]

optical_axis (previously called **polarization**) is used to specify the orientation of the optical axis with respect to the waveguide. Note that with the 2013 version, material properties which depend on the TE/TM orientation of the electric field are both calculated automatically by the software.

This command enables the use of birefringent materials; see **delta_real_index_caxis**

Note that using this setting overrides the TE/TM mode setting in the **active_reg** statement.

Parameters

- **angle** is the angle of the electric field with respect to the QW plane. For a TE waveguide, this angle is 0; for TM, it is 90 degrees.

- **aniso_index_factor** is used in birefringent materials and defines the projection of the extraordinary axis on the polarization axis (TE or TM) being considered.

- **aniso_index_polar** applies the index difference on the extraordinary axis to one of the polarizations (1=TE, 2=TM). If zero (the default), this index difference is ignored.

Code Notes

```
sn=rldx(node)+delindx(ku)
if(janiso_index_polar.gt.0) then
  if(jpolar.eq.janiso_index_polar) then
    sn=sn+delta_rldx_caxis(node)*aniso_index_factor
  endif
endif
```

Examples

```
optical_axis angle=45.
```

22.506 optical_field

parameter	data type	values [defaults]
profile	char	[gaussian], effective_index
y.data	char	[void]
update	char	no
standing_wave	char	no
uniform_stdwave	char	no
gainguiding_stdwave	char	no
x_prof	realx4	x1,x2,dx1,dx2 (μm)
y_prof	realx4	y1,y2,dy1,dy2 (μm)
x_segment	intg	40

optical_field defines the solution of the optical field for the lateral mode of a waveguide. This statement can be used in conjunction with **multimode** if multiple lateral modes are present.

This statement has two main modes of operation: it can either define a basic Gaussian-shaped optical profile or use an 1D effective index method to solve the mode profile.

Application Notes

Because of the restrictions in the **optical_field** algorithm, this command is the preferred lateral mode solver for 1D simulations (without lateral mesh variations) and broad-area devices (with many lateral mesh points but without a lateral structural variation).

For devices with a lateral structural variation, **direct_eigen** should be used. Since the algorithm used inside **optical_field** can only solve 1D problems, the effective index of 2D structures is found using successive cuts along the y and x directions, with the x direction assumed to have a slowly-varying refractive index. Such an approach is significantly less accurate than a direct solution of the 2D wave equation.

The iterative SOR method in **use_sor** can also be used to refine the accuracy of the solution provided by **optical_field**. However, the SOR algorithm is considered obsolete and Crosslight *strongly* recommends using **direct_eigen** in 2D cases.

Conversely, **direct_eigen** is not recommended for use in the situations where **optical_field** excels; please refer to the application notes section of that command. Note that these two commands should not be used together.

Parameters

- **profile** is the profile of the user-defined optical field:
  - *gaussian* instructs the software to use a roughly Gaussian shape, based on the settings in **x_prof** and **y_prof**.
  - *effective_index* activates a 1D solution of the effective index as explained above, with the x direction as the slowly-changing index direction.
- **y.data** is a user-supplied data file for the optical field distribution in the y-direction. This allows the optical field in the y-direction to be imported from a data file instead of generated by the self-consistent optical field solver.

  This method is of particular interest in multimode simulations where we wish for all higher-order modes to be in the x direction while the y direction stays in its fundamental mode.
- **update** determines if the optical mode defined by this statement is updated at all bias steps. If set to *no*, the mode is only updated at the start of each **scan** command.
- **x_prof** specifies the profile of the Gaussian-shaped mode defined in **profile**. This parameter is defined as a set of four values: x_1, x_2, dx_1, dx_2. The mode is defined as constant in the $[x_1, x_2]$ interval and decays with Gaussian tails with standard deviations dx_1 and dx_2 on either side.
- **y_prof** is identical to **x_prof** but defines the beam shape in the y direction.
- **standing_wave** is used to turn on an analytical model for the lateral standing wave in a broad-area device. The mode in the x direction will vary according to a $sin(\beta(x - x_0))$ shape.

  This option is useful in devices where the lateral dimension is in order of several hundred microns and the lateral mesh spacing is coarser than the standing wave period.
- **uniform_stdwave** forces the use of a uniform magnitude in the x direction instead of a sine wave when **standing_wave** is used.
- **gainguiding_stdwave** partially overrides the normal decoupling of the lateral (xy) and longitudinal (z) wave in PICS3D edge-emitter simulations. The normal behavior is to use the modal gain in the longitudinal propagation constant so that the whole mode sees the same amplification factor. When this parameter is *yes* however, a small correction to the round-trip gain is added so that regions with a higher local gain (i.e. "hot spots") in the lateral standing wave have a stronger amplification than the rest of the device.

- **x_segment** is the number of points used in the x direction when using the 1D effective index model. This number can be kept small for 1D cases but should be increased for broad-area devices in order to sample the lateral standing wave.

Examples

```
optical_field profile=effective_index update=yes
```

22.507 organic_exciton_diff

parameter	data type	values [defaults]
add_oxd_memory	real	[0.] (MB)
heteroj_scrn_factor	real	[0.1]
el_frac_non_active	real	[0.9]

organic_exciton_diff enables organic exciton diffusion model. The exciton diffusion across a heterojunction follows the thermionic emission model: excitons with thermal energy may emit from barrow bandgap to larger bandgap material. Similar to thermionic emission model in electrons, the following flux density is used for exciton (S) flowing from wider bandgap material 2 to material 1:

$$J = V_{th}\{S_2 - S_1 exp[-\gamma_s(E_{g2} - E_{g1})/k_B T]\} \tag{22.87}$$

where $V + th$ is thermal velocity of the exciton. γ_s is a screening factor less than unity. Since excitons are charge neutral, they should experience less crystal field potential than charged particles and thus the bandgap difference they experience should be less than electrons or holes. The screening factor is used to denote this effect.

- **add_oxd_memory** is used to add memory to the sparse solver for the diffusion equations.
- **heteroj_scrn_factor** is the heterojunction screening factor above.
- **el_frac_non_active** is the fraction of excitons in non-active (or undoped layers) that are eventually harvested by dopants.

Example(s):

```
organic_exciton_diff el_frac_non_active=0.85
```

The above statement directs 85 percent of excitons in undoped (non-active) layers be collected by dopants to emit light.

22.508 outer_section

parameter	data type	values [defaults]
core_radius	real	(um)
indexi,i=1..9		
sec_num	intg	[1]

outer_section is used to inform the program of the termination of a VCSEL section with an insulator (e.g. outside air). It is only used in .vcsel files.

Parameters

- **core_radius** is the radius of core region, which indicates to the simulator that this VCSEL section is terminated by an insulator layer on the outside.
- **indexi**,i=1..9 is the outer refractive index matching one or more DBR layers in the core region of the VCSEL.
- **sec_num** is a section number.

Example(s)

```
outer_section core_radius=6.0 sec_num=7
```

22.509 output

parameter	data type	values [defaults]
sol_outf	char	

output defines the base name of the output data files; extensions are added to this name automatically to account for the data set set number and the type of output information the file contains.

Examples

```
output sol_outf=bulk2d.out
```

22.510 output_suprem_mesh

This statement instructs the layer.exe to generate SUPREM mesh declarations instead of the .geo format used in APSYS, LASTIP and PICS3D. It is outside the scope of this manual.

22.511 ox_dopant_el_transfer

parameter	data type	values [defaults]
transfer_from_host	char	[no]
transfer_fraction	real	[0.]
efield_ref	real	[1.e8] (V/m)
mater_from	intg	[1]
mater_to	intg	[1]
dopant_index_from	intg	[1]
dopant_index_to	intg	[1]

ox_dopant_el_transfer is used to enable the transfer of exciton energy from one type of dopant to another type. This reveals itself in the shift of color from shorter to longer wavelengths.

The formula used a factor $factor = 1 - frac - F/F_r$ on the shorter wavelength dopant EL where frac is the transfer fraction. F is the electric field and F_r is a reference field. Similarly a factor $factor = 1 + frac + F/F_c$ is used for the longer wavelength EL.

- **transfer_from_host** would let the energy be transferred from the host instead of from a dopant.
- transfer_fraction is the fraction of exciton transferring from one type of dopant to another.
- efield_ref is the reference electrical field above.
- mater_from is the material index from which energy transfer originates.

- mater_to is the material index to which energy transfer arrives.
- dopant_index_from is the dopant index (for multiply dopants) from which energy transfer originates.
- dopant_index_to is the dopant index (for multiply dopants) to which energy transfer arrives.

Example(s):

```
ox_dopant_el_transfer mater_from=5 mater_to=4 &&
  transfer_fraction=0.2 efield_ref=-4.e8
```

22.512 ox_el_weight

parameter	data type	values [defaults]
(see) material_par		

The material statement **ox_el_weight** is used to define the electroluminescent spectrum weight for the host organic material. This is useful when there is dopant in the system so that one has to define the relative contribution from host and the dopant.

For EL spectrum model when there is dopant, two approaches are used, depending on whether exciton diffusion equations are solved. If exciton transport is solved by **organic_exciton_diff**, dopant and host spectra at each mesh point are simply mixed by local exciton density, divided by their respective life times, and weighted by this factor. If exciton transport is not considered, their respective bimolecular recombination rates are used along with this factor to determine the relative contributions.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
ox_el_weight value=0.8 mater=1
```

22.513 ox_exciton_eg

parameter	data type	values [defaults]
(see) material_par		

The material statement **ox_exciton_eg** is the exciton bandgap of the organic material.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

```
Example(s)
ox_exciton_eg value=1.9 mater=2
```

22.514 ox_extern_spectrum

parameter	data type	values [defaults]
(see) material_par		

The material statement **ox_extern_spectrum** is used to define the electroluminescent (EL) spectrum of the host material imported from an external source. This may be based on experimental data or on a different type of model. If table format is used, the program expects a 2-column uniformly spaced data with first column being the wavelength in micron meters. The unit of the EL is arbitrary and the program will normalize it.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

```
Example(s)

ox_extern_spectrum variation=table
table(wavelength)
  0.488524E+00  0.309522E-01
  0.499601E+00  0.421672E-01
  ...
end_table
```

22.515 ox_gaussian_sdj

parameter	data type	values [defaults]
(see) material_par		

The material statement **ox_gaussian_sdj** is the standard deviation (in unit eV) of the Gausian function used to broaden the exciton transition energy. If more than one Gaussian function is used in different energy range, this parameter works with parameter **ox_gaussian_div(k=1...)** as follows: For example, the spectrum is broadened with N parameters: **ox_gaussian_sd(j=1...N)**. Then **ox_gaussian_sd(j)** acts on an energy range defined between **ox_gaussian_div(j-1)** and **ox_gaussian_div(j)**. Please note thta **ox_gaussian_div(0)** equals zero and **ox_gaussian_div(N)** equals infinity.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

```
Example(s)
ox_gaussian_sd1 value=0.12 mater=2
```

22.516 ox_gaussian_divj

parameter	data type	values [defaults]
(see) material_par		

The material statement **ox_gaussian_divj** is the division in energy (in unit eV) for different Gausian function to be used to broaden the exciton transition energy. If more than one Gaussian function is used in different energy range, this parameter works as follows: For example, the spectrum is broadened with N Gaussian function of parameters: **ox_gaussian_sd(j=1...N)**. Then **ox_gaussian_sd(j)** acts on an energy range defined between **ox_gaussian_div(j-1)** and **ox_gaussian_div(j)**. Please note thta **ox_gaussian_div(0)** equals zero and **ox_gaussian_div(N)** equals infinity.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

```
Example(s)
ox_gaussian_div1 value=2.21 mater=2
```

22.517 ox__hopping__energy

parameter	data type	values [defaults]
(see) material_par		

The material statement **ox__hopping__energy** is the hopping integral (in units of the vibrational quanta) describing the nearest-neighbor interaction of the exciton (as quasiparticle) in a tight-binding model.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material__par** in section 22.456.

```
Example(s)
ox_hopping_energy value=0.6 mater=1
```

22.518 ox__life__field__dependence

parameter	data type	values [defaults]
increase	char	[yes]
triplet	char	[yes]
mater_label	char	
pf_lifetime_expo	real	[0.5]
efield0_pf	real	[2.e8] (V/m)
mater	intg	[1]

ox__life__field__dependence is used to enable a field-dependent model for the organic exciton lifetime. As with many other field models for organic semiconductors, we assume a Poole-Frenkel-like dependence:

$$\tau_{ox} = \tau_0 e^{\pm(F/F_{t0})^{\alpha}} \tag{22.88}$$

Parameters

- **increase** determines whether to choose positive or negative sign for the above formula.
- **triplet** indicates whether the model applies to singlet or triplet diffusion.
- **pf_lifetime_expo** is the exponent α in the above formula.
- **efield0_pf** is the threshold field F_{t0} in above formula.
- **mater** is the material number affected by this statement. If a label has previously been defined as an alias, **mater_label** may be used instead.

Examples

```
$ field-dep lifetime needed to tune blue color change
ox_life_field_dependence increase=yes triplet=no &&
  pf_lifetime_expo=0.5 efield0_pf=1.e8 mater=2
```

22.519 ox_peak_abs

parameter	data type	values [defaults]
(see) material_par		

The material statement **ox_peak_abs** is the absorption peak in units of 1/m of the organic material.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

```
Example(s)
ox_peak_abs value=1.e7 mater=2
```

22.520 ox_vib_quanta

parameter	data type	values [defaults]
(see) material_par		

The material statement **ox__vib__quanta** is the organic molecular vibrational quanta, or phonon energy unit, in eV.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material__par** in section 22.456.

```
Example(s)
ox_vib_quanta value=0.16 mater=1
```

22.521 ox__xp__coupling

parameter	data type	values [defaults]
(see) material__par		

The material statement **ox__xp__coupling** is the exciton-phonon (XP) coupling coefficient. It is a dimensionless number with its square equal to the vibrational relaxation energy (Franck-Cordon energy) divided by the vibrational quanta (or molecular phonon quanta).

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material__par** in section 22.456.

```
Example(s)
ox_xp_coupling value=1.0 mater=1
```

22.522 oxd__diff__length

parameter	data type	values [defaults]
(see) material__par		

The material statement **oxd__diff__length** is used to define the exciton diffusion length in micron meters for organic material.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
oxd_diff_length value=1.e-2 mater=2
```

22.523 oxd_lifetime

parameter	data type	values [defaults]
(see) material_par		

The material statement **oxd_lifetime** is used to define the exciton diffusion life time in seconds for organic material.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
oxd_lifetime value=1.e-6 mater=2
```

22.524 oxd_quench

parameter	data type	values [defaults]
mater_label	char	
tau	real	[1.0] (s)
elec_factor	real	[0.0] (m^3/s)
hole_factor	real	[0.0] (m^3/s)
biex_factor	real	[0.0] (m^3/s)
mater	intg	[1]

oxd_quench is used in organic semiconductor simulations to define the quenching term τ_q used in the exciton diffusion equation (Eq. 14.73) for singlets.

The quenching term depends on several terms:

$$\frac{S}{\tau_q} = \frac{S}{\tau} + AnS + BpS + \frac{1}{2}CS^2$$

where n,p and S are the free electron, free hole and exciton concentrations. The other terms are defined below.

Parameters

- **tau** is the fixed quenching term τ defined above. It is usually slow enough that other terms dominate.
- **elec__factor** is A in the above formula.
- **hole__factor** is B in the above formula.
- **biex__factor** is the bi-exciton quenching term C in the above formula. This term is usually zero for singlet states.
- **mater** is the material number affected by this statement. If a label has previously been defined as an alias, **mater__label** may be used instead.

Examples

```
oxd_quench tau=10.e-6 mater=2
```

22.525 oxd2__diff__length

parameter	data type	values [defaults]
(see) material__par		

The material statement **oxd2__diff__length** is used to define the diffusion length of the 2nd type (triplet) of exciton in micron meters for organic material.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material__par** in section 22.456.

```
Example(s)

oxd2_diff_length value=1.e-2 mater=2
```

22.526 oxd2_lifetime

parameter	data type	values [defaults]
(see) material_par		

The material statement **oxd2_lifetime** is used to define the diffusion life time of the 2nd type of exciton (triplet) in seconds for organic material.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

Example(s)

```
oxd2_lifetime value=1.e-6 mater=2
```

22.527 oxd2_quench

oxd2_quench is identical to **oxd_quench** except that it defines the quenching term for triplet exciton states.

Unlike singlets, the triplet-triplet bi-exciton quenching term is usually not zero.

22.528 para_extract

parameter	data type	values [defaults]
file	char	[extract.txt]
append_data	char	[no]
type	char	[intercept]
fit_hori_from	real	[-1.e49]
fit_hori_to	real	[1.e49]
fit_vert_from	real	[-1.e49]
fit_vert_to	real	[1.e49]
vert_intercept	real	
hori_intercept	real	
xy_hori_from	real	[-1.e49]
xy_hori_to	real	[1.e49]
xy_vert_from	real	[-1.e49]
xy_vert_to	real	[1.e49]
xy_product_scale	real	[1.]
add_vert_intercept	real	[0.]
add_hori_intercept	real	[0.]
scale_fit_slope	real	[1.]
scale_fit_inv_slope	real	[1.]
vt_search_range	realx2	[0 1.e49], 0. 10,1. 3.,-10. 0.

para_extract is used in .plt file in order to extract data curve characteristics generated by the next **plot_scan** statement issued after this command. Only one set of parameters can be extracted from each figure so multiple sets of **para_extract** and **plot_scan** commands may be needed to extract all of the relevant information from a curve.

Parameters

- **file** is the output file name where the extracted parameters are saved (in ASCII text format). An existing file will be overwritten unless **append_data** is used; this point is of special importance when generating series and doing Design of Experiments analysis as all of the extracted parameters need to be extracted to the same output file.
- **type** is the type of data extracted from the curve:
  - *intercept* - finds the intercept point with the x/y axis or with an arbitrary vertical/horizontal line.

  - *fit_line* - fits a set of data to a straight line.
  - *xy_product_search* - searches for an interpolated data point which maximizes the x*y product.
  - *hori_max_search* - searches for a data point with maximum x-value.
  - *vert_max_search* - searches for a data point with maximum y-value.
  - *vt* - used in Id-Vg transistor curves, searches for the threshold voltage.

  Please note that in the above, x & y refer to the horizontal and vertical axes of the graph and not spatial coordinates.

- **fit_hori_from** and **fit_hori_to** is the horizontal fitting range used to fit data to a straight line. A matching set of parameters for the vertical fitting range is given by **fit_vert_from** and **fit_vert_to**. These parameters are used when **type**=*fit_line*.

- **vert_intercept** defines the x value of an arbitrary vertical line. It is used when **type**=

- *intercept*.

- **hori_intercept** defines the y value of an arbitrary horizontal line. It is used when **type**=

- *intercept*.

- **xy_hori_from**, **xy_hori_to**, **xy_vert_from** and **xy_vert_to** define the search range in the horizontal and vertical directions when **type**= *xy_product_search*.

- **xy_product_scale** may be used to artificially scale the maximum x*y product found during the extraction.

- **add_vert_intercept** may be used to artificially offset the vertical line intercept result. Similarly, **add_hori_intercept** may be used to offset the horizontal line intercept result.

- **scale_fit_slope** may be used to artificially scale the slope when **type**=*fit_line*. Similarly, **scale_fit_inv_slope** may be used to artificially scale the inverse of the slope.

- **vt_search_range** is the horizontal (voltage) search range used to find the threshold voltage on a Id-Vg plot when **type**=*vt*.

Examples

The following statement fits the L-I curve above 1 mW to a straight-line so that threshold current can be found as an intercept with the horizontal axis; it also finds the drive current at 8 mW. Results of the parameter extraction is written to file extract.txt by default.

```
para_extract type=fit_line fit_vert_from=1 &&
  hori_intercept=8.
plot_scan scan_var=current_1 variable=laser_power
```

The following command finds the maximum power from an I-V curve, a type of analysis often used in solar cell simulations:

```
para_extract type=xy_product_search xy_hori_from=0 xy_vert_from=0
plot_scan scan_var=voltage_1 variable=current_2
```

22.529 parallel_linear_solver

parameter	data type	values [defaults]
solver	char	[mumps], pardiso
pardiso_iterative	char	[void], cgs, outer_loop
mumps_workspace	intg	[] (percent)

parallel_linear_solver may be used to specify alternate parallel sparse Newton solvers. Using this statement also forces **mf_solver**=*3* in **newton_par**.

Examples

- **solver** is used to specify the Newton solver. Available choices are:
  - MUMPS: public domain solver available at http://graal.ens-lyon.fr/MUMPS/
  - PARDISO: originally developed at http://www.pardiso-project.org/, the version included in our software is licensed as part of the Intel Math Kernel Library ™.

  The PARDISO solver has poor convergence properties for the problems solved in Crosslight and is not recommended.

- **pardiso_iterative** can be used to turn on an iterative solver method in PARDISO and specifies the preconditioning algorithm. The sparse matrices encountered in Crosslight are badly conditioned and do not generally respond well to iterative methods so these options are not recommended.

- **mumps_workspace** exposes an internal setting of the MUMPS solver and allows users direct control over the amount of extra memory used for fill-in: by default the MUMPS solver uses 20% of the sparse matrix size but this may prove insufficient in certain problems.

  Please note that as of the 2015 version, the software will analyse MUMPS error messages and automatically attempt to retry solver calls that fail due to a lack of memory. The main benefit of setting this variable explicitly is to avoid the extra time associated with this retry.

22.530 passive_3d

parameter	data type	values [defaults]
index	real	
intern_loss	real	(1/m)
carrier_conc	real	$(1/m^3)$
sec_num	intg	

In the coupled RTG method of PICS3D, optical properties for the longitudinal propagation are usually provided through interpolation of mesh plane data from the electrical problem. However, it is also possible to define optical parts of the cavity which are not meshed and are purely passive: external cavities, fiber pigtails, etc... **passive_3d** is used to define these regions.

Parameters

- **index** is the refractive index of the passive section.

- **intern_loss** is the internal loss of the passive section.

- **carrier_conc** provides a reference carrier concentration for plotting only. It is otherwise unused in current versions of the software.

- **sec_num** is the section number (or section index) of the corresponding optical section.

22.531 passive_carr_loss

parameter	data type	values [defaults]
mater_label	char	
ncarr_loss	real	[0] m^2
pcarr_loss	real	[0] m^2
two_photon_loss	real	[0] (m^2)
two_photon_carr	real	[0] (m^5)
mater	intg	[1]

passive_carr_loss is used to specify in the .sol file the free carrier absorption in regions other than those which have been assigned an active macro; for those regions, equivalent settings are available in **active_reg**. The free carrier loss is expressed as:

$$\alpha = \text{ncarr\_loss} \times (n - n_0) + \text{pcarr\_loss} \times (p - p_0) \tag{22.89}$$

Note that the same functionality can also be achieved using **elec_carr_loss** and **hole_carr_loss**; these statements are meant to be used inside a material macro but may also be used in the .sol as a material parameter override.

It is also important to mention that unlike the gain/index change calculations used in active regions, no attempt is made by the software to reconcile this parameter with the carrier-dependent index change mechanism defined in **index_model**. This command is meant to be used for experimental/empirical coefficients so it is the user's responsibility to provide values which obey the Kramers-Kroenig relationship.

Parameters

- **ncarr_loss** is the coefficient of free carrier absorption for electrons. **elec_carr_loss** may also be used.
- **pcarr_loss** is the coefficient of free carrier absorption for holes. **hole_carr_loss** may also be used.
- **two_photon_loss** is a two-photon absorption term in S^2. **two_photon_loss** may also be used.
- **two_photon_carr** is a free carrier loss coefficient related to the two-photon absorption process: these two effects combine to create an S^3 loss term. See **two_photon_carr** for details.

- **mater** is the material number. If a label has previously been defined for this material, **mater_label** may instead be used.

Examples

```
passive_carr_loss ncarr_loss=2.e-21 pcarr_loss=1.e-21 mater=1
```

22.532 passive_fiber

parameter	data type	values [defaults]
intern_loss	real	[5.e-5] (1/m)
index_core	real	[1.8]
index_cladding	real	[1.3]
core_radius	real	[8.] (um)
group_index	real	[1.48]
sec_num	intg	[1]
bessel_order	intg	[0]

passive_fiber may be regarded as a special case of **passive_3d**. In addition, fiber scalar modes are computed within the fiber section.

- **intern_loss** is the internal loss of the fiber section.
- **index_core** is the core index of the fiber.
- **index_cladding** is the cladding index of the fiber.
- **core_radius** is the core radius of the fiber.
- **group_index** is the group index of the fiber.
- **sec_num** is the section number of the fiber section.
- **bessel_order** is the Bessel function order used to represent the scalar lateral modes within the fiber section. The fundamental mode order is zero.

Example(s)

```
passive_fiber intern_loss=4.6e-5 sec_num=4 &&
  index_core=1.5 index_cladding=1.496 core_radius=9.
```

22.533 pc_led_model

parameter	data type	values [defaults]
top_emission	char	[yes]
surface_y_label	char	[void]
rectangle	char	[no]
surface_y	real	(um)
grating_height	real	[0.1](um)
grating_dia_ratio	real	[0.2](dia/period)
x_range	realx2	(-9999. 9999.)(um)
height_range	realx2	[0.1 1.](um)
surface_mater	intg	[1]
model_points	intg	[10]

The statement **pc_led_model** is used to define model parameters for photonic crystal LED (PhCLED) model which is treated as a post-processor module.

- **top_emission** indicates whether the LED emits from the top.
- **surface_y_label** is a position label for the material interface of the (air/semiconductor) LED.
- **rectangle** indicates whether the air holes of the PhC is rectangle or hexagonal.
- **surface_y** is the absolute y-coordinate for the material interface of the (air/semiconductor) LED.
- **grating_height** is the height of the air-hole grating.
- **grating_dia_ratio** this ratio of the air-hole diameter over the period (or the real space lattice vector constant) of the photonic crystal.
- **x_range** is range in x over which the current spreading effect is taken into account for the calculation of distribution of extracted power emission.
- **height_range** is the range of air-hole height over which the LED emission properties are to be evaluated.
- **surface_mater** is the semiconductor material number over which the air-holes of the photonic crystal are to be made.
- **model_points** is the number of spacing divisions over the **x_range** for the calculation of spatial variation of LED properties.

Example(s):

```
pc_led_model top_emission=yes surface_y_label=air_hole &&
  grating_height=0.3 grating_dia_ratio=0.5 surface_mater=1 &&
  model_points=20 x_range=(50 200)
```

22.534 pf_model_setting

parameter	data type	values [defaults]
min_shift_vt	real	[0.03] (eV)
bias_dependent	char	[no]

pf_model_setting controls various aspects of the Poole-Frenkel incomplete ionization model.

Parameters

- **min_shift_vt** is used to adjust the Poole-Frenkel formula.

  The standard form of Sec. 5.1.4 indicates that the ionization energy would shows no shift at zero bias; however, the derivative vs. field also becomes infinite at this point which causes problems for the Jacobian used to solve the Drift-Diffusion equations self-consistently.

  To solve the problem, the following formula is used instead:

$$\Delta E_{PF} = \sqrt{q\frac{F+F_0}{\pi\epsilon_0\epsilon}} - \sqrt{q\frac{F_0}{\pi\epsilon_0\epsilon}} \tag{22.90}$$

  where **min_shift_vt** is the second term on the right-hand side.

- **bias_dependent** determines if the energy shift is calculated only once (at equilibrium) or updated throughout the simulation.

22.535 piezo_d11

piezo_d11 and related commands are a set of parameters use to define the piezoelectric properties in SAWAVE.

It is important to note that these parameters are different from the piezoelectric parameters defined in **e15_bulk** and other related commands. Using the strain-charge form, the two sets of coefficients are related by:

$$d_{ij} = \frac{\partial D_i}{\partial T_j} \tag{22.91}$$

$$e_{ij} = \frac{\partial D_i}{\partial S_j} \tag{22.92}$$

where D is the electric displacement, T is the stress and S is the strain. The two sets of commands therefore differ in that e_{ij} shows the effect of built-in strain while d_{ij} shows the effect of external stress: in SAWAVE, the acoustic wave propagation is described by such a stress term.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.536 piezo_d22

See Sec. 22.535.

22.537 piezo_d33

See Sec. 22.535.

22.538 piezo_d31

See Sec. 22.535.

22.539 piezo_d32

See Sec. 22.535.

22.540 piezo_d24

See Sec. 22.535.

22.541 piezo_d15

See Sec. 22.535.

22.542 plot_1d

parameter	data type	values [defaults]
data_file	char	[vttek]
variable	char	(see list)
n_variables	charxn	
integration	char	[no]
x_from_label	char	[void]
y_from_label	char	[void]
x_to_label	char	[void]
y_to_label	char	[void]
math_oper	char	[void]
nonloc_esc_cap_path	char	yes,[no]
nonloc_fly_over_path	char	yes,[no]
from	realx2	
to	realx2	
xrange	realx2	
yrange	realx2	
integration_start	real	[0.0](um)
integration_length	real	[1.e5](um)
qw_wave_ht	real	[0.1] (eV)
var_num	intg	[1]
mode_index	intg	[1]
trap_index	intg	[1]
qw_wave	intg	[0] 1
cond_subband	intg	[1] 2
val_subband	intg	[1] 2 3

plot_1d is a post-processor statement used to plot data along a 1D cut line. It is similar to **lplot_xy** and **lplot_xyz**. Depending on the 2D/3D nature of the simulation, the following rules should guide the choice of plotting command:

- 2D simulations: use **plot_1d**
- 3D cylindrical simulations with one mesh plane: use **plot_1d**

- xy cut of a 3D simulation: use **lplot_xy**
- z cut of a 3D simulation: use **lplot_xyz**

Parameters

- **data_file** is the name of a text file in which a copy of the plot data will be saved.
- **variable** is the physical quantity to be plotted. A detailed list of available variables is given in App. G.2. Additional data may also be presented for certain variables. For example, plotting "wave_intensity" prints the value of the optical confinement factor.
- **n_variables** is used to plot several variables simultaneously.
- **integration** can be used to tell the solver to perform a numerical integration of the plotted variable. The results of the integration will be printed in the .plt.msg file.
- **x_from_label**, **y_from_label**, **x_to_label**, and **y_to_label** are used to define the beginning and end points of the cut line using position labels. These labels must have been previously defined by using **x_position** or **y_position**. The latter statements are often generated automatically in the layer file preprocessing stage.

  Using position labels has the advantage that the plotting commands will automatically adjust to changes in device geometry. It is no longer necessary to find the absolute coordinates before plotting.
- **math_oper**, if defined, modifies the variable being plotted so it shown on a log10 or power10 scale. Note that some variables are already plotted on a log10 scale by default (e.g. carrier and dopant concentrations) and in some cases, there may be a predefined alternate variable name to plot the same value on a linear scale.
- **from** and **to** are the (x,y) coordinates which determine the beginning and end of the 1D cut line. In the 3D variations on this command, **xy_from** and **xy_to** serve the same purpose. Note that **xrange** and **yrange** also provide an alternate way to define the extent of the cut line.
- **integration_start** and **integration_length** are used to define the integration range when **integration** is used. The two values are defined relative to the abscissa of the 1D cut line.

- **qw_wave_ht** is the height of the quantum wave amplitude as plotted on the band diagram. This parameter is only useful when **qw_wave**= 1.

- **var_num** is the number of variables to be plotted when using **n_variables**.

- **mode_index** specifies the number of the lateral mode for the quantity being plotted. If only the fundamental mode is of interest, the default value should be used.

- **trap_index** identifies the trap (SRH) recombination center.

- **qw_wave** enables plotting of the quantum well wave functions on the band diagram when **qw_wave**= 1. However, the simulation must first have been run using the **self_consistent** statement. It is also necessary to output the QW wave data using **more_output**.

- **cond_subband** is used when **qw_wave**= 1 to indicate which conduction band is being plotted. The main condition band valley (Γ) is equal to 1 while 2 represents the side valley (L, X).

- **val_subband** is used when **qw_wave**= 1 to indicate which valence band is being plotted. Values of 1, 2 and 3 are respectively used to represent the HH, LH and CH bands or their wurtzite equivalents.

- **nonloc_esc_cap_path** and **nonloc_fly_over_path** are helpful when viewing the results of a **q_transport** simulation or any other simulation which has defined non-local current transport paths. When turned on, these parameters add lines to the band diagram that explicitly show the carrier capture/escape and fly-over non-local paths, respectively.

Examples

```
plot_1d variable=elec_conc from=(0.5, 0.) to=(0.5, 1.9)

plot_1d 2_variables=(elec_conc hole_conc) var_num=2 &&
from=(0.5, 0.) to=(0.5, 1.9)

plot_1d variable=recomb_aug from=(0.5 0.) to=(0.5, 3.)
```

22.543 plot_ac_curr

parameter	data type	values [defaults]
data_file	char	[void]
variable	char	[void]
imag_part	char	[no]
input_file	char	[void]
from	realx2	
to	realx2	

plot_1d_ac_curr is used to plot the AC current distribution in a 1D slice. It works with the statement of **ac_voltage** assuming a unit AC voltage signal in the input electrode.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **input_file** is the input file containing the AC analysis results to be plotted.
- **variable** is the variable being plotted. Its value takes the format “type_curr_d", where type may be one of “elec", “hole",“disp" or “total", standing for electron, hole, displacement, and total current components. “d" can take “x" or “y" for directions.
- **imag_part** indicates whether imaginary part of the current is plotted.

  from is the starting point of the 1D slice.

  to is the starting point of the 1D slice.

Examples

```
plot_1d_ac_curr variable=elec_curr_y imag_part=no input_file=test.ac &&
  from=(0.5 0.) to=(0.5 2.5)
```

The above statement plots 1D distribution of real part of AC current from test.ac file.

22.544 plot_2d

parameter	data type	values [defaults]
variable	char	(see list)
data_file	char	
mater_boundary	char	[no]yes
point_ll	realx2	(μm)
point_ur	realx2	(μm)
xrange	realx2	
yrange	realx2	
maxvector_scale	real	[1.]
grid_sizes	intgx2	[20, 20]
level	intg	[10]
mode_index	intg	[1]
trap_index	intg	[1]
overlap_with_mode	intg	

plot_2d is a post-processor statement used to plot structural data on a 2D plane. The software will automatically switch between the vector fields and contour plots according to variable selected.

Other related commands exist and should be used depending on the 2D/3D nature of the original simulation results. The following rules apply:

- 2D simulations: use **plot_2d**
- 3D cylindrical simulations with one mesh plane: use **plot_2d**
- xy plane from a 3D simulation: use **cplot_xy** for contour plots of scalar variables, **splot_xy** for 3D surface plots of scalar variables or **vplot_xy** for vector variables.
- xyz plane from a 3D simulation: use **cplot_xyz** for contour plots of scalar variables, **splot_xyz** for 3D surface plots of scalar variables or **vplot_xyz** for vector variables.

Parameters

- **variable** is the variable to be plotted. See App. G for a full list of available variables.

  data_file is the name of a text file in which a copy of the plot data will be saved.

- **maxvector_scale** is used to scale the maximum length of a 2D vector field. A vector field is plotted by drawing only the vectors of length between the minimum and maximum vector length. Vectors of lengths below the minimum are not plotted. Vectors of lengths above the maximum are plotted with the same vector size. Increasing this parameter generates a plot with many small size vectors. Decreasing this value magnifies the smaller vectors while leaving the large ones with the same size.

- **mater_boundary** is used to indicate if material boundaries are plotted.

- **point_ll** is the lower-left corner point of the 2D plotting window.

- **point_ur** is the upper-right corner point of the 2D plotting window.

- **xrange** specifies a range for the x-axis.

- **yrange** specifies a range for the y-axis.

- **grid_sizes** are the grid sizes of the 2D plot. The physical variable being plotted is interpolated from the original mesh onto this regular grid.

- **level** is the number of divisions in the contour plot.

- **mode_index** is the index of the lateral mode in a multi-lateral mode simulation. If only the fundamental mode is concerned, the default value of one should be used.

- The parameter **trap_index** has been added for SRH recombination, so that the user can specify which trap center is being plotted.

- **overlap_with_mode** will calculate and print the overlap between the mode specified in **mode_index** and the mode specified by this parameter.

Examples

```
plot_2d variable=total_cur xrange=(0. 1.5) yrange=(0. 3.)
```

22.545 plot_3d

parameter	data type	values [defaults]
variable	char	(see list)
data_file	char	[vttek]
integration	char	[no]yes
point_ll	realx2	(μm)
point_ur	realx2	(μm)
view_xrot	real	[0.]
view_zrot	real	[0.]
xrange	realx2	
yrange	realx2	
zrange	realx2	
integration_xrange	realx2	[-1.e4 1.e4](μm)
integration_yrange	realx2	[-1.e4 1.e4](μm)
grid_sizes	intgx2	[20, 20]
mode_index	intg	[1]
trap_index	intg	[1]

plot_3d is a post-processor statement used to plot two-dimensional scalar variable data in a 3D view.

- **variable** is the variable to be plotted. The variable listing is given in App. G.2.
- **data_file** is the file to which the graphic data is written in ASCII format.
- **integration** indicates whether integration is performed for the physical variable being plotted. If "yes", the results of the integration will be printed in .plt.msg file.
- **point_ll** is the lower-left corner point of the 2D window.
- **point_ur** is the upper-right corner point of the 2D window.
- **view_xrot** rotates the plot around the x-axis.
- **view_zrot** rotates the plot around the z-axis.
- **xrange** specifies a range for the x-axis.
- **yrange** specifies a range for the y-axis.

- **zrange** specifies a range for the z-axis.
- **integration_xrange** specifies a x-range for numerical integration. (see also integration parameter above).
- **integration_yrange** specifies a y-range for numerical integration. (see also integration parameter above).
- **grid_sizes** are the grid sizes for the 3D plot.
- **mode_index** is the index of the lateral mode in a multi-lateral mode simulation. If only the fundamental mode is concerned, the default value of one should be used.
- The parameter **trap_index** has been added for SRH recombination, so that the user can specify which trap center is being plotted.

Example(s)

```
plot_3d variable=hole_conc point_ll=(0., 0.) &&
 point_ur=(2., 3.)
```

22.546 plot_3dcolor

parameter	data type	values [defaults]
data_file	char	
connect_planes	char	[yes]
variable	char	(see list)
new_gnuplot	char	[no]
xrange	realx2	
yrange	realx2	
zrange	realx2	
view_xrot	real	[0.]
view_zrot	real	[0.]
mode_index	intg	[1]
trap_index	intg	[1]
plane_range	intg	[1 9999]
exclude_materi(i=1..5)	intg	

plot_3dcolor is a post-processor statement used to plot 3-dimensional scalar variable data in a 3D color view.

Parameters

- **variable** is the variable to be plotted; the list of available variables is given in App. G.2.
- **data_file** is a filename that can be used to export the data in ASCII format.
- **connect_planes** connects the mesh planes.
- **new_gnuplot** generates a plotting format supported only by newer versions of GnuPlot.
- **xrange**, **yrange** and **zrange** define the plotting range.
- **view_xrot** and **view_zrot** rotate the plot around the x and z axes, respectively.
- **mode_index** is the index of the lateral mode in a multi-lateral mode simulation. By default, the fundamental mode (#1) is plotted which may not necessarily be the lasing mode.
- **trap_index** specifies the recombination center being plotted.
- **plane_range** restricts the plotting to a specific set of mesh planes.
- **exclude_materi(i=1..5)** can be used to exclude a specific material number from the plot.

Examples

```
plot_3dcolor variable=hole_conc
```

22.547 plot_3dmesh

parameter	data type	values [defaults]
data_file	char	[void]
connect_planes	char	[yes]
xrange	realx2	
yrange	realx2	
zrange	realx2	
view_xrot	real	[0.]
view_zrot	real	[0.]

plot_3dmesh is a post-processor statement used to plot 3-dimensional mesh.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **connect_planes** connects the planes with mesh lines.
- **xrange** specifies a range for the x-axis.
- **yrange** specifies a range for the y-axis.
- **zrange** specifies a range for the z-axis.
- **view_xrot** rotates the plot around the x-axis.
- **view_zrot** rotates the plot around the z-axis.

Example(s)

```
plot_3dmesh  connect_planes=yes
```

22.548 plot_3dvtk

parameter	data type	values [defaults]
variable	char	(see list)
vtk_file	char	[vttek]
xrange	realx2	
yrange	realx2	
zrange	realx2	
mode_index	intg	[1]
trap_index	intg	[1]
grid_sizes	intgx3	[20 20 20]

plot_3dvtk is a post-processor statement used to output 3-dimensional scalar variable data in VTK format. VTK format file can be visualized by another softwares such as MayaVi and ParaView.

- **variable** is the variable to be plotted. The variable listing is given in App. G.2.

- **vtk_file** is the file to which the 3D grid data is written in VTK format. If **vtk_file** is not supplied, **variable** will be used for the naming of output file.
- **xrange** specifies a range for the x-axis.
- **yrange** specifies a range for the y-axis.
- **zrange** specifies a range for the z-axis.
- **mode_index** is the index of the lateral mode in a multi-lateral mode simulation. If only the fundamental mode is concerned, the default value of one should be used.
- The parameter **trap_index** has been added for SRH recombination, so that the user can specify which trap center is being plotted.
- **grid_sizes** specifies the grid size in each axis of 3D space.

Example(s)

```
plot_3dvtk variable=optical_energy grid_sizes=[50 50 50]
```

22.549 plot_ac_curr

parameter	data type	values [defaults]
data_file	char	[void]
input_file	char	[void]
scale_2	char	[linear],log
variable	char	[void]
input_f2	char	[void]
n_variables	charxn	
xrange	realx2	
yrange	realx2	
factor_vertical	real	[1]
var_num	intg	[1]

---

plot_ac_curr is used to plot the AC current responses and related AC characteristics produced as the result of 1V of applied AC bias. The AC data must have been previously generated by **ac_voltage**.

Parameters

- **data_file** is the file to which the graphic data is written in ASCII format.
- **input_file** is the input file containing the AC analysis results to be plotted.
- **scale_2** is used to control the scale of the AC characteristics being plotted (the ordinate).
- **factor_vertical** is a factor scaling the vertical axis of the plot.
- **variable** is the variable being plotted. It is defined as follows:

Variable	definition
n_real_k (k=1,2,...)	Real part of electron AC current at kth electrode.
n_imag_k (k=1,2,...)	Imaginary part of electron AC current at kth electrode.
p_real_k (k=1,2,...)	Real part of hole AC current at kth electrode.
p_imag_k (k=1,2,...)	Imaginary part of hole AC current at kth electrode.
d_real_k (k=1,2,...)	Real part of displacement AC current at kth electrode.
d_imag_k (k=1,2,...)	Imaginary part of displacement AC current at kth electrode.
t_real_k (k=1,2,...)	Real part of total (electron + hole + displacement) AC current at kth electrode.
t_imag_k (k=1,2,...)	Imaginary part of total (electron + hole + displacement) AC current at kth electrode.
n_magn_k (k=1,2,...)	Magnitude of electron AC current at kth electrode.
p_magn_k (k=1,2,...)	Magnitude of hole AC current at kth electrode.
d_magn_k (k=1,2,...)	Magnitude of displacement AC current at kth electrode.
t_magn_k (k=1,2,...)	Magnitude of total (electron + hole + displacement) AC current at kth electrode.
ratio_ik/ij (k=1,2,...) (j=1,2,...)	Ratio of the total AC current magnitudes at the kth and jth electrodes.

$\left\lvert\frac{i_k}{i_j}\right\rvert_{v_1}$	
ratio_i_real_k/ij (k=1,2,...) (j=1,2,...)	Ratio of the real part of the AC current at the kth electrode over the AC current magnitude at the jth electrode.
ratio_i_imag_k/ij (k=1,2,...) (j=1,2,...)	Ratio of the imaginary part of the AC current at the kth electrode over the AC current magnitude at the jth electrode.
ratio2_ik/ij (k=1,2,...) (j=1,2,...)	Ratio of the total AC current magnitude at the kth electrode in response to the AC voltage defined in **input_file** to the total AC current magnitude on the jth electrode in response to the AC voltage defined in **input_f2**. To generate the two AC data input files, **ac_voltage** must be used for each electrode. $\frac{\lvert i_k\rvert_{v_1}}{\lvert i_j\rvert_{v_2}}$
conductance_k (k=1,2,...)	Conductance at kth electrode, obtained from the real part of the total AC current on that electrode.
capacitance_k (k=1,2,...)	Capacitance at kth electrode, obtained from the imaginary part of the total AC current on that electrode.

Units in the above will be changed depending on the 2D/3D nature of the simulation. For example, 2D simulation give currents in A/m and 3D simulations will use Amperes directly.

- **input_f2** is the 2nd AC input data file when **variable**=*ratio2_ik/ij*.
- **n_variables** may be used to list more than one variable so that more that one variable may be plotted on the same graph.
- **xrange** and **yrange** specify the axis plotting range.
- **var_num** is the number of variables being plotted and if used, must match the prefix in **n_variables**.

Examples

```
plot_ac_curr input_file=pmosB.ac variable=capacitance_3
plot_ac_curr input_file=pmosB.ac variable=conductance_3
```

The above two statements are used to plot capacitance and conductance at electrode 3 in response to the AC voltage response saved in the pmosB.ac file.

```
plot_ac_curr scale_2=log variable=ratio_i3/i4
```

The above statement plots the current ratio at electrode 3 over that at electrode 4. The AC data is automatically obtained from a preceding **ac_voltage** statement.

```
ac_voltage output_file=pmosB.ac log_freq1=6.  log_freq2=12. &&
      contact_num=4 freq_point=20
ac_voltage output_file=pmosB.ac2 log_freq1=6.  log_freq2=12. &&
      contact_num=3 freq_point=20
plot_ac_curr input_file=pmosB.ac scale_2=log variable=ratio2_i3/i3 &&
  input_f2=pmosB.ac2
```

The above statements generate two sets of AC responses on electrodes 3 and 4. The final plot shows the AC voltage ratio or voltage amplification by showing the current ratios on electrode #3.

22.550 plot_ac_laser

parameter	data type	values [defaults]
data_file	char	[void]
input_file	char	[void]
facet	char	[void] front, back
xrange	realx2	
yrange	realx2	

plot_ac_laser is used to plot the power response of a laser diode.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **input_file** is the input file containing the AC analysis results to be plotted. If this is not defined, an internal data filename will be assigned.

- **facet**, if specified, is the facet where the power is calculated. If not specified, the total power will be computed.
- **xrange** specifies a range for the abscissa.
- **yrange** specifies a range for the ordinate.

Examples

```
plot_ac_laser facet=front
```

The above statement plots frequency response of front facet power.

22.551 plot_ac_minispice

parameter	data type	values [defaults]
data_file	char	[void]
variable	char	[voltage], current, conductance, capacitance
node	char	[void]
element	char	[void]
scale_2	char	[linear], log
imag_part	char	[no],yes
xrange	realx2	[] (log10)
yrange	realx2	[]
factor_vertical	real	[1.]

plot_ac_minispice is used to plot AC results in external circuit elements from a mixed-mode (**minispice**) simulation.

Parameters

- **data_file** can be used to save the graphic plot data in ASCII format.
- **variable** is the AC variable being plotted; when appropriate, **imag_part** controls whether the real or the imaginary part of this variable is being plotted.
- **node** is the SPICE node number or node name used for an AC voltage plot.

- element is the SPICE element name used to plot AC current. Since conductance and capacitance values are extracted from AC current (see **ac_voltage**), this parameter also affects these variables.
- **scale_2** is used to control the scale of the AC characteristics being plotted (the ordinate).
- **xrange** and **yrange** are used to zoom the plot. As the abscissa is usually a frequency axis, the x-range is given using a logarithmic scale.
- **factor_vertical** is an artificial scaling factor that can be applied to the ordinate.

Examples

See **ac_voltage**.

22.552 plot_ac_modal_gain

parameter	data type	values [defaults]
data_file	char	[void]
input_file	char	[void]
xrange	realx2	
yrange	realx2	

plot_ac_modal_gain is used to plot the modal gain of a wave guided device. Here modal gain is defined local material gain averaged over optical power distribution of all lateral modes, assuming no interference effects between different modes.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **input_file** is the input file containing the AC analysis results to be plotted. If this is not defined, an internal data filename will be assigned.
- **facet**, if specified, is the facet where the power is calculated. If not specified, the total power will be computed.
- **xrange** specifies a range for the abscissa.
- **yrange** specifies a range for the ordinate.

22.553 plot_ac_parameters

parameter	data type	values [defaults]
data_file	char	[void]
input_file	char	[void]
scale_vertical	char	[linear] log
parameter_type	char	[s], y, h
smith_chart	char	[yes] no
touchstone_export	char	[void] myfile.s2p
touchstone_freq_units	char	[ghz]hz,khz,mhz,ghz
touchstone_format	char	[ma]ma,db,ri
char_impedance	real	[50] (Ohm)
zdim	real	[10] (um)

plot_ac_parameters is used to plot the AC 2-port parameters produced by the AC analysis defined in **ac_parameters**.

Parameters

- **data_file** can be used to save the graphic plot data in ASCII format. Unlike other plotting commands, multiple columns of data are printed as a result of this command. The first column corresponds to the frequency (or bias, depending on the type of AC analysis that was performed) and subsequent columns each correspond to one of the graphs that was produced as a result of this command.

  To output the results of a frequency analysis, the **touchstone_export** command should be preferred over **data_file**.

- **input_file** is the input file containing the AC analysis results to be plotted. If this is not defined, an internal data filename will be assigned.

- **scale_vertical** is used to control the scale of the AC characteristics being plotted (the ordinate).

- **parameter_type** defines the type of 2-port analysis plotted: s, h or y.

- **smith_chart** is applicable only when parameter type is s and specifies that a Smith chart should be used to plot the results. If negative, the S-parameters will be plotted against frequencies in the conventional way.

- **touchstone_export** instructs the program to export data in http://en.wikipedia.org/wiki/Touchstone_fileTouchstone .s2p format. The frequency units used in this file are specified by **touchstone_freq_units**.

  The format of the data and the representation of the complex numbers is specified by **touchstone_format**:

  - *RI* for a cartesian (real and imaginary) representation
  - *MA* for an polar (magnitude and angle) representation
  - *dB* for a decibel-angle representation commonly used in electronics textbooks.

  See also the following http://vhdl.org/ibis/connector/touchstone_spec11.pdflink for a more complete description of the Touchstone format.

- **char_impedance** is the characteristic impedance of the high frequency system.

- **zdim** is valid for 2D simulations only. It is the omitted third dimension (perpendicular to the 2D simulation mesh) which is needed to compute the absolute impedance of the device being simulated.

Examples

```
plot_ac_parameters parameter_type=y zdim=10
```

The above statement plots Y11, Y21, Y12, and Y22 parameters as functions of frequencies.

```
plot_ac_parameters parameter_type=s zdim=10 smith_chart=no
```

The above statement plots S11, S21, S12, and S22 parameters as functions of frequencies.

```
plot_ac_parameters parameter_type=s zdim=10 smith_chart=yes
```

The above statement plots S11 and S22 in a Smith chart and S21 and S12 in a polar plot.

22.554 plot_bias

parameter	data type	values [defaults]
data_file	char	[void]
variable	char	[void]
n_variables	charxn	
start	real	[0.]
end	real	[1.]
scale_x	real	[2.]
scale_y	real	[1.]
scale_power	real	[2.]
var_num	intg	[1]
curve_math	intg	[0], 1
dataset_start	intg	[1]
dataset_end	intg	[99]

The post-processor statement **plot_bias** is used to plot many important physical variables as a function of bias.

- **data_file** is the data file where the graphic data is to be stored.

- **variable** can be one of following:

delta_freq	Main long. mode frequency deviation from the reference frequency.
power_left	Main long. mode power emission from the left facet.
power_right	Main long. mode power emission from the right facet.
sms_ratio	Side main suppression ratio.
linewidth_spon	Linewidth due to spontaneous emission.
linewidth_carrier	Linewidth due to carrier fluctuation.
linewidth_cross	Linewidth due to cross correlation between
carrier and photon density.	
linewidth_sidemode	Linewidth due to side long. mode.
linewidth_total	Linewidth from all of the above.
linewidth*power_left	Linewidth power product.
effective_alpha	Effective alpha α_{eff}.
real_freq_#	Real part of the complex frequency of the long. mode number #.
imag_freq_#	Imaginary part of the complex frequency of the long. mode number #.
shd_#	Second harmonic distortion due for channel number #.
power_left_lm_i	Left facet emission power of lateral mode (lm) number i.
power_right_lm_i	Right facet emission power of lateral mode (lm) number i.
power_total_lm_i	Total (left+right) facet emission power of lateral mode (lm) number i.

- **n_variables** takes more than one of the variables tabulated above.
- **start** is the starting bias point in the plot.
- **end** is the relative ending bias point in the plot.
- **scale_x** is used to scale the horizontal axis (i.e., the current). By default, we assume that only one half of the symmetric device is simulated. Thus, a default scale of 2 is needed to recover the current for the whole structure.
- **scale_y** is used to scale the vertical axis of the bias plot for variables other than laser power.
- **scale_power** is used to scale the vertical axis of the light vs. current plot. A default of scale 2 is used to recover the current for the whole structure.

- **var_num** is the number of variables listed in **n_variables**.

- **curve_math** is used to indicate whether or not there is mathematical operation between different curves (*e.g.*, whether or not they should be added together). For a value of 0, no math operation is done between the curves. If it takes a value of 1 all the curves on the same plot is added together to produce an additional curve.

- **dataset_start** and **dataset_end** are used to set the starting and ending data sets used in the plot. If **dataset_end** exceeds the existing number of data sets, the last data set will be used.

Example(s)

```
plot_bias variable=power_left
plot_bias variable=power_right
plot_bias variable=sms_ratio
plot_bias variable=shd_1
```

22.555 plot_data

parameter	data type	values [defaults]
plot_device	char	[postscript]

plot_data determines how data is plot during the post-processing stage (.plt file). The following **plot_device** settings are available:

- *postscript* generates postscript files (.ps) through a GnuPlot script.

- *windows* displays the plots through a series of GnuPlot windows.

- *data_file* merely writes the data to text files without displaying any figures

Note that in addition to these options, interactive plotting of figures is also supported through the CrosslightView GUI.

22.556 plot_longitudinal

parameter	data type	values [defaults]
data_file	char	
variable	char	[wave_total_peakmode]
select_lateral_mode	intg	[1]

plot_longitudinal is used in the post-processing (.plt) of PICS3D results to plot data on the optical longitudinal mesh defined in the **section** statements.

Unlike **lplot_xyz** and other conventional plotting statements, this command does not gather data directly from the (electrical) mesh planes. This distinction is important since it allows **plot_longitudinal** to plot data from passive un-meshed sections.

Parameters

- **data_file** is a file name used to save the plot data.
- **variable** is the variable being plotted. It must be one of the following:
  - ase_density: amplified spontaneous emission profile
  - photon_density
  - photon_density_allmode
  - wave_total: $|L(z)|^2 + |R(z)|^2$; total envelope of the optical wave
  - wave_left
  - wave_right
  - wave_total_use_phase: $|L(z)+R(z)|^2$; total optical wave including standing wave pattern. A dense optical mesh is required to resolve these oscillations clearly.
  - wave_total_peakmode
  - wave_left_peakmode
  - wave_right_peakmode
  - alpha_l_over_2: imaginary part of the β propagation constant, normalized to cavity length; corresponds to a gain/absorption term. The factor of 2 shows this is the value applied to the optical field and not the photon flux.

  - delta_beta_l: real part of the β propagation constant, normalized to cavity length; only the deviation away from the β_0 used for the expansion is shown.
  - density_mode_average: average density in the active region, weighed by the longitudinal mode profile.
  - kappa_l_real: grating strength $\Re\kappa L$
  - kappa_l_imag: grating strength $\Im\kappa L$
  - phase_index: effective index of each mode; goes into the β calculations. This value is calculated on the first mesh plane of each segment: use multiple z-segments to increase the sampling of this value.

- **select_lateral_mode** selects the lateral mode number for the plot.

Examples

```
plot_longitudinal variable=photon_density
```

22.557 plot_mesh

parameter	data type	values [defaults]
mesh_infile	char	
plot_device	char	
xrange	real	(μm)
yrange	real	(μm)

plot_mesh is used by the mesh generator to plot the generated mesh.

Parameters

- **mesh_inf** is the input meshfile, not ordered.
- **plot_device** is the Gnuplot device used for the output (e.g. “postscript”).
- **xrange** and **yrange** allow the user to define an area of the mesh to zoom in on.

Examples

```
plot_mesh mesh_inf=pn.msh plot_device=x11 &&
 xrange=(0.15 0.3) yrange=(0.8 1.2)
```

22.558 plot_minispice

parameter	data type	values [defaults]
data_file	char	
variable	char	[voltage],current
node	char	
element	char	
set_scan_var	char	[void], virtual_time, time
hori_variable	char	
hori_node	char	
hori_element	char	
variable2	char	
node2	char	
element2	char	
hori_scale	char	[linear],log
vert_scale	char	[linear],log
scan_var_range	realx2	(sec)
hori_range	realx2	
vert_range	realx2	
circuit	intg	[1]
scan_num_range	intgx2	[0 99]

plot_minispice is new to the 2013 version and is used to plot the results of a mixed-mode simulation (see **minispice**.

Parameters

- **data_file** is the name of a text data file which can be used to export the plot data.
- **variable** indicates whether the voltage at a SPICE node or the current through an element should be plotted:
  - *voltage*: the voltage at the specified **node** value is plotted

  - *current*: the current through the specified **element** is plotted. A **node** parameter connected to this element must also be given to set the sign of the current: by convention, current flowing *into* a node is positive.

- **variable2** is a second variable plotted on the same plot; **node2** and **element2** serve the same purpose as their normal counterpart for this new variable.

- **set_scan_var** sets whether variables are shown as a function of the transient simulation *time* or the *virtual_time* used in the **scan** statement; the latter is used for DC sweeps.

- **hori_variable** is similar to **scan_var** in **plot_scan**. If this parameter is used, then instead of showing (e.g.) V1(t) the plot will show V1(t) vs. V2(t) where V2 is the variable defined as the horizontal variable.

  For this horizontal variable, **hori_node** and **hori_element** serve the same purpose as their normal counterpart in **variable**.

- **hori_scale** and **vert_scale** control the linear/log scaling of the horizontal and vertical axis, respectively.

- **scan_var_range** may be used to restrict the *time* or *virtual_time* variable to a certain range in the plot. If using a horizontal variable to plot rather than this time variable, **hori_range** and **vert_range** may be used to restrict the horizontal and vertical ranges.

- **circuit** refers to a specific external circuit file used in the simulation. These files are numbered following the order of the **minispice** statements in the .sol input file.

- **scan_num_range** may be used to restrict which data sets are used in the plot.

Examples

The following plots the voltage at node 1; the x axis will correspond to either the time in a transient simulation or the "virtual_time" used for a SPICE DC sweep.

```
plot_minispice  variable=voltage  node=1
```

For the next example, assume the following declarations in the .cir layout:

```
R1  1   2    10k
R2   1   2    1k
```

To plot the different currents flowing through the two parallel branches, the following commands should be used:

```
plot_minispice variable=current element=R1 node=1
plot_minispice variable=current element=R2 node=1
```

22.559 plot_more_dos_fermi

parameter	data type	values [defaults]
data_file	char	
energy_range	realx2	(eV)

plot_more_dos_fermi is a post-processing statement that plots additional data previously generated by a **more_dos_fermi_output** statement.

Parameters

- **data_file** is user-specified text file containing a copy of the plot data.
- **energy_range** is abscissa range of the DOS(E) plot.

22.560 plot_more_spectrum

parameter	data type	values [defaults]
data_file	char	
variable	char	[more_spectrum1], more_spectrumi(i=1..29)
user_xlabel	char	
user_ylabel	char	

plot_more_spectrum is a post-processing statement that plots additional data previously generated by a **more_spectrum_output** statement.

Parameters

- **data_file** is user-specified text file containing a copy of the plot data.
- **variable** is the variable being plotted. The names of these variables (*more_spectrumi(i=1..29)*) correspond to the order of the **more_spectrum_output** commands in the input file.
- **user_xlabel** and **user_ylabel** are user-defined labels added to the plot.

22.561 plot_more_trap

parameter	data type	values [defaults]
data_file	char	
tag	char	
trap_index	intg	[1]

plot_more_trap is a post-processing statement used to plot additional trap information generated by **more_trap_output**. This statement will plot the distribution of trap states relative the energy bands along with their occupancy.

Parameters

- **data_file** is user-specified text file containing a copy of the plot data.
- **tag** is a user-defined label that is used to link this command with the appropriate **more_trap_output** statement.
- **trap_index** identifies the species of traps being plotted.

22.562 plot_multilayer_optics

parameter	data type	values [defaults]
variable	char	[reflection], transmission, absorption
data_file	char	
wavelength_range	realx2	[-9999. 9999.] (μm)

plot_multilayer_optics is a post-processing statement used to get results from the multi-layer transfer matrix model (TMM) used for **light_power**. This value is averaged from the cut lines used in the TMM but it is not weighed by the input light profile. For full-spectrum simulations, it plots the spectrum of the coefficient. For single-wavelength pumping, a single value will be output in the simulation window.

This statement is typically used in solar cells to verify the spectral response of the device.

Parameters

- **variable** is one of the outputs from the TMM model: reflection, absorption or transmission.
- **data_file** is a file where the plot data can be saved.
- **wavelength_range** can be used to restrict plotting to a part of the spectrum.

Examples

```
plot_multilayer_optics variable=absorption
```

22.563 plot_qw_raw_data

parameter	data type	values [defaults]
complex	intg	[1]
cond_band	intg	[1]
val_band	intg	[1]

plot_qw_raw_data may be used to plot the band structure, levels and wave functions directly from the Schrödinger solver rather than in a band diagram sampled from the finite element mesh.

For example, if the self-consistent MQW model is turned off, the software will use a flat-band model to obtain the QW levels. Under these conditions, this command will show the flat bands in the QW and barrier while the usual band diagram would show the potential distorted by the applied potential.

However if the self-consistent model is turned on, then after an initial iteration using flat band conditions, the potential for the Schrödinger solver is sampled using a single cut line across the QW region. Under these conditions, **plot_qw_raw_data** will then show that sampled potential while the usual band diagram would show the potential at the cut line specified by that plotting command.

Parameters

- **data_file** is the name of a text file in which a copy of the plot data will be saved.
- **complex** is the complex MQW region number being plotted. **cond_band** is the number of the conduction band valley (Γ=1). **cond_band** is the number of the valence band valley (HH=1, LH=2, CH=3).

22.564 plot_rtgain

parameter	data type	values [defaults]
plot_device	char	[void]
rtgain_data	char	[void]
standing_wave	char	[void]
std_wave_range	realx2	

The statement **plot_rtgain** is used as a preprocessing tool to plot the round trip gain and standing wave pattern within a LD or a VCSEL. It takes the data (.rtd file) from PICS3D running a .sol file on a statement **rtgain_phase**.

- **plot_device** is the Gnuplot device used for the output (i.e. “postscript”).
- **rtgain_data** is the .rtd file as a result of running **rtgain_phase** statement in .sol file.
- **rtgain_data** is the .rtd file as a result of running
- **standing_wave** is the file (.stw file) containing standing wave in a VCSEL. This will be generated for a VCSEL by the **rtgain_phase**.
- **std_wave_range** specifies the wave range to be plotted.

Example(s)

```
plot_rtgain rtgain_data=jim.rtd standing_wave=jim.stw &&
  plot_device=postscript
```

22.565 plot_scan

parameter	data type	values [defaults]
data_file	char	[vttek]
scale_1	char	log, [linear]
scale_2	char	log, [linear]
scan_var	char	(see table below)
variable	char	(see table below)
n_variables	charxn	
include_data	char	[void]
facet	char	[void] front back
user_xlabel	char	[void]
user_ylabel	char	[void]
user_title	char	[void]
user_label1	char	[void]
user_label2	char	[void]
user_label3	char	[void]
merge_next	char	[no] yes
filter_name	char	[void]
plot_slope	char	[no] yes
set_grid	char	[no] yes
show_points	char	[no] yes
scan_label	char	[void]
parallel_circuit	char	[void]
oscillation_smooth	char	[no]
series_circuit	char	[no]
save_as_excel_csv	[char]	[]
csv_col_labeli(i=1...9)	char	[]
scale_lit	real	2.
scale_curr	real	2.
xrange	realx2	
yrange	realx2	
xy_label1	realx2	[0. 0.]
xy_label2	realx2	[0. 0.]
xy_label3	realx2	[0. 0.]
scale_horizontal	real	[1.]
scale_vertical	real	[1.]

filter_value	real	[0.]
horizontal_power	real	[]
vertical_power	real	[]
parallel_resistor	real [50.0] (ohm*m | ohm)	
series_resistor	real [50.0] (ohm*m | ohm)	
var_num	intg	[1]
mode_index	intg	[1]
trap_index	intg	[1]
scanline	intg	
scan_num	intg	

plot_scan is a post-processor statement used to plot "scan" data as a function of the applied bias on the device. This should be contrasted with "xy" (structural) data which is plotted as a function of the local mesh points.

The data sets plotted by this command are specified in the preceding **get_data** statement.

Parameters

- **data_file** is the name of a text file in which a copy of the plot data will be saved.
- **scale_1** is used to indicate linear or log scale on the abscissa.
- **scale_2** is used to indicate linear or log scale on the ordinate.
- **scan_var** is the scanned variable (on the abscissa). Refer to App. G for a full list of scan variables.
- **variable** is the scanned variable (on the ordinate). Refer to App. G for a full list of scan variables.
- **n_variables** is a group of variables to be plotted on the ordinate.
- **include_data** is the name of a text file containing data to be added to the plot. This can be used, for example, to add experimental data to a figure and compare with the simulation results.

  The expected format of the text file is that the first line defines the number of points while all the other lines are x-y pairs. The following may be used as an example:

```
3
0.1   0.5
0.2   0.8
0.5   1.2
```

- **facet** is a LASTIP-only parameter used to determine whether the laser output power corresponds to the front or back facet (or both). Note that the power is also scaled by **scale_lit** to account for the symmetry of the device.

- **user_xlabel**, **user_ylabel** and **user_title** are user-defined labels for the abscissa, ordinate and plot title, respectively. White spaces are not supported and should be replaced with an underscore (_). The upper limit on the number of characters in the string is 20 except for **user_xlabel** which allows for 68 characters.

- **user_label1**, **user_label2** and **user_label3** are user-defined labels which can be added at certain coordinates of the plot. These coordinates are defined by **xy_label1**, **xy_label2** and **xy_label3**, respectively.

  The same rules as for **user_title** apply concerning the content and length of the user-supplied string.

- **merge_next** is used to instruct the program to not to plot this curve but to transfer the data and merge the curve into the next plot statement.

- **filter_name**, if used, is the name of a control variable used to filter plotting results. For example, this can be used to plot some quantity p1 vs. p2 with the condition that only results that match the condition p3=value are included. This control value is set by **filter_value**.

- **plot_slope** will cause the statement to plot the slope instead of the original quantity.

- **set_grid** turns on the display of the plot grid.

- **show_points** shows the data points of the plot instead of just the lines.

- **parallel_circuit** defines a resistor in parallel to an electrode. This is different from a more complex external circuit which would be defined in the .sol file with the **minispice** statement.

  Instead, this parameter merely adjust the value of the current being plotted by adding or subtracting a component $\Delta I = \frac{V}{R}$. The resistor value is defined by **parallel_resistor**.

- **oscillation_smooth** tells the software to try to smooth the plot.

- **series_circuit** is similar to **parallel_circuit** circuit above but defines a resistor connected in series with the device. The resistor value is defined by **series_resistor**.

- **save_as_excel_csv**, if used, is a file name used to export the plot data in a comma-separated values (CSV) format; the .csv extension is automatically added. Column headers for this file may be defined using the **csv_col_labeli(i=1...9)** parameters.

- **scale_lit** and **scale_curr** are LASTIP-only parameters used to to scale the output power and current (respectively) to account for the symmetry of the device. In previous versions of the software, a default value of 2 was used to account for implied symmetry; this is no longer the case as of the 2012 version.

- **xrange** and **yrange** specify value ranges for the abscissa and ordinate, respectively.

- **horizontal_power** and **vertical_power** are used to scale the abscissa and ordinate, respectively, by applying a power rule of the form x^n.

- **var_num** is the number of variables to be plotted when using **n_variables**.

- **inc_curve** is the number of curves in the file specified with **include_data**.

- **mode_index** specifies the number of the lateral mode for the quantity being plotted. If only the fundamental mode is of interest, the default value should be used.

- **trap_index** identifies the trap (SRH) recombination center.

- **scale_horizontal** and **scale_vertical** are user-supplied scaling factors for the abscissa and ordinate, respectively. This can be used, for example, to scale to other units or to compute average current densities. It is recommended to provide user-specific axis labels when scaling the results in this manner.

  Note that this is different from LASTIP-specific scaling provided by **scale_lit** and **scale_curr**

- **scanline** is used to group together all data sets generated from a single **scan** statement and limit the plot data to these particular data sets.

  The data generated from the **equilibrium** statement is counted as scanline=1 and every **scan** statement afterwards increases this value by 1 so scanline=2,3,4,etc...

  Note that the plotting is still restricted to the data sets specified in the preceding **get_data** statement. The scanline mechanism merely provides a way to narrow the plot data further.

- **scan_num** is the same as **scanline** except that the count starts at zero. **scan** statements after equilibrium are thus numbered as scan_num=1,2,3,etc...
- **scan_label** works in the same way as **scanline** but allows the scan to be selected using a user-defined label instead of the automatic numbering scheme. This label must be supplied in the original **scan** statement.

Examples

```
plot_scan scan_var=voltage_1 variable=current_1
```

This would plot all the I-V data for data collected by the **get_data** command.

```
plot_scan scan_var=voltage_1 variable=current_1 scan_num=3
```

This would plot all the I-V data for data produced by the 3rd **scan**.

22.566 plot_spectrum

plot_spectrum is identical to **gain_spectrum**.

22.567 pml

parameter	data type	values [defaults]
gain_correction	char	[yes]
pure_index_loss	char	[no]
permittivity_real	real	[1.0]
permittivity_imag	real	[0.1]
pml_length	real	[6.0](um)
pml_mesh	intg	[10]

The statement **pml** is used to activate the perfectly matched layer (PML) boundary model. This is a special type of boundary condition for the lateral optical mode solver which absorbs radiative waves from a leaky waveguide structure. An effective anisotropic PML method is used in the simulator.

- **gain_correction** may be used to enable the optical gain in the rate equation to be corrected for radiation loss. When set to "no", the simulator will calculate the modal gain by a weighed averaging of the local material gain with the wave intensity.
- **pure_index_loss**. There are two ways to estimate the modal radiative loss at the PML boundary. The first is the pure index method in which all imaginary material indices are set to zero throughout the device. The imaginary part of the wave equation eigenvalue is then the modal loss due to radiation. The other method involves comparing the total waveguide loss given by the eigenvalue with a weighed averaging of the local material gain with the wave intensity. The first method is more numerically stable while the second method is more accurate.
- **permittivity_real** is the real part of permittivity in the absorbing PML material, relative to the material being matched. The recommended value is around unity.
- **permittivity_imag** is the imaginary part of permittivity in the absorbing PML material, relative to the material being matched. The recommended value is between 0.01 and 0.1.
- **pml_length** is the length (or thickness) of the PML layer used for the absorption.
- **pml_mesh** is the number of mesh lines within the PML layer.

Examples

```
pml permittivity_real=1.0 permittivity_imag=0.02 pml_mesh=50 &&
  pure_index_loss=no
```

22.568 point

parameter	data type	values [defaults]
label	char	
xy	real	(μm)

point defines a point in 2D space.

- **label** labels a point.

- **xy** are the coordinates of the point.

Example(s)

```
point label=a xy=(0.4, 0.5)
```

22.569 poisson_ratio

poisson_ratio defines Poisson's ratio for the acoustic wave propagation model in SAWAVE.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.570 polarization

In the 2013 version, this statement has been renamed to avoid confusion and to reflect the fact that TE/TM material properties are now both computed automatically. See **optical_axis** for the new command used to set the orientation of the waveguide.

For interface polarization charges (e.g. wurtzite), see **set_polarization** or **polarization_charge**.

22.571 polarization_charge

This statement is used in the material macro files to define the polarization vector magnitude ($P(z)$ in the literature) for piezoelectric materials such as GaN and ZnO. This command defines the total polarization including both the spontaneous and strain-induced components so some assumptions must be made regarding the growth conditions of the material to define the strain and provide relevant formulas. By default, the macros included with the software assume a base lattice constant matching that of GaN; for materials grown on AlN or other substrates, these macro formulas should be revised. As an alternative to this command, **spont_charge** may be used to define a model where the strain-induced component of the polarization vector is computed automatically using piezoelectric tensor elements.

We also note that a reduction of the interface charge due to screening by defects and other effects is often reported experimentally; while it is possible to include this effect in the macro formula, we do not recommend this approach. Instead,

the **polarization_charge_model** command which activates this model provides built-in parameters to control screening and the effects of plane orientation.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.572 polarization_charge_model

parameter	data type	values [defaults]
mater_label	char	
input_piezo_param	char	[yes], no
screening	real	[0.5]
vector	realx3	[0. 1. 0.]
growth_plane_normal	realx3	[0. 1. 0.]
hex_lattice_a0	real	[3.189] (Å)
hex_lattice_c0	real	[5.185] (Å)
growth_plane_miller_index	intgx4	
mater	intg	
between_mater	intgx2	

polarization_charge_model works in conjunction with either the **polarization_charge** or the **spont_charge** macro statements. When used in the .sol file, this allows the charge from piezoelectric materials to be used in the simulation but only if the matching material macros define these quantities.

This command is new to the 2011 version of the software and is intended to eventually replace **set_polarization**. Do not use both methods as this will double-count the interface charge. The code to use piezoelectric tensor elements to account for the strained-induced polarization was added in the 2013 version.

Note that **polarization_charge_model** is implemented as a 3D charge distribution at the level of Poisson solver while **set_polarization** implements a 2D charge via the **interface** statement. Integrating the 3D distribution should give the same result as the 2D interface charge since both methods implement the Bernardini[117] formulas by default. However, the 3D charge distribution is very narrow so a fine interface mesh may be required to integrate it properly; the statement **internal_xpoint** may be used to control this.

Parameters

- **screening** is a screening coefficient to represent the fraction of the theoretical interface charge observed experimentally due to compensation by defects and other effects. If set 1, the full value of the interface charge is used; at a value of 0, no interface charges are present. The exact value to use in any given application is the subject of much controversy but values near 0.5 are commonly reported in the literature.

- **vector** is used to define the orientation of the polarization vector when **polarization_charge** is positive. Typically, this means the c-plane Ga-face growth direction so it corresponds to [0 1 0] (+y) for a 2D simulation.

- **input_piezo_param** controls whether the strain-induced component of the polarization vector is computed by the software using the piezoelectric tensor elements. If *yes*, the tensor elements are used alongside the spontaneous component of the polarization vector defined in **spont_charge**. Otherwise, the total (spontaneous and strain-induced) polarization vector formula defined in the **polarization_charge** statement is used to compute the charge.

- **growth_plane_normal** indicates the direction of the growth according to the simulation axes. In a normal 2D simulation, this is +y [0 1 0] while in a 3D simulation, +z [0 0 1] would be typical. More complex situations may occur in nanowires.

  This parameter is used in conjunction with **growth_plane_miller_index** to indicate the crystal orientation ((h k i l) notation) that is being grown. Both parameters are optional and are an alternate means of defining the orientation of the **vector** parameter.

  In order to calculate the orientation of the polarization vector, the crystal lattice parameters a and c must also be defined using **hex_lattice_a0** and **hex_lattice_c0**. The default values are those of GaN.

- **mater** may be used to define different orientations of the polarization vector for different materials of the device. This may occur in the side walls of nanowires.

- **between_mater** defines the polarization charge only at the interface between these two specific materials. This may be used to quickly and easily ignore all other passivated interfaces in the device.

Examples

```
polarization_charge_model screening=0.5 vector=(0 1 0)
```

22.573 polygon

parameter	data type	values [defaults]
name	char	
4_points	realx4	
3_points	realx3	
material	intg	
boundary_i	charx2	
limits_i	realx2	(μm)

polygon is used in the mesh generator to define the geometry of the device.

- **name** is the name of the polygon.
- **4_points** are the labels of the four points used to define a polygon. The points should be arranged in counterclockwise order starting from the reference point, which is equivalent to the (0,0) point of a rectangle.
- **3_points** are the labels for the three points used to define a triangle. The points should be arranged in counterclockwise order starting from the reference point, which is equivalent to the (0,0) point of a rectangle.
- **material** is the material number of the polygon being described.
- **boundary_i** (i=1,2,...) contains the labels of two corner points. These two points define an edge where the ith boundary is to be defined.
- **limits_i** contains the starting and ending points of the ith boundary along the edge defined by **boundary_i**. The reference direction is along the counterclockwise direction.

Example(s)

```
polygon name=rect1 4_points=(a b c d) boundar_1=(a b)
  limits_1=(0. 0.7)
```

22.574 previous_layer

parameter	data type	values [defaults]
file	char	[void]
auto_sequence	char	yes, [no]

This statement is used in a .layer file when that file is used to represent a single x-y plane of a larger 3D structure. Since the layer.exe processing of the file assigns material numbers to each region, it must be made aware of the previously-declared layers (in other files) to increment these materials number correctly.

Parameters

- **file** is the name of the .layer file representing the previous z-plane in the 3D structure (starting from the bottom at z=0).
- **auto_sequence**, if enabled, will automatically take the name of the previous .layer file from a temporary file created by the previous layer.exe call.

Examples

```
previous_layer file=test1.layer
```

22.575 print_active_layer

parameter	data type	values [defaults]
mater_label	char	
mater	intg	[1]

print_active_layer is used in the gain preview mode to print the active layer material macro in the .gain.msg output file. The material affected by this command is either specified through its number (**mater**) or a label (**mater_label**) if one has previously been defined.

22.576 print_macro

parameter	data type	values [defaults]
mater_label	char	
mater	intg	[1]

print_macro is used in the gain preview mode to print the bulk material macro in the .gain.msg output file. The material affected by this command is either specified through its number (**mater**) or a label (**mater_label**) if one has previously been defined.

22.577 print_optowizard_data

parameter	data type	values [defaults]
more_data	char	[no]

print_optowizard_data is used to force the Optowizard program to generate output data files (.out, .std) used by other Crosslight simulation programs. This allows the use of plotting tools such as CrosslightView.

22.578 print_sparse_matrix

parameter	data type	values [defaults]
stop_after_print	char	[yes]
print_solution	char	[no]
line_scan	intg	[2]
bias_point	intg	[1]
iteration	intg	[1]

print_sparse_matrix is primarily used for testing and debugging of the software. It exports (to a text file) the sparse matrix (A) and RHS vector data (b) for the $AX = b$ problem being solved at a given iteration of the non-linear Newton solver.

Parameters

- **stop_after_print** terminates the solver after the export.
- **print_solution** also prints the solution vector from the sparse matrix solver during the export step.

- **line_scan** .... in the **scan** statement.

- **bias_point** is the bias step number where the data export will occur.

- **iteration** is the non-linear Newton iteration where the data export will occur.

22.579 prop_constant_model

parameter	data type	values [defaults]
precalculated_gain	char	[no]
precalculated_index	char	[no]
zseg_num	intg	[1]

By default, PICS3D uses tabulated gain and index values only during the round-trip gain (RTG) preview (c.f. Sec. 21.2 for details). This command extends the use of tabulated values to the full calculations of the RTG during the **scan** statements. This is occasionally helpful to deal with convergence problems (e.g. noisy gain curves) or to speed up the simulation when the normal gain calculations take too much time.

Parameters

- **precalculated_gain** turns on or off the use of tabulated gain values.

- **precalculated_index** turns on or off the use of tabulated index values.

- **zseg_num** is the segment number where this command will take effect.

Examples

```
prop_constant_model precalculated_gain=yes
```

22.580 put_mesh

parameter	data type	values [defaults]
polygon	char	
edge	charx2	
point_from	real	(μm)
point_to	real	(μm)
number	intg	
ratio	real	
shift_center	real	[0.] (μm)

put_mesh places mesh lines on a polygon edge. This statement is automatically generated when the **layer** and [st:column]**column** statements from the .layer files are processed to create .geo files or when defining mesh directly in the GeoEditor GUI.

Parameters

- **polygon** is the name of the polygon to be affected.
- **edge** is the edge where mesh lines are defined.
- **point_from** and **point_to** are the beginning and ending of the mesh point allocation region. Distances are measured relative to the edge, in counter-clockwise order.
- **number** is the number of mesh lines to be placed.
- **ratio** is the ratio of mesh point intervals between neighbors. A schematic diagram illustrating the effect of various ratios is given in Fig. 22.22.

  1. **ratio** = 1 Uniform mesh
  2. **ratio** > 1 Mesh interval increases with distance, starting from **point_from**.
  3. 0 < **ratio** < 1 Mesh interval decreases with distance, starting from **point_from**.
  4. −1 < **ratio** < 0 Mesh interval decreases with distance, starting from a symmetric point.
  5. **ratio** <= −1 Mesh interval increases with distance, starting from a symmetric point.

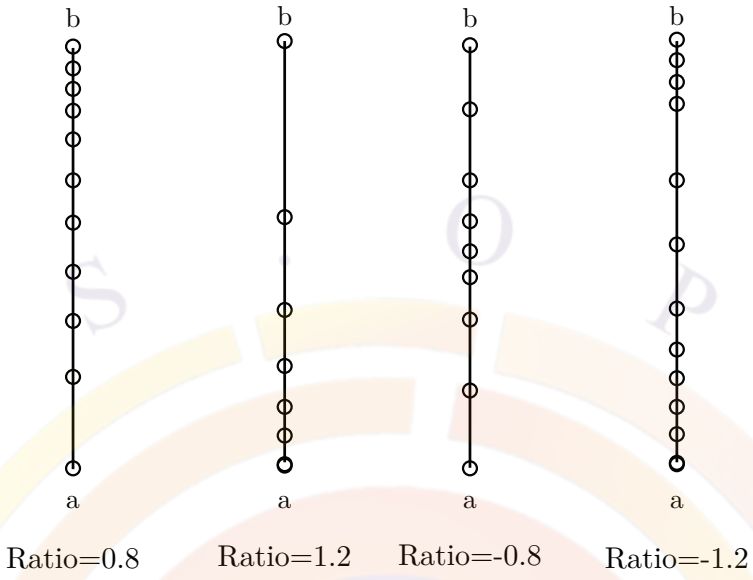


Figure 22.22: Effects of different ratio on putting mesh points on an edge (a b).

- **shift_center** may be used when mesh lines are allocated around a symmetric point: that is, when **ratio** is negative. When used, this parameter shifts the symmetric point away from the center. The input value is relative so for example, 0.2 means a shift of 20 percent of the total length.

Examples

```
put_mesh  polygon=p1 edge=(a b) point_from=0. point_to=1.4
```

22.581 q_transport

parameter	data type	values [defaults]
n_side_down	char	[yes],no
sequential_model	char	[yes],no
optic_pumped_escape	char	[no],yes
optic_pumped_fly	char	[no],yes
well_barrier_transport	char	[thermionic], trap_detrap_tau, hot_auger_thermionic, hot_auger_direct, hot_auger_indirect
set_q_trap_landing	char	[yes], no
use_drude_model	char	[no],yes
sequential_capture	char	[no],yes
collective_capture	char	[no],yes
split_qw_states	char	[no] yes
temp_dep_tau_model	char	[no] yes
scale_elec_fly_over	real	[1.]
scale_hole_fly_over	real	[1.]
scale_elec_capture	real	[1.]
scale_hole_capture	real	[1.]
set_elec_mean_fp	real	[0.01] (um)
set_hole_mean_fp	real	[0.01] (um)
drift_away	real	[0.01] (um)
bottom_drift_away	real	(um)
top_drift_away	real	(um)
q_trap_tau	real	[1.e-12] (s)
q_elec_trap_tau	real	(s)
q_hole_trap_tau	real	(s)
landing_point	real	[0.5]
split_b_coef	real	[1.e-16](m^3/s)
temp_dep_tau_t0	real	[20.] (degK)
hot_auger_cn	real	[0.], 1.e-42, 1.e-44 (m^6/s)
hot_auger_cp	real	[0.], 1.e-42, 1.e-44 (m^6/s)
hot_auger_threshold	real	[1.e24] m^{-3}
hole_drift_away	real	[0.01] (um)
hole_bottom_drift_away	real	(um)
hole_top_drift_away	real	(um)
spread_top_drift_away	real	(um)
sequential_neigbor	intg	[1]

The statement **q_transport** uses non-local transport technique to modify the behavior of the default drift-diffusion model in a quantum well or quantum dot structure.

22.581.1 Default drift-diffusion transport model

By default Crosslight uses drift-diffusion (DD) theory with a thermionic emission model as the boundary condition for heterojunctions. A quantum well is treated as having two back-to-back heterojunctions and transport is mesh-point sequential: carriers from the left barrier must first drop into the well before they jump to the right barrier.

The basic assumption here is that carriers are treated as fluid-like continuous medium where the current flow goes from one mesh point to the nearest neighbor when solving the DD partial differential equations. Put another way, imagine a 1D discretized set of equations: the DD model creates equations which couple mesh point j with its immediate neighbors $j-1$ and $j+1$. However in a real device, carriers make discrete random thermal motions before scattering with lattice and other carriers. In the 1D model, this means mesh point j may need to directly couple with a far-away point $j+n$ which introduces terms away from the diagonal in the sparse matrix solver.

If the feature size of the device is less than or comparable to the mean free path (MFP) of the carriers, the fluid-like model may become inaccurate on a microscopic scale. When dealing with nitride-based devices where the QWs are thin (2nm) and the barriers are high, this assumption may cause inaccuracies and high voltage drops not seen in experiments. This command attempts to find a solution to model this kind of structure.

22.581.2 Carrier trapping with non-local transport

A. Theoretical Basis

The non-local carrier trapping theory is based on a modified version of the drift-diffusion equation where quantum wells/dots boundaries are treated as carrier traps. We therefore aim to write a trapping rate similar to that of phonon scattering theory in the form:

$$R_{qw} = (n - n_0)/\tau \tag{22.93}$$

where τ is believed to be in the order of picoseconds.

It is important to note that this trapping describes how the 3D states of the barrier are confined to 2D confined states in the quantum well (and vice versa). It is therefore

the goal of this section to explain how the normal heterojunction boundary conditions and R_{qw} are formulated at a quantum well interface. For convenience, we only discuss electrons here; the case for holes is similar.

The carrier capture rate can be written as [136]:

$$R_{qw} = \int_{E_b}^{\infty} dE_{3D} \int_{E_{qw}}^{\infty} dE_{2D} g_{2D}(E_{2D}) g_{3D}(E_{3D}) S(E_{2D}, E_{3D}) \times f_{3D}(E_{3D})(1 - f_{2D}(E_{2D})) \tag{22.94}$$

where S is the transition probability between 3D to 2D states and the g_i are densities of states. The integrand in the above is usually a complicated function of Fermi levels and 2/3D carrier energy. To simplify the above, we make a few approximations.

The first is to ignore the dependence of $S(E_{2D}, E_{3D})$ on E_{3D} and let it equal to the conduction band energy E_c of the barrier. Noting that $n_{3D} = \int dE_{3D} g_{3D}(E_{3D}) f_{3D}(E_{3D})$, the above can be rewritten as:

$$R_{qw} = n_{3D} \int_{E_{qw}}^{\infty} dE_{2D} g_{2D}(E_{2D}) S(E_{2D}, E_c)(1 - f_{2D}(E_{2D})) \tag{22.95}$$

We also assume that most carriers are at located at the bottom of the QW so that $g_{2D} \approx \delta(E_{2D} - Eqw)$. This gives us:

$$R_{qw} = n_{3D} S(E_{qw}, E_c)(1 - f_{2D}(E_{qw})) \tag{22.96}$$

It is reasonable to expect that the transition probability S depends on the injection conditions. However, it is convenient for implementation to replace it with a single time constant τ. We also have the restriction that we expect to have $R_{qw} = 0$ when no bias is applied so that the equilibrium condition of zero current is maintained. We therefore rewrite the above as:

$$R_{qw} = \left(\frac{n_{3D} - n_{3D,0}}{\tau} \right) (1 - f_{2D}(E_{qw})) \tag{22.97}$$

where $n_{3D,0}$ is the bulk (barrier) carrier density at equilibrium.

We again stress that R_{qw} is a re-statement of the same thermionic emission boundary conditions that are used everywhere else in the drift-diffusion model. However, this formulation allows for an explicit time constant to be easily set and override this boundary condition.

B. Implementation in Drift-Diffusion solver

The main difference between this model and the default DD theory is that this model allows carriers to directly flow over the quantum well as if it did not exist: the well interface only contributes a trapping/escape term. Carriers which flow over the QW

can be considered to not follow the normal thermalization rules and this may be considered as a form of hot carrier transport; further theoretical justifications will be given below.

The potential profile of a quantum well can be of complex shape and distorted by internal and external electrical fields (see Fig. 22.23). Given the finite size and arbitrary shape of the QW in general, the question arises what happens to the carriers that do not get trapped but fly over the quantum well. Where do they land after the flight? Thus we introduce a "landing-point" to describe the non-local nature of this problem.

Crosslight supports two models to determine the landing-point for the fly-over carriers. The first is based on mean-free-path (MFP landing-point). In this approach, we assume the carriers can fly as far as the MFP allows. Since MFP scales with τ, variation of this parameter should affect the landing-point. Thus, this approach has only τ as independent variable once relationship between MFP and τ has been calibrated. This model corresponds to setting **set_q_trap_landing=no**.

Another model is based on the energy division line between bound and unbound states. In the **self_consistent** statement, the **wave_range** has been used to choose an end point in the energy band profile: the landing point can be made to coincide with this value. Within this picture, the unbound carriers fly above the energy division line $\overline{AB}$ and lands at landing point B (see Fig. 22.23). This model corresponds to setting **set_q_trap_landing=yes**.

As a reference, the simple Drude model may be used to obtain some indicative values of the MFP. It can be written as:

$$\lambda_{Drude} = v_{th}\tau_{Drude} \tag{22.98}$$

where

$$\tau_{Drude} = \mu m/q \tag{22.99}$$

$$v_{th} = \sqrt{\frac{kT}{2m\pi}} \tag{22.100}$$

where m is the carrier mass, μ the carrier mobility, q the electronic charge and v_{th} the thermal velocity. For convenience, the following formula may be used for a quick estimate:

$$\lambda_{Drude} = 1.35 \times 10^{-7}\mu\sqrt{m^*} \tag{22.101}$$

where μ is in $m^2/(V.sec)$, m^* is the relative mass and the MFP is in unit of meters. A typical value is in the range of 50-100 Å. This model corresponds to setting **use_drude_model=yes**.

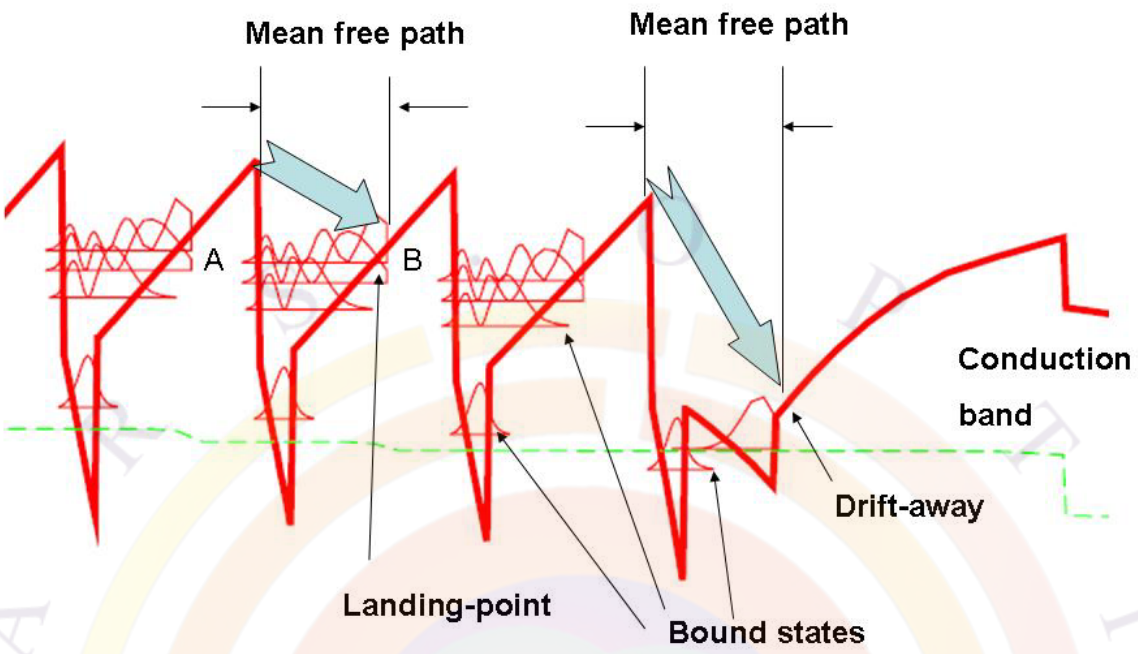


Figure 22.23: Schematic diagram explaining the quantum trapping model.

C. Mean free path-adjusted non-local transport model

This model should be regarded as a numerical approach to take into account discrete scattering events in a fluid-like DD solution. The basic physics is based on the classical DD model with thermionic emission as boundary.

The basic idea (Dr. Z.M. Simon Li, Crosslight, private communication 2009) is that when mesh size or QW size is less than the MFP of the scattering events, it is possible for the transport to occur non-locally to a remote mesh point, with a probability of:

$$exp(-\frac{distance}{MFP})$$

This correction becomes negligible if the mesh size and/or QW size are much larger than MFP. In this case, the continuous fluid-like behavior of drift-diffusion can be expected.

Another explanation on why such approach should work is based on the argument (Dr. Z.M. Simon Li, Crosslight, private communication 2004, also in [137]) that due to strong disturbance from the abrupt potential profiles of the MQW, carriers deviate from the ideal Fermi distribution, or the carriers are not being thermalized to local Fermi levels. Those non-thermal carriers are responsible for the flyover and non-local capture/escape processes. However, such interpretation is more difficult to quantify without more advanced tools such as Monte-Carlo simulation or non-equilibrium

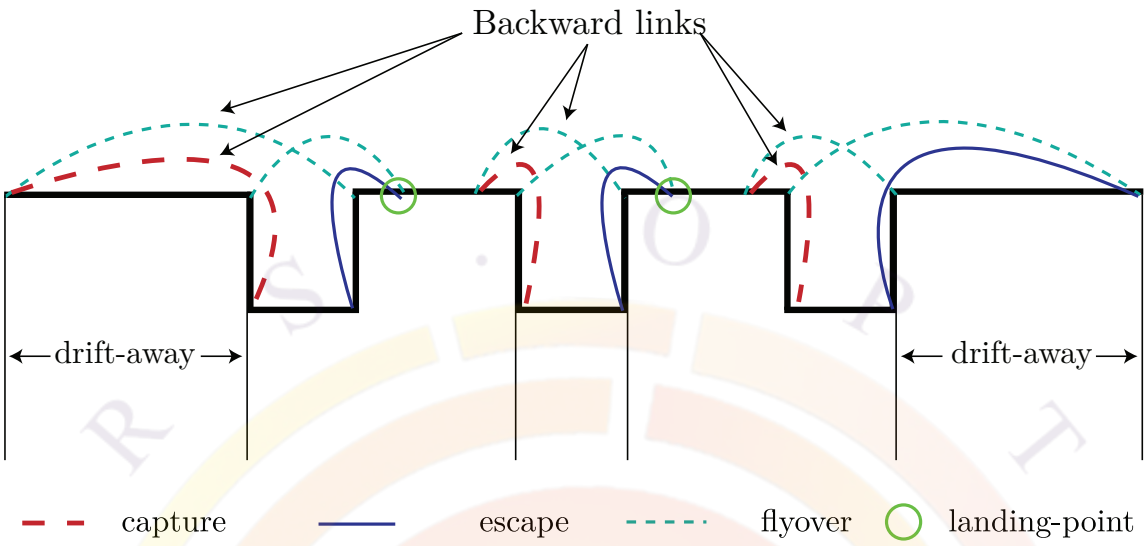


Figure 22.24: Schematic diagram explaining the sequential transport model.

Green's function (NEGF).

The implementation of this method in Crosslight is to add extra mesh point connections to those already defined by the DD model. Non-local transport is allowed to occur from barrier to barrier (over the quantum well) and between the well and barrier (for carriers with sufficient energy). In other words, we express the current density in the same way we usually use to solve the DD equations; the only difference is that the discretized current density is scaled by a factor

$$scale \times exp(-\frac{distance}{MFP})$$

where $scale = 1$ by default. The MFP value may be set by user (with a default of 100 Å) or can be computed from the simple Drude model if **use_drude_model=yes**.

Ideally, this MFP based non-local transport model is non-sensitive to the remote mesh location due to the exponential scaling. Therefore, it is sufficient to pick a mesh point in the middle of the barrier (landing-point). This model supports two modes: sequential and collective (non-sequential) non-local transport.

The sequential model allows transport to occur between the adjacent well/barriers. (see schematic in Fig. 22.24). The collective transport model allows the flow directly to from and to a emitter and collector on both sides of the MQW system (see schematic in Fig. 22.25). An interpretation of the collective injection model is that non-local transport is more favored for those carriers coming from the emitter because they do not experience the many potential barriers within the MQW.

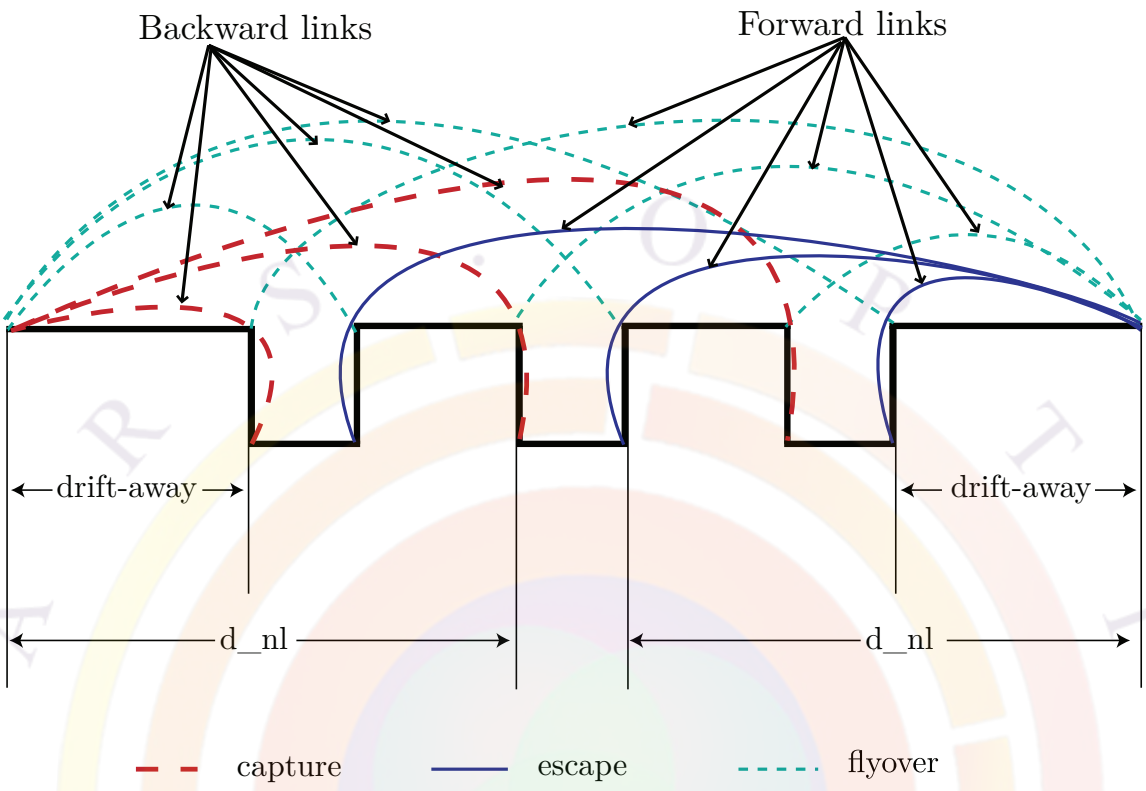


Figure 22.25: Schematic diagram explaining the non-sequential or collective transport model.

D. Relationship to hot carrier theory

Non-local transport with a mean free path can be related to hot carrier theory. By software implementation, we define **scale_elec_fly_over** as a scaling factor for the current component that directly flies over the QW to the next barrier without falling into the QW: this can be interpreted as the proportion of QW carriers which are "hot". Similarly, **scale_elec_capture** is to scale the current that is directly captured from or escaped to (two reversible processes in the software) the barrier non-locally. Again, this would be interpreted as the hot carrier fraction in the QW.

For simplicity, we imagine a situation where the transporting carriers are thermalized on the left barrier upon injection. When reaching the QW, part of the carriers fall into the QW via the trapping rate derived in the first part of this subsection and get thermalized to the lattice temperature while the rest are regarded as non-local hot carriers.

We start with the hot carrier theory developed by Azoff[2]. The current density in the hot carrier theory is written as:

$$J_n = \mu_n[n(T_e)\nabla E_c + \nabla(n(T_e)T_e) - (3/2)nT_e\nabla ln(m_n)] \tag{22.102}$$

where T_e is the electron temperature ($W = (3/2T_e)$ in Azoff's notation. To simplify our derivation, we assume the gradient of T_e can be ignored within the QW and also there is no composition grading.

It is a common practice (see also Appendix C) to introduce a parameter:

$$\gamma_f = N_c F[(E_c - E_{fn})/kT_e]/exp[-(E_c - E_{fn})/kT_e] \tag{22.103}$$

so that the electron density within the well can be written in a simple exponential form:

$$n(T_e) = N_c \gamma_f exp[-(E_c - E_{fn})/kT_e] \tag{22.104}$$

and the current flow can be expressed as:

$$J_n = \mu_n N_c \gamma_f exp[-(E_c - E_{fn})/kT_e] \nabla E_{fn} \tag{22.105}$$

We treat $(T_e - T)/T$ as a small quantity and expand the electron concentration around the lattice temperature to get a correction term due to hot carriers:

$$n(T_e) = N_c \gamma_f exp[-(E_c - E_{fn})/kT] + \beta N_c \gamma_f exp[-(E_c - E_{fn})/kT] \tag{22.106}$$

where the scaling factor

$$\beta = \frac{(E_c - E_{fn})(T_e - T)}{kT^2}. \tag{22.107}$$

Therefore the current flow can be separated into two terms:

$$J_n = J_{loc}(T) + \beta(T_e) J_{nl}(T) \tag{22.108}$$

where the first term is the usual current determined by local gradient of Fermi level and expressed as a function of local band edge and local lattice temperature.

The second term is also written mathematically as a function of local lattice temperature but the it is no longer suitable to use local quantity such as band edge E_c. The reason is that hot carriers have higher energies and do not experience scattering by local energy barriers contained in E_c, which is defined at equilibrium for carriers with lattice temperature. Thus we treat J_{nl} using a different and higher band edge. A phenomenological approach is to define a split higher band edge and assume the hot carriers fly without scattering to a distance of the mean free path (the landing point).

In Crosslight, the scaling factors for the non-local current are interpreted as the β factor above.

E. Hot Auger Models

One recent but controversial theory for the origin of droop in GaN LEDs is that the Auger recombination process creates hot carriers which can more easily leak over

blocking layers[138, 139]. For example in the CCCH process, the recombination of the electron-hole pair transfers an energy of E_g to a second electron and it takes a certain amount of time for that carrier to relax through a phonon emission process. Depending on how fast this process occurs, the carriers may or may not be at the lattice temperature by the time they reach the current blocking layers.

Such a model can only be properly handled within a framework that describes the carrier energy transport. At present, our hydrodynamic model does not include the necessary terms to represent this effect although Crosslight is also currently developing a Non-Equilibrium Green's Function (NEGF) model which will hopefully include this mechanism.

While it is not possible to represent the energy transport without these additional equations, we can attempt to represent the additional carrier leakage terms associated with hot Auger processes: these are added to the standard thermionic emission process at the well/barrier boundary. Starting with the 2014 version, three different models are offered.

Thermionic This model attempts to formulate the additional leakage current in the form of an effective emission velocity:

$$\frac{J}{q} = v_{eff} n_{\text{above barrier}} \tag{22.109}$$

Converted to current, the Auger rate has the form:

$$\frac{J}{q} = c_n n_{well}^2 p_{well} \times \text{well\_thickness} \approx c_n n_{well}^3 \times \text{well\_thickness} \tag{22.110}$$

To write the equivalence between the two terms, we write $n_{\text{above barrier}} \approx \beta n_{well}$ where β is a fraction which we treat as constant. Using this approximation, we get:

$$v_{eff} = \frac{c_n}{\beta} n_{well}^2 \times \text{well\_thickness} \tag{22.111}$$

This model is thus controlled by the parameter $\frac{c_n}{\beta}$ which we treat as a constant. However, by numerical experiment, we find that β is a bias-dependent parameter which reaches around unity when droop sets in at higher bias. For information purposes, APSYS prints outs the β value in the simulation log:

```
==>Special report on hot_auger_thermionic model
Ratio of density_emission/density_at_well=  0.103595E+01
```

This model may be regarded as a mixture of thermionic and hot-Auger electron models (hence the name). Physically, it means that only those carriers excited above the barrier are free to leak through the Auger process which explains the smooth droop this model produces. The major disadvantage of this model is the bias dependent β which may also be structure-dependent.

Auger direct escape This model assumes the full amount of confined electrons in the well are free to escape. It is then trivial to show that

$$v_{eff} = c_n n_{well}^2 \times \text{well\_thickness} \tag{22.112}$$

However, experimental observations of droop suggest that the Auger process does not contribute at low bias values. This can be understood by the fact that if the electrons are not sufficiently accelerated, the phonon emission will relax the hot carriers before they reach the blocking layers and the situation will not differ significantly from the cold-carrier model. APSYS therefore assumes the existence of some sort of threshold density for the hot Auger process and v_{eff} is scaled by:

$$f_{onset} = \frac{n_{well}}{n_{well} + n_{threshold}} \tag{22.113}$$

In numerical experiments, threshold densities around 1.e24 to 4.e24 m^{-3} would give a smooth droop similar to experimental observation. However, the initial zero efficiency from this model is somewhat odd: it may simply be unrealistic to assume confined carriers in the well are free to emit at any time.

Auger indirect escape This model is a combination of the two models above and attempts to remove the uncertainty in the β coefficient:

$$v_{eff} = c_n \frac{n_{well}^3}{n_{\text{above barrier}} + n_{threshold}} \times \text{well\_thickness} \tag{22.114}$$

This model is recommended since the rate is identical to the common Auger rate at higher injection. By adjusting $n_{threshold}$, a smooth droop can be achieved.

Notes on equilibrium balance In all of the above models, a balancing term accounting for the equilibrium contribution is included. This is similar to the normal case where $R_{aug} = c_n n(np - n_i^2)$ and the local carrier density at equilibrium balances out the net recombination rate. In the case of a non-local link, the balancing term must come from the remote point; it must also ensure that the Fermi level difference between these remote points is zero at equilibrium (i.e. no net current).

22.581.3 Split-state local carrier trapping model

A. Current Theory

In the non-local model previously described, carrier trapping/escape occurs only at well boundaries and exchanges occur between bulk 3D states in the barrier and

confined 2D states in the QW ("left-right"). Carriers flow locally in the quantum well by following a certain Fermi level and carriers which are allowed to escape/fly over the quantum well do so by deviating from normal Fermi statistics.

A different implementation of this idea is more commonly found in the literature and uses a split-state solution to describe carrier trapping in quantum wells. In this model, carrier trapping occurs continuously in the QW and not just at the barrier interfaces. Instead, an artificial band of bulk 3D states is created above the quantum well which has its own Fermi level, separate from that of the confined 2D states below. Carrier trapping/escape occurs at all mesh points inside the QW and allows exchange between these two states ("up-down"). Carriers from the barrier are no longer injected directly into the QW but instead first flow into the upper band ($3D \Leftrightarrow 3D$) and are only confined in 2D states via this exchange mechanism.

This model has the advantage of being easier to understand since all current components are strictly local, from one mesh point to its neighbor, just like the classical drift-diffusion model. The only deviation from standard DD theory is that one has to construct an artificial upper conduction band (see Fig. 22.26): in a MQW region with a distorted potential (e.g. from piezoelectric effects), the choice of upper conduction affects the transport behavior significantly. Our current implementation to simply use a straight-line connection between two barrier reference points to form the upper conduction band.

The current model in the 2014 version of the software is to assume that given a certain quasi-Fermi level E_{fn}, the total carrier density at any given point is the same in the split-state and regular cases: $n_{tot} = n_{2D} + n_{3D}$. What is needed then is a way to determine the ratio between the 2D and 3D populations.

As we discussed above, we assume that all current pumping is done through the 3D split band and that QW states are populated only through a local capture/escape mechanism: from this, we can deduce that for every mesh point, the recombination from QW states must be equal to the net capture rate from the split band. As seen in Ref. [136], the net capture rate must have the form:

$$R_{cap} = \frac{n_{3D}}{\tau}(1 - f_{2D}) - \frac{n_{3D,0}}{\tau}(1 - f_{2D,0}) \tag{22.115}$$

where f_{2D} is the 2D QW state filling factor (required due to the Pauli exclusion principle) and n_{3D} is the 3D carrier concentration in the upper state; we can see that this formulation is quite close to that of R_{qw} above. Due to our assumption on the total carrier density, we write $f_{2D} \approx \frac{n_{2D}}{n_{tot}}$.

For the recombination of the QW states, we currently assume that it is dominated by a spontaneous emission term: $R_{2D} = B(n_{2D}p_{2D} - n_{2D,0}p_{2D,0})$. Under a further assumption of charge neutrality ($n = p$), we can write an analytical formula to determine the split ratio f:

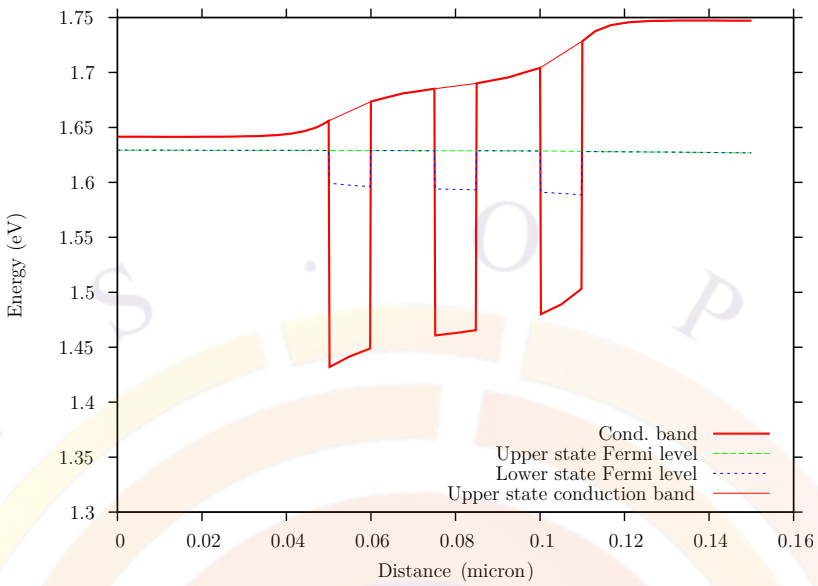


Figure 22.26: Schematic diagram explaining the split-state local trapping model.

$$B(n_{tot}^2 f^2 - n_{2D,0}^2) = \frac{n_{tot}}{\tau}(1 - f^2) - \frac{n_{3D,0}}{\tau}\left(1 - \frac{n_{2D,0}}{n_{2D,0} + n_{3D,0}}\right) \tag{22.116}$$

The 2D Fermi level is then adjusted numerically to match the target value of n_{2D}; we assume the same ratio of 2D to 3D states in both n and p carrier populations. This is done after each bias step, after the main Newton solver has converged so the split ratio is only periodically updated: use **loopback** iterations in **newton_par** to get more frequent updates.

B. Improved Theory

As seen in the previous subsection, the previous split-state model has several flaws. Aside from the arbitrary energy position of the upper split band, assuming that all current flow occurs in the upper split band ***requires*** that the Fermi level of the 2D states be flat, something which is not guaranteed by enforcing a ratio between split and confined QW densities.

Proof. When separating the 2D and 3D states, the steady-state current continuity equation for electrons is written as:

$$\nabla \cdot J_n = R_{3D} + R_{2D} \tag{22.117}$$

where R_{3D} and R_{2D} are the total net recombination from the 2D and 3D bands.

If we consider the two bands separately, then two governing equations appear:

$$\nabla \cdot J_{n,3D} = R_{3D} + R_{cap} \tag{22.118}$$
$$\nabla \cdot J_{n,2D} = R_{2D} - R_{cap} \tag{22.119}$$

so that the net capture/exchange rate is invisible to the overall discretization ($J_n = J_{n,3D} + J_{n,2D}$).

Enforcing the condition that all current flow occurs in the 3D band requires that $J_{n,2D} = 0$ so that as above, $R_{2D} = R_{cap}$. However, since $J_n = n\mu\nabla E_{fn}$, it also means that E_{fn} for the QW states must be a constant and that the entire quantum-confined region shares a uniform Fermi level. In other words, only the 3D (split-state) Fermi level may have a gradient since it is the one driving the drift-diffusion current. □

Due to time constraints for the 2014 release, this improved model is not yet available; please contact Crosslight if you require it. Interested users should also inquire as to the availability of our Non-Equilibrium Green's Function (NEGF) model which should eventually render the entire split-state approach obsolete.

As the improved split-state model is not ready yet and the older version has an obvious flaw, users should consider our existing implementation of the split-state model as extremely experimental.

22.581.4 Parameters

- **n_side_down** lets the program to recognize which is the n-doping side. The program assumes that the model operates under forward bias and thus needs to know the direction of flow based on the position of the n-side.

- **sequential_model** indicates the quantum well transport happens from one well to the next instead of collectively or coherently from all the wells to the drift-away distance. For a comparison of the collective and sequential escape/capture models, please see Figs. 22.24 and 22.25.

  For the sequential model, only the forward links are active and there is no scaling of the non-local transport terms. For the collective model, all links are active but the non-local transport terms are scaled via the MFP so interactions from distant wells is less likely to occur.

- **well_barrier_transport** replaces an older parameter called **q_trap_model** and indicates how the trapping occurs at well/barrier interfaces. The following models are available:

  - *thermionic*: default thermionic emission theory
  - *trap_detrap_tau*: R_{qw} based on **q_trap_tau**, as described above

  - *hot_auger_thermionic*
  - *hot_auger_direct*
  - *hot_auger_indirect*

- **optic_pumped_escape** is used to describe a non-equilibrium process where bound carriers are excited by optical pumping and acquire momentum to escape the quantum well. Specifically, this is used to model optical generation process in QWIP. The details are as follows: optical pumping causes bound carriers to become unbound carriers with density N_{ub}.

  The N_{ub} are estimated based on the following theory: within one period of the QWIP, optical pumping causes unbound carriers to escape quantum confinement with the thermal velocity v_{th}. It can be shown that

$$N_{ub} = \frac{pumping\_rate \times period}{v_{th}}$$

  Given unbound carrier density, it is easy to estimate the total QWIP photocurrent by drift-diffusion theory: like the pumping rate, the quantum escape term is proportional to the light intensity.

- **set_q_trap_landing** is used only when with *trap_detrap_tau*. If set to no, the MFP is used to set the landing point and drift-away distance. Otherwise, the landing point and drift-away will be set by **landing_point** and **drift_away**, respectively.

- **use_drude_model** would set the MFP according to the simple Drude model.

- **sequential_capture** and **collective_capture** are used to turn on/off the backward nonlocal mesh links as indicated in Figures 22.24 and 22.25. These parameters were previously labeled **sequential_back_links** and **collective_back_links**.

  Numerical experiment indicates that sequential transport is more directional and more localized so it would make sense to turn off the backward mesh links to keep the current flowing unidirectionally. However, the collective model is more non-local and backward links are needed for injection; it is recommended to keep the links active in that case.

- **split_qw_states** would turn on the split-state local trapping model and disable all the non-local current components. The drift-away parameter is used to find the end points of the upper split band in the barrier regions.

- **split_b_coef** is used to estimate the Fermi level split between the bulk 3D states and the QW 2D states, for a given τ constant.

- **scale_elec_fly_over**, or **scale_hole_fly_over** is a parameter to scale the direct fly-over current for electrons or holes.

- **scale_elec_capture**, or **scale_hole_capture** is a parameter to scale the non-local electron or hole current between barrier and well.

- **set_elec_mean_fp**, or **set_hole_mean_fp** is used to set the electron or hole mean free path. For the quantum-trapping model, this is used to set the landing point and drift-away distance if **set_q_trap_landing=no**. For the MFP non-local transport model, these are used for the exponential factor to control the non-local current flow intensity.

- **drift_away** is the distance from the first and last QW at which the non-local fly-over and capture/escape current would be directed towards. This setting would be overridden by the MFP if **set_q_trap_landing=no** in the quantum trapping model. This parameter would have similar meaning as **landing_point** except it also serves as emitter and collector point for the transport processes.

  This parameter applies to both carriers by default. However, a separate parameter for holes is available: **hole_drift_away**.

- **bottom_drift_away**, if defined, is the bottom drift-away distance. Please use **drift_away** instead.

  This parameter applies to both carriers by default. However, a separate parameter for holes is available: **hole_bottom_drift_away**.

- **top_drift_away**, if defined, is the top drift-away distance. Please use **drift_away** instead.

  This parameter applies to both carriers by default. However, a separate parameter for holes is available: **hole_top_drift_away**.

- **spread_top_drift_away** modifies **top_drift_away** for electrons. If used, the landing point for non-local carriers is spread over the length defined by this parameter: this length is split equally in each direction.

- **q_trap_tau** is the time constant for the trapping model. It is used in both the calculation of R_{qw} and R_{trap}. **q_elec_trap_tau** and **q_hole_trap_tau** can be used to set different times for the electrons and holes.

- **landing_point** is the landing-point of the fly-over non-local current after capture by the QW or QD. Another interpretation of this landing point is that energy level intercepting this landing-point in the band diagram would be used to define the band-edge for the unbound carriers. This parameter is expressed as a fraction of the barrier width with zero locating at the left (or bottom) side of the barrier.

  For the quantum-trapping model, this parameters affects the voltage drop of the MQW since a far landing point means current can flow without resistance for a longer distance.

- **temp_dep_tau_t0** corresponds to the term t_0 in the following formula:

$$\frac{1}{\tau} = \frac{1}{\tau_{300}} exp\left(-\frac{T-300}{t_0}\right)$$

  This is used to modify the trapping time constant as a function of the temperature; the default value is defined at 300 K.

- **hot_auger_cn** represents either c_n or $\frac{c_n}{\beta}$ depending of which hot Auger model is used. **hot_auger_cp** is the matching parameter for the hole current.

- **hot_auger_threshold** is the threshold density used in some of the hot Auger models.

- **sequential_neighbor** is the farthest neighbors to which the sequential transport model allows the carriers to make a quantum leap with a mean free path.

22.581.5 Examples and recommended usage for InGaN/GaN devices

For new users, the following default setting may suffice:

```
q_transport
```

Another common usage is the non-sequential model with the quantum trapping model:

```
q_transport sequential_model=no q_trap_model=yes
```

In this model, the user may need to fit the time constant (default of 1 ps) to affect the amount of carriers being trapped into the QW/QD. For a longer tau (i.e. less trapping):

```
q_transport sequential_model=no q_trap_model=yes &&
  q_trap_tau=1.e-11
```

The landing point may also be adjusted from the default value of 100 Å(0.01 μm):

```
q_transport sequential_model=no q_trap_model=yes &&
  set_elec_mean_fp=0.015 set_hole_mean_fp=0.015
```

22.582 q_transport_mqw_bundle

parameter	data type	values [defaults]
bundle_tag	char	mqw1
mater_label	char	
mater	intg	1

The statement **q_transport_mqw_bundle** works in conjunction with the **q_transport** statement. It is used to bundle (group) together different sets of quantum wells which share the same orientation and confinement direction: this allows the software to correctly set the non-local transport path between the different wells.

22.582.1 Parameters

- **bundle_tag** is a user-defined label shared by all quantum-confined regions that are part of the same bundle.
- **mater** is the material number of the region being added to the bundle. If a label has been defined for this material, **mater_label** may be used instead.

22.583 qc_laser_preview

parameter	data type	values [defaults]
export_property	char	void
search_wavelength	char	[no], yes
current_range	realx2	[1.e-3 1.] (A)
waveguide_loss	real	[5000] (m^{-1})
confinement	real	[0.5]
mirror_refl_2facets	realx2	[0.32 0.32]
cavity_length	real	[1360] (μm)
group_index	real	[3.8]
laser_width	real	[15] (μm)
injection_threshold	real	[0] (A/m^2)
field_current_ratio	real	[1.] ($\frac{A}{Vm} = \frac{A/m^2}{V/m}$)
wavelength_range	realx2	[5. 10.] (μm)
current_points	intg	[50]

This statement defines a basic rate equation model for a Fabry-Perot laser. It is used in the gain preview mode (.gain) in conjunction with the **qcl_3level_model** statement to model the lasing behavior of a quantum cascade laser (QCL).

The macroscopic rate equations for the laser are quite simple:

$$\begin{aligned} \frac{dN}{dt} &= \frac{I}{qV} - R_{sp} - v_g g_{net} S - R_{nr} \\ \frac{dS}{dt} &= v_g g_{net} S - \frac{S}{\tau_p} + \beta R_s p \end{aligned}$$

In more recent versions of the software, a full device model is also available based on [135]. See the tutorial examples in the LASTIP or PICS3D installation directory for details.

Parameters

- **export_property** is a file name used to save certain parameters from the QCL model such as those from **qcl_3level_model**.
- **current_range** is the range of current values at which the rate equation model is solved.
- **active_area** is the area of the active region (width × thickness).
- **waveguide_loss** is the optical loss coefficient.
- **confinement** is the waveguide confinement factor (Γ).
- **mirror_refl_2facets** is the two mirror facet reflectivities.
- **cavity_length** is the length of the laser cavity.
- **group_index** is the group index (n_g).
- **laser_width** is the width of the laser. It should be the same as the thickness used in **active_area**.
- **injection_threshold** can be used to artificially offset the threshold current to match experiments.
- **field_current_ratio** converts the applied current into the effective field value which is used in the 3 level QCL gain model.
- **current_points** is the number of data points used to sample **current_range**.

- **search_wavelength** controls how the lasing wavelength is determined. If =*no*, it is determined by the energy of the confined levels used in **qcl_3level_model**. If =*yes*, it is determined by the peak of the gain spectrum inside the **wavelength_range** limits.

Examples

```
qc_laser_preview  current_range=(10.e-3 500.e-3) &&
   waveguide_loss=13000  &&
   confinement=0.5  &&
   mirror_refl_2facets=[0.32 0.32] &&
   cavity_length=1360  &&
   group_index=3.8 &&
   active_area=18.  &&
   laser_width=14
```

22.584 qc_laser_vs_current

parameter	data type	values [defaults]
data_file	char	void
qc_variable	char	
label_horizontal	char	
label_vertical	char	
scale_horizontal	real	1.
scale_vertical	real	1.

qc_laser_vs_current plots various quantities generated by the quantum cascade laser (QCL) rate equation model (gain preview only) versus the bias current.

Parameters

- **data_file** may be used to export the plot data to a text file.
- **qc_variable** is the variable being plotted. It must be one of the following:
  - *laser_power_2facets*, the total laser power output.
  - *laser_power_left*, the laser power output from the left facet.
  - *laser_power_right*, the laser power output from the right facet.

  - *conc_injection_layer*, the carrier concentration in the injection layer.
  - *conc_active_level1*, the carrier concentration in level 1.
  - *conc_active_level2*, the carrier concentration in level 2.
  - *conc_active_level3*, the carrier concentration in level 3.
  - *peak_net_modal_gain*, the peak of the net modal gain.
  - *slope_eff_per_period*, the L-I slope efficiency divided by the number of QCL periods.

- **label_horizontal** and **label_vertical** may be used to define user labels on the plot axes.
- **scale_horizontal** and **scale_vertical** may be used to scale the axes of the plot.

Examples

```
qc_laser_vs_current qc_variable=laser_power_2facets
```

22.585 qc_net_gain_spectrum

parameter	data type	values [defaults]
data_file	char	void
wavelength_range	realx2	[5. 10.] (μm)
current_range	realx2	[100.e-3 200.e-3] (A)
current_points	intg	[5]
spectrum_points	intg	[80]

qc_net_gain_spectrum plots the net modal gain spectrum generated by the quantum cascade laser (QCL) rate equation model.

Parameters

- **data_file** is a file name used to save a copy of the plotted data
- **wavelength_range** is the wavelength range of the plot.
- **current_range** controls the bias current of the various curves in the plot.

- **current_points** is the number of points used to sample **current_range**. It controls how many curves are plotted.
- **spectrum_points** is the number of points used to sample **spectrum_range**. It controls the spectral resolution of the plot.

Examples

```
qc_net_gain_spectrum   &&
   wavelength_range=[2 12]  &&
   current_range=[10.e-3  500.e-3]  &&
   current_points=10 spectrum_points=200
```

22.586 qcl_3level_model

parameter	data type	values [defaults]
complex_start	char	void
active1_start_label	char	void
active1_end_label	char	void
active2_start_label	char	void
active2_end_label	char	void
qc3_gain_tau	real	2.5e-13 (s)
qc3_tau21	real	0.4e-12 (s)
qc3_tau31	real	12.3e-12 (s)
qc3_tau32	real	7.3e-12 (s)
qc3_injection_tau	real	8.e-12 (s)
qc3_extraction_tau	real	7.e-12 (s)
thick_qc_period	real	0.051 (μm)
average_qc_index	real	3.47
total_sheet_density	real	[1.6e15] (m^{-2})
equil_3lev_density	real	[0.4e15] (m^{-2})
qc3_gain_adjust	real	[0.] (1/m)
level3_saturation_conc	real	[3e15] (m^{-2})
thick_inj_barrier	real	[3.e-3] (um)
detuning_field_fwhm	real	[5.e7] (V/m)
resonant_field_temp_dep	real	[-1.e4] (V/m)/K
state_broadening	real	[3.e-22] (J)
shift_active_levels	intg	[0]

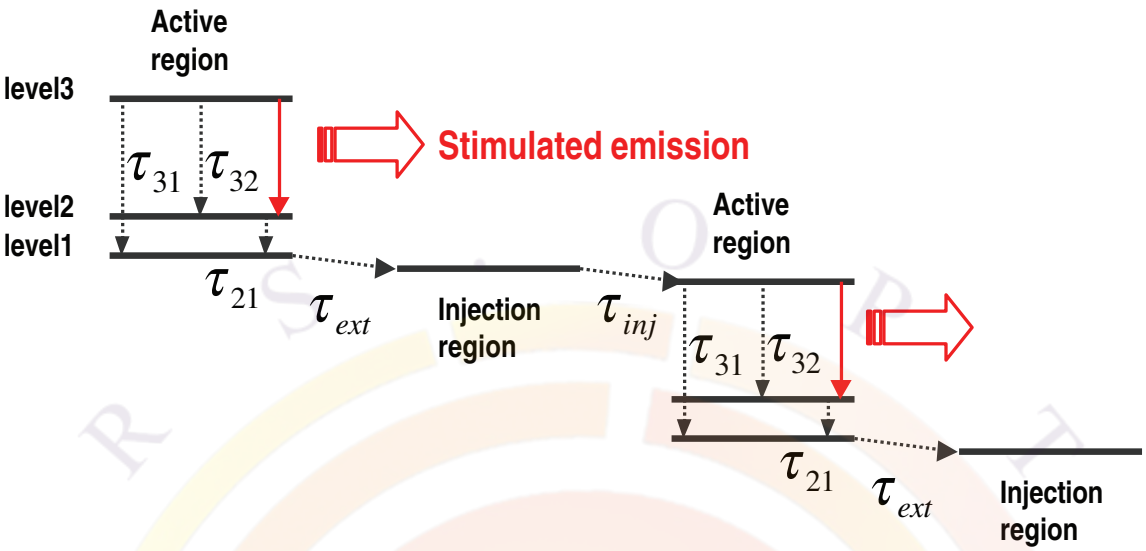


Figure 22.27: Energy levels and lifetimes used in the microscopic rate equation model of the QCL

This statement activates a 1D 3-level rate equation model used to describe the active region of a quantum cascade laser (QCL). It can be used in gain preview mode to compute the material gain and a rough estimate of the L-I curve or more recently, to provide data for a full device simulation. Our model is based on [140] and developed in more detail in [135].

For a given period of the QCL, we solve for all quantum states. Based on the shape and location of the wave functions and their respective energy levels, we identify the states that belong to the active region and those that belong to the injection region. Coupled microscopic rate equations then determine the population of three critical levels in the active region: from this, we can calculate the transition strengths and the optical gain.

Figure 22.27 shows the various levels and lifetimes involved in the quantum cascade.

Parameters

- **complex_start** is a position label that defines the start of the search region for quantum states: this position may lie outside the active region.
- **active1_start_label** and **active1_end_label** are the labels used to define the beginning and end positions of the first active region.
- **active2_start_label** and **active2_end_label** are the labels used to define the beginning and end positions of the second active region.

- **qc3_gain_tau** is the broadening constant used in the gain calculations.
- **qc3_tau21** is the time constant of the transitions between levels 2 and 1 of the active region.
- **qc3_tau31** is the time constant of the transitions between levels 3 and 1 of the active region.
- **qc3_tau32** is the time constant of the transitions between levels 3 and 2 of the active region.
- **qc3_injection_tau** is the time constant of the transition between the injection region and level 3 of the active region.
- **qc3_extraction_tau** is the time constant of the transition between level 1 of the active region and the injection region.
- **thick_qc_period** is the period of the quantum cascade.
- **average_qc_index** is the average refractive index in the active region: this is used to calculate the gain.
- **shift_active_levels** is used to shift up the selection of the three levels for the quantum cascade if level 1 of the active region is not the lowest well level.
- **total_sheet_density** is the total sheet density at which the material gain is computed.
- **equil_3lev_density** is the sheet density at equilibrium and can be estimated from the doping.
- **qc3_gain_adjust** can be used to artificially adjust the material gain.
- **level3_saturation_conc** is the level 3 sheet density at which gain saturation starts to occur.
- **thick_inj_barrier** is the thickness of the injection barrier.
- **detuning_field_fwhm** is the full width at half maximum of the Lorentzian detuning field. The peak frequency is further adjusted with the temperature based on **resonant_field_temp_dep**.
- **state_broadening** is used to broaden the energy level (Γ).

Examples

```
qcl_3level_model qc3_gain_tau=2.5e-13 &&
  active1_start_label=active1_start active1_end_label=active1_end &&
  active2_start_label=active2_start active2_end_label=active2_end &&
  complex_start=complex_start_well &&
  qc3_injection_tau=7.e-12 qc3_extraction_tau=5.e-12
```

22.587 qcl_lo_phonon_scattering

parameter	data type	values [defaults]
phonon_energy	real	[0.035] (eV)
diel_inf_minus_static	real	[-3.0]
upper_level_k_factor	real	[10.]

qcl_lo_phonon_scattering enables LO phonon scattering in the quantum cascade laser models.

This model is used in QCL devices with high k-space lasing[141]; in these devices, local minima in k-space ("electron pools") may create competitive alternate transition paths where the LO phonon scattering occurs before the optical transition.

Parameters

- **phonon_energy** is the LO phonon energy.
- **diel_inf_minus_static** is the difference between the relative dielectric constant at $\omega = \infty$ at the static value at $\omega = 0$.
- **upper_level_k_factor** denotes the position of the high k-space local minima as a multiple of the LO phonon energy: in other words, this is the average number of phonon events that occur prior to the optical transition.

22.588 qcl_period_location

parameter	data type	values [defaults]
y_start_label	char	
y_end_label	char	
yrange	realx2	(um)

qcl_period_location defines the range of one superlattice period in a QCL structure.

The range can be defined using either position labels (**y_start_label** and **y_end_label**) or absolute coordinates (**yrange**).

22.589 qcl_qw_region

parameter	data type	values [defaults]
property	char	[active], injection
mater_label	char	
mater	intg	[1]

qcl_qw_region identifies the role a particular QW layer serves in a QCL structure. The layer can be part of either the active (gain) region or part of the injection region.

The layer tagged in this fashion must be identified by either its material number or a previously defined label alias.

22.590 qcl_temperature_model

parameter	data type	values [defaults]
backfill_tau	real	[1.e-12] (s)
gain_tau_temp_coef	real	[0.001] (1/K)
lo_phonon_energy	real	[0.1] (eV)

qcl_temperature_model defines parameters which control the temperature dependence of the QCL gain model[135].

Parameters

- **backfill_tau** is the backfill time constant τ_{bf}.
- **gain_tau_temp_coef** is the coefficient (A_g) controlling the temperature dependence of the gain broadening time constant τ_g.

- **lo_phonon_energy** is the LO photon energy E_{LO}.

22.591 qdot_individual

parameter	data type	values [defaults]
tau_broaden	real	[0.6e-13] (s)

qdot_individual is used to turn on a stand-alone model in which the drift-diffusion model is solved (along with the 3D Schrödinger equation) in an isolated quantum dot region such as in the tip region of a GaN nanowire.

Note that this model is different than our older approach using **qdot_material**, in which quantum dots are declared as an embedded material inside a larger-scale wetting layer. The older model merely solves Schrödinger equation using a separate sub-project to provide extra confined states that affect the calculation of the optical gain/spontaneous emission spectra of the main device simulation. The QDOT region is thus ignored for most other purpose, including carrier transport.

On the other hand, **qdot_individual** allows for a more tightly coupled solution of the dot region but it completely ignores the behavior of the rest of the device. This model should thus be used when circumstances dictate that the QDOT region behaves differently than the rest of the device.

Parameters

- **tau_broaden** is a broadening time constant with the same meaning as in **qdot_material**.

22.592 qdot_layer_mater

parameter	data type	values [defaults]
macro_name	char	[void]
active_macro	char	[void]
var_symboli(i=1..9)	char	[void]
avar_symboli(i=1..9)	char	[void]
vari(i=1..9)	real	[-9999.]
avari(i=1..9)	real	[-9999.]
mater_lib	char	[void]
column_num	intg	[1]

qdot_layer_mater is used to indicate the presence of quantum dots (QDOTs) in a certain layer. It must be used after the **layer_mater** statement describing the wetting layer in which the quantum dots are embedded.

Please note that in general, the .layer format is used to define the device layout on a macroscopic scale (several μm): as the QDOTs are much smaller objects, they are not directly represented in the .layer file. Instead, the .layer format declares them as being an "embedded" material to be declared later. The full device simulation thus requires a preliminary step of finding the 3D quantum levels of the dots using a separate project file. Once these levels are available, they can be imported into the main device simulation.

Parameters

All parameters for this command are the same as in **layer_mater**.

Please note that if you are using the "macro" system instead of the "library" system to declare material parameters, a complex MQW macro (typically prefixed by "cx-") must be used; see Sec. 3.5 for details.

Examples

```
qdot_layer_mater macro_name=ingan active_macro=cx-InGaN &&
  var_symbol1=x avar_symbol1=xw var1=0.450 avar1=0.450 &&
  column_num=1 height=0.005
```

22.593 qdot_material

parameter	data type	values [defaults]
import_dir	char	[void]
import_qdot_data	char	[void]
import_diri, i=2...19	char	[void]
import_qdot_datai, i=2...19	char	[void]
surface_density	real	[1.e15] (m^{-2})
tau_broaden	real	[0.6e-13] (s)
wetting_area_ratio	real	[0.1]
set_cond_level1	real	
set_cond_level2	real	
set_cond_level3	real	
set_hh_level1	real	
set_hh_level2	real	
set_hh_level3	real	
set_lh_level1	real	
set_lh_level2	real	
set_lh_level3	real	
set_ch_level1	real	
set_ch_level2	real	
set_ch_level3	real	
weight	real	[100.]
weight2-19	real	[0.]
tau_broaden2-19	real	[0.6e-13]
dos_reduction_factor	real	1.
mater_active	intg	
coupled_dots	intg	[1]

qdot_material defines the properties of one type of quantum dot (QDOT) material. This QDOT material must be defined as an embedded (or mildly active) material inside a quantum well region which serves as the wetting layer. The structure of the quantum dot is defined using a separate sub-project and the properties of this dot are loaded back into the main macroscopic simulation using this command.

For more details on embedded active materials, please see statement **active_reg**.

Parameters

- **import_dir** is a directory containing a sub-project for an isolated quantum dot region; inside this sub-project, the Schrödinger solution is solved and the confined states are loaded back into the main macroscopic simulation using a data file specified by **import_qdot_data**.

  If multiple species of QDOTs are present in the same device, multiple directories may be loaded using **import_diri** and **import_qdot_datai**, with i as a placeholder number for the i^{th} species of dot.

- **surface_density** is the surface density of the quantum dots inside a given layer. If multiple species of QDOTs are present, this value defines the total dot density and the density of individual dots is derived through relative weighing parameters in **weight** and **weighti**, i=2...19.

- **tau_broaden** and **tau_broadeni**, i=2...19, are the energy broadening time constants used to broaden the spectrum. This parameter lumps together different effect including inhomogeneous broadening (due to size/composition fluctuation) and intraband scattering.

  Please note that different type of QDOTs may be associated with different time constants.

- **wetting_area_ratio** is a parameter related to the surface area of the quantum well material vs. the surface area of quantum dots in the wetting layer; this ratio controls the relative intensity of the two emission spectra. Microscopically, the quantum dot size is determined by the spatial extent of the wave function.

- **set_cond_level1-3**, if defined, are used to manually set the quantum levels of the conduction subbands.

- **set_hh_level1-3**, if defined, are used to manually set the quantum levels of the heavy hole subbands.

- **set_lh_level1-3**, if defined, are used to manually set the quantum levels of the light hole subbands.

- **set_ch_level1-3**, if defined, are used to manually set the quantum levels of the crystal field hole subbands. This is used for material of wurtzite structure (such as GaN).

- **dos_reduction_factor** may be used to artificially reduce the density of states of the quantum dot.

- **mater_active** is the material number of the active layer serving as the host material for the embedded qdots.

- **coupled_dots** is used to indicate whether a QDOT is quantum mechanically coupled to QDOTs in other wells. If so, this parameter is used to avoid double counting. For example, if two QDOTs in two different wells are coupled to each other, they should be counted only once since there will be a lifting of degeneracy in quantum levels.

Examples

```
qdot_material surface_density=5.e15 tau_broaden=0.30e-13 &&
  mater_active=3 import_dir=dot import_qdot_data=dot.qdd &&
  coupled_dots=1 wetting_area_ratio=0.2 &&
  tau_broaden2=0.18e-13 import_dir2=dot1 import_qdot_data2=dot.qdd &&
  tau_broaden3=0.10e-13 import_dir3=dot2 import_qdot_data3=dot.qdd &&
  tau_broaden4=0.10e-13 import_dir4=dot3 import_qdot_data4=dot.qdd &&
  weight=100 weight2=75 weight3=18 weight4=4
```

22.594 qwell_normal

parameter	data type	values [defaults]
dir	char	x,[y],z
force_n_side	char	[void], down, left
zdir_xy_ref	realx2	(um)

The **qwell_normal** statement is used in the .sol file to specify the direction normal to the quantum well plane (where there is quantum confinement). This statement is not needed for 2D devices grown along the y-axis.

Parameters

- **dir** is the direction normal to the quantum well plane.
- **force_n_side** may be used to help the software define the orientation of the QW normal relative to the p-n junction. This may be necessary for some non-local transport models.
- **zdir_xy_ref** is a coordinate used to evaluate the well thickness in the z-direction.

Examples

```
qwell_normal dir=y
```

22.595 qw_optics_control

parameter	data type	values [defaults]
parameterize	char	[yes]no
junction_type	char	[forward] reverse, non-optical
extrap_energy	char	[yes]no
extrap_conc	char	[yes]no
extrap_pn_ratio	char	[no]yes
extrap_temper	char	[yes]no
conc_range	realx2	(m^{-3})
wavel_range	realx2	(um)
pn_ratio_range	realx2	[0.5 1.5]
temper_range	realx2	(K)
wavel_points	intg	[50]
conc_points	intg	[10]
pn_ratio_points	intg	[3]
temper_points	intg	[8]

qw_optics_control is related to **import_gain_data**. While the latter is explicitly used to import data from a data file, the former is to let the program generate the tabulated gain data internally and then use the interpolated data table to perform the simulation.

- **parameterize** is used to enable or suppress this statement.
- **junction_type**
  - forward sets conc_range = 5.e23→5.e24.
  - reverse sets conc_range = 1.e22→5.e23.
  - non-optical will suppress optical gain/spon. em. in active layers.
- **extrap_energy** indicates if extrapolation is used when photon energy is outside of the range in imported data.
- **extrap_conc** indicates if extrapolation is used when concentration is outside of the range in imported data.

- **extrap_pn_ratio** indicates if extrapolation is used when hole/electron ratio is outside of the range in imported data.
- **extrap_temper** indicates if extrapolation is used when temperature is outside of the range in imported data.
- **wavel_range** is the wavelength range.
- **conc_range** is the electron concentration range in the well.
- **pn_ratio_range** is the range of ratio of hole over electron concentrations.
- **temper_range** is the temperature range.
- **wavel_points** is the number of points in the **wavel_range**.
- **conc_points** is the number of points in the **conc_range**.
- **pn_ratio_points** is the number of points in the **pn_ratio_range**.
- **temper_points** is the number of points in the **temper_range**. For uniform temperature simulation, only a single temperature is used in gain table evaluation. For heat transport simulation, more than one points must be used. The default for heat transport simulation is eight.

Example(s)

```
qw_optics_control parameterize=yes
```

22.596 qw_trap_assisted_tunneling

parameter	data type	values [defaults]
field	real	[1.e7] (V/m)
s_huang	real	[0.3]
phonon_energy	real	[0.09] (eV)
elec_mass	real	[0.2]
scale_phonon_term	real	[0.3]
qw_level_from_ec_bar	real	[0.5] (eV)
trap_level_from_ec_bar	real	[0.5] (eV)
scale_total_rate	real	[1.e-2]

qw_trap_assisted_tunneling enables a phonon and trap-assisted tunneling model for confined QW states; this model has been proposed as a possible mechanism for GaN LED droop.

Using this approach, the emission rate into the barrier is given by[142, Eq. 19]:

$$e(T) = \sqrt{\frac{k_B T}{2\pi m^* L_w^2}} e^{-\Delta E \Big/ k_B T} + k \cdot \left(1 - e^{-\hbar\omega \Big/ k_B T}\right) \times \sum_{n=0}^{\infty} \sum_{p} e^{-n\hbar\omega \Big/ k_B T} \Gamma(\Delta_p) J_p^2 \left[2\sqrt{S\left(n + \frac{1}{2}\right)}\right] \tag{22.120}$$

where $\Gamma(\Delta_p)$ is Korol's formula for the ionization rate of an electron trapped in a delta function potential well.

Parameters

- **field** is the electrical field used to define $\Gamma(\Delta_p)$.
- **s_huang** is the Huang-Rhys electronâĂŞphonon coupling constant S.
- **phonon_energy** is the $\hbar\omega$ term in the above equation.
- **elec_mass** is the relative electron mass m^* in the above equation.
- **scale_phonon_term** is the scaling coefficient k in the above equation.
- **qw_level_from_ec_bar** is the position of the confined state relative to the conduction band barrier.
- **trap_level_from_ec_bar** is the position of the defect state relative to the conduction band barrier.
- **scale_total_rate** is a scaling factor applied to the total recombination rate when this term is used for the non-radiative recombination rate in the Drift-Diffusion model.

22.597 qwip_model

parameter	data type	values [defaults]
period_thickness	real	[0.05] (um)
average_refr_index	real	[3.5]
spectrum_tau	real	[2.5e-13](sec)

qwip_model is used to set a number of parameters for absorption spectrum calculation of a quantum well in a QWIP (quantum well infrared photodetector).

- **period_thickness** is the period of the QWIP.
- **average_refr_index** is the average refractive index used by the absorption spectrum formula.
- **spectrum_tau** is the spectral broadening tau constant used by a Gaussian function. It is the standard deviation of the Gaussian.

Example(s)

```
qwip_model  period\_thickness=0.05 average_refr_index=3.5 &&
   spectrum_tau=2.5e-13
```

This would be a typical use of the command.

22.598 qwip_preview

parameter	data type	values [defaults]
export_spectrum	char	[qwip_specfile.txt]
data_file	char	[void]
wavelength_range	realx2	[3 9] (um)
spectrum_points	intg	[50]

qwip_preview is used to preview the QWIP model characteristics. It instructs how the program should present the QWIP absorption data. This command is used in conjunction with **band_distance** command which supplies the carrier density and with the **gain_module** command which defines the applied field.

- **export_spectrum** exports the absorption spectrum to this file.
- **data_file** if not set to void, instructs the program to print data points of all the plots of this command.
- **wavelength_range** is wavelength range over which the spectrum is calculated.
- **spectrum_points** is the data point number of the spectrum.

Example(s)

```
qwip_preview  light_power_range=(0 1.e4) &&
   wavelength_range=(3 15) spectrum_points=150
```

This command would cause the program to print the absorption spectrum data to qwip_specfile.txt after the plot.

22.599 qwire_complex_region

parameter	data type	values [defaults]
mater_label	char	
mater	intg	[1]

qwire_complex_region is analogous to **complex_region** except that it is used in the declaration of 2D quantum-confined regions (quantum wire) rather than complex MQW regions with 1D confinement.

This statement is usually generated automatically when processing the .layer file. See **start_qwire_complex**.

Parameters

- **mater** is the material number that is part of the quantum wire cross-section. If a label has previously been defined, **mater_label** may be used instead.

Examples

```
begin_qwire_complex &&
```

```
  x1=0.1  x2=0.15 &&
  y1=0.12 y2=0.3
qwire_complex_region mater=    5
qwire_complex_region mater=    6
qwire_complex_region mater=    7
end_qwire_complex
```

22.600 radiation_heavy_ion

parameter	data type	values [defaults]
let_file	char	
let_unit	char	[eV/Angstrom], eV/nm, eV/um, MeV/um, pC/um, MeV*cm^2/mg
depth_unit	char	angstrom, [um] , nm
layeri_material (i=1..5)	char	[Si],Ge,GaAs,SiO2
model	char	[import_let_profile], uniform
lateral_straggle	real	[0.1] (μm)
angle	real	[0] (degrees)
location_xz	realx2	[0.1 0] (μm)
pulse_stddev	real	(sec)
layeri_from_top (i=1..5)	real	(μm)
pulse_fwhm	real	(sec)
lateral_entry	real	(μm)
rotation	real	[0] (degrees)
uniform_ion_energy	real	[100] (MeV)
uniform_let	real	[10]

radiation_heavy_ion is used to implement an advanced heavy ion hit model in the device modeling. The path of an ion beam in a semiconductor device must be modeled using external tools such as "Stopping and Range of Ions in Matter":www.srim.org.

Note that simpler models of heavy ion impacts can also be implemented using the **generation_rate** statement.

Parameters

- **let_file** is the name of a text file containing Linear Energy Transfer (LET) data. This data should be in two columns: depth (in units set by **depth_unit**) and energy loss (in units set by **let_unit**).

- **layeri_material (i=1..5)** describes the material in layer #i with "i" as a placeholder value; the thickness of each layer is defined in **layeri_from_top (i=1..5)**.

  In addition to the few standard materials included by default, custom materials may be defined here using a name starting by "eV" and followed by the ionization energy (e.g. *eV5.91*). The term ionization energy used here refers to the energy necessary to create electron-hole pairs since there are several contributions to the ion energy loss (LET) term.

- **model** may be used to switch between importing a profile or defining a simple uniform profile for the whole device.

- **lateral_straggle** is the lateral uncertainty (standard deviation) of the ion beam's position (at half-maximum). The size of the beam on entry is defined using **lateral_entry** and also serves to describes the ion beam's path into the device.

- **location_xz** is the location of the ion hit on the x-z plane.

- **angle** is the angle of the incoming ion beam with respect to the -y direction (inside the x-y plane); **rotation** describes the rotation of this beam about the y-axis. In essence, these two values are analogous to the ϕ and θ angles of spherical coordinates but with the y-axis as a reference.

- **pulse_fwhm** is the full-width at half-maximum of the carrier generation pulse following the ion hit. It is related to the pulse standard deviation **pulse_stddev** by the following formula: fwhm $= 2.355 \times \sigma$. Only one of these two quantities should be specified.

- **uniform_ion_energy** defines the ion energy in the simple uniform model.

- **uniform_let** defines the energy transfer term in the simple uniform model; this value is defined using the same units as **let_unit**.

Examples

The following commands declare two separate ion strikes:

```
radiation_heavy_ion let_file=trim_ioniz.txt let_unit=eV/angstrom &&
  depth_unit=angstrom layer1_material=Si layer2_material=SiO2 &&
  layer3_material=Si &&
  lateral_straggle=0.2 location_xz=(0 0) &&
  layer1_from_top=0.5 layer2_from_top=0.025 layer3_from_top=10 &&
  pulse_fwhm=1.5e-12
```

```
$ 2nd ion strikes
radiation_heavy_ion let_file=trim_ioniz.txt let_unit=eV/angstrom &&
  depth_unit=angstrom layer1_material=Si layer2_material=SiO2 &&
  layer3_material=Si &&
  lateral_straggle=0.2 location_xz=(0.2 0) &&
  layer1_from_top=0.5 layer2_from_top=0.025 layer3_from_top=10 &&
  pulse_fwhm=1.5e-12
```

During the simulation, the *light* variable in the **scan** command controls the timing of the ion impacts:

```
$ The normalized light and light2 control the radiation generation
$ rate from 1st and 2nd ion strikes, respectively.
$ You can define up to 9 different ions, hit at different spots
$ and strike at different times.
scan var=time value_to=6.e-12 &&
  var2=light  function_label2=gaus_pulse1 &&
  var3=light2  function_label3=gaus_pulse2 &&
  init_step=0.1e-12 max_step=0.2e-12

scan_function label=gaus_pulse1 type=gaussian &&
  gsn_t1=0.  gsn_dt=1.5e-12 gsn_s1=0.  gsn_s2=1.

scan_function label=gaus_pulse2 type=gaussian &&
  gsn_t1=2.e-12  gsn_dt=1.5e-12 gsn_s1=0.  gsn_s2=1.

scan var=time value_to=1000.e-12 &&
  init_step=0.1e-12 max_step=50.e-12 min_step=0.001e-12 &&
  var2=light  value2_to=0.0  &&
  var3=light2  value3_to=0.
```

Note that in the above, two different scan functions (and labels) are used to change the timing between the initial ion hit and the secondary “echo”.

22.601 radiative_boundary

parameter	data type	values [defaults]
x_label	char	
y_label	char	
within_x1_label	char	
within_x2_label	char	
within_y1_label	char	
within_y2_label	char	
x	real	[-9999.] (μm)
y	real	[-9999.] (μm)
within_x	realx2	[-1.e5 1.e5] (μm)
within_y	realx2	[-1.e5 1.e5] (μm)
emissivity	real	[0.9]
environ_temp	real	[300.]

This statement is similar to **thermal_interf** and is used to describe a radiative boundary. The heat flow through the boundary is given by the StefanŰBoltzmann law.

Parameters

Most of the position dependent parameters are the same as those of Sec. 22.696.

- **emissivity** is the fraction of black body radiation being emitted.
- **environ_temp** is the environment temperature.

Examples

```
radiative_boundary y=0.8 emissivity=0.9
```

22.602 radiative_recomb

radiative_recomb is the radiative (or spontaneous) recombination coefficient in units of m^3/sec. It is usually defined as the coefficient B such that the radiative recombination is given by $B(np - n_i)$ where n_i is the intrinsic carrier density. Note that this coefficient is only used for the non-active region of the laser. For active

regions, the spontaneous emission rate is computed from first principles, analogous to the optical gain.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.603 raw_output

parameter	data type	values [defaults]
variable	char	
output_file	char	void
mode_index	intg	[1]
trap_index	intg	[1]

The statement **raw_output** is a post processing command used to print all available 2D data in the following form:

```
total node number
listing of x,y,z,variable
```

- The output is written to the output file defined by **output_file**
- **mode_index** is the mode index or mode number of the wave modes if the quantity being printed is wave function.
- **trap_index** is the trap index or trap number of the deep level traps if the quantity being printed is related to traps (eg, trap concentration).
- **variable** is one of the variables listed in appendix G.2.

22.604 rcled_dbr

parameter	data type	values [defaults]
x_start_label	char	[void]
x_end_label	char	[void]
y_start_label	char	[void]
y_on_top	char	[no]
y_at_bottom	char	[no]
layer1	real	[0.1]
complex_index1	realx2	[(3.5 1.e-6)]
layer2	real	[0.1]
complex_index2	realx2	[(3.5 1.e-6)]
period_number	intg	

The statement **rcled_dbr** is used to define a DBR structure within an RCLED. The basic set up of this model is that optical layers are parallel to the x-axis and the device emits mainly in +y or -y directions.

- **x_start_label**, **x_end_label** are the position labels for the left and right bounds of the DBR. The labels were previously defined by **x_position** statements. If one or more of the labels are not defined, the program uses the boundary of the device.

- **y_start_label** is used to define the vertical position of the DBR. The bottom of the DBR stack starts from this position. It is a position label previously defined by a **y_position** statement. Please note that y_start_label , **y_on_top** and **y_at_bottom** are conflicting parameters and should be set with care so that only one of the three is used.

- **y_on_top** is used to define the vertical position of the DBR if the DBR is on top of the RCLED. Please note that y_start_label , **y_on_top** and **y_at_bottom** are conflicting parameters and should be set with care so that only one of the three is used.

- **y_at_bottom** is used to define the vertical position of the DBR if the DBR is at the bottom of the RCLED. Please note that y_start_label , **y_on_top** and **y_at_bottom** are conflicting parameters and should be set with care so that only one of the three is used.

- **layer1** and **complex_index1** are the layer thickness and complex refractive index of first layer of the DBR pair.

- **layer2** and **complex_index2** are the layer thickness and complex refractive index of second layer of the DBR pair.
- **period_number** is the number of periods of DBR pairs.

Example(s):

```
$ 5 periods  TiO2/SiO2 DBR at 0.525
rcled_dbr x_start_label=x2 x_end_label=x3 &&
  period_number=5 y_on_top=yes &&
  layer1=0.0597  complex_index1=(2.2 0.0) &&
  layer2=0.0899  complex_index2=(1.46 0.0)
```

The above example puts a DBR stack on top of the device. The DBR is defined between position of x2 and x3.

22.605 rcled_model

Parameters

parameter	data type	values [defaults]
spon_all_points	char	no
optic_range_x1_label	char	void
optic_range_x2_label	char	void
optic_range_y1_label	char	void
optic_range_y2_label	char	void
top_emission	char	yes
update_spec_source	char	no
photon_recycle	char	yes
wavelength	real	
top_exit_index	real	1.
bottom_exit_index	real	1.
total_active_layer	real	0.1
group_index	real	3.8
ref_density	real	1.e23
ref_pn_ratio	real	1.
max_emit_angle	real	60
stack_view_point	real	[9999.]
num_optic_y_grid	intg	1000
num_spec_source	intg	120
num_emit_angle	intg	30

The statement **rcled_model** is used to control parameters related to the resonant cavity light emitting diode (RCLED) model. The basic set up of this model is that optical layers are parallel to the x-axis and the device emits light that is roughly parallel to the y-axis.

- **spon_all_points** indicates whether all mesh points are used to evaluate the spontaneous emission spectrum. Computation time is longer if all points are used.
- **optic_range_x1_label**, **optic_range_x2_label**, **optic_range_y1_label**, and **optic_range_y2_label** are four labels referring to four coordinate values defined by two previous **x_position** and two previous **y_position**. These are used to define a rectangle region within which multiple layer optical wave propagation calculation is performed. If one or more of these labels are not defined, the program uses the boundary of the device for optical wave modeling.
- **top_emission** indicates the main direction of power emission (top or bottom).
- **update_spec_source** indicates that the resonant cavity source shape function should be updated at all bias steps. The RCLED model relies on an angular and wavelength-dependent resonant shape function (representing the Q-factor of the cavity); this function is scaled with the source dipole spectrum (spontaneous bulk emission spectrum) and output as the total emission spectrum. Since the shape function mostly depends on the reflectivity of the cavity and the DBR layers, it is usually unnecessary to update the shape function at every bias.
- **photon_recycle** is used to enable the generation term due to the internal photon density. In thermal simulations, this setting allows heat to be generated by optical absorption. However, please note that APSYS does not posses a photon rate equation: it is expected that LEDs operate below transparency.
- **wavelength** is the main emission wavelength in microns.
- **top_exit_index** is the refractive index of the top exit medium.
- **bottom_exit_index** is the refractive index of the bottom exit medium.
- **total_active_layer** is an estimate of the total active layer thickness for the purpose of normalizing the optical power. This will affect the plotted power transmission distribution but will not affect the other results of the simulation.
- **group_index** is the group index used to calculate the group velocity of light within the RCLED cavity.

- **ref_density** is a reference carrier density, or typical carrier density of the active layer which is used to estimate the spontaneous dipole spectrum at equilibrium. Using this information, the resonant cavity properties are printed to a .res file (for resonance spectrum) and a .stw file (for standing wave). These files may be examined before running the full device simulation to help calibrate the DBR mirror settings.

- **ref_pn_ratio** is the ratio of hole density over electron density. This ratio is used together with **ref_density** above.

- **max_emit_angle** sets the maximum angle of emission to be computed in angular power distribution.

- **stack_view_point** is a reference point used to compute the reflectivity of the mirror stacks, as seen from that particular view point; these reflectivities can be plotted using the **stack_refl** parameter in **rcled_spectrum_angle**.

- **num_optic_y_grid** is the number of optical grid points in the y direction used to compute the standing wave pattern.

- **num_spec_source** is the number of RCLED emission spectrum points.

- **num_emit_angle** is the number of data points used in angular power distribution.

Examples

```
rcled_model wavelength=0.93 total_active_layer=0.024 &&
  optic_range_y1_label=dbr1_start  top_emission=no
```

The above example defines an RCLED cavity to be between a y-position labeled as dbr1_start and the top of the device. The parts below dbr1_start will not be regarded as part of the optical cavity. Emission direction is towards the bottom.

22.606 rcled_optic_layer

parameter	data type	values [defaults]
x_start_label	char	[void]
x_end_label	char	[void]
y_on_top	char	[no]
layer1	real	[0.1] (um)
complex_index1	realx2	[(3.5 1.e-6)]
layer2	real	
complex_index2	realx2	[(3.5 1.e-6)]
layer3	real	
complex_index3	realx2	[(3.5 1.e-6)]
layer4	real	
complex_index4	realx2	[(3.5 1.e-6)]
layer5	real	
complex_index5	realx2	[(3.5 1.e-6)]

The statement **rcled_optic_layer** is used to define one or more optical layer within an RCLED. The basic set up of this model is that optical layers are parallel to the x-axis and the device emits mainly in +y or -y directions.

- **x_start_label**, **x_end_label** are the position labels for the left and right bounds of the layers. The labels were previously defined by **x_position** statements. If one or more of the labels are not defined, the program uses the boundary of the device.

- **y_on_top** is used to define the vertical position of the layer.

  layerk (k=1..5) and **complex_indexk (k=1..5)** are the layer thickness and complex refractive index of the kth layer.

Example(s):

```
$ SiO2 coating
rcled_optic_layer x_start_label=x1 x_end_label=x2 &&
      y_on_top=yes layer1=0.2  complex_index1=(1.46 0.)
```

22.607 rcled_plot_y

parameter	data type	values [defaults]
data_file	char	[void]
variable	char	[void],relative_power, real_index, imag_index, spon_rate
add_variable	char	[void],relative_power, real_index, imag_index, spon_rate
distance_range	realx2	[-1.e49 1.e49] (um)
variable_range	realx2	[-1.e49 1.e49]
at_mesh_near_x	real	[0.5](um)
scale_variable	real	[1.]
scale_add_variable	real	[1.]

plot_plot_y is a post-processor statement used to plot quantities related to RCLED as a function of the y-coordinate.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **variable** is the variable to be plotted. It takes one of "relative_power" (power distribution relative to the power source), "real_index" (real refractive index), "imag_index" (imaginary refractive index), or "spon_rate" (spontaneous emission rate).

  add_variable adds a second variable to be plotted on the same figure. It takes one of the parameters listed under **variable**.
- **distance_range** is the distance range in y-direction.
- **variable_range** specifies a range for the function variable to be plotted.

  at_mesh_near_x specifies the position where a 1D slice is cut for the 1D plot.

  scale_variable is used to scale the function variable to be plotted.

  scale_add_variable is used to scale the 2nd function variable added by **add_variable** on the same plot.

Example(s)

```
rcled_plot_y  variable=relative_power add_variable=real_index &&
    at_mesh_near_x=0.5 scale_variable=0.3 scale_add_variable=1
```

The above statement plots the relative optical power along the y-direction. Added to the same plot is the real part of the refractive index.

22.608 rcled_power_angle

This command is similar to **rcled_spectrum_angle** except the optical power has been integrated over the whole spectrum for a particular angle.

22.609 rcled_refl_coating

parameter	data type	values [defaults]
x_start_label	char	void
x_end_label	char	void
y_on_top	char	yes
power_refl	real	0.32
refl_phase_in_pi	real	0.
power_loss	real	0.

The statement **rcled_refl_coating** is used to define a reflection optical coating for an RCLED. The basic set up of this model is that optical layers are parallel to the x-axis and the device emits mainly in +y or -y directions.

- **x_start_label**, **x_end_label** are the position labels for the left and right bounds of the layer. The labels were previously defined by **x_position** statements. If one or more of the labels are not defined, the program uses the boundary of the device.
- **y_on_top** is used to define the vertical position of the coating.
- **power_refl** power reflectivity of the optical coating.
- **refl_phase_in_pi** is the phase of the reflectivity in π.
- **power_loss** is the power loss percentage related to the optical coating. It may also be used to define the power loss beyond the optical coating so that the total output power is affected by it.

Example(s):

```
$ For HR from metal, phase should be close to 180 degree to make zero
$ metal boundary:
rcled_refl_coating x_start_label=x1 x_end_label=x4 &&
  y_on_top=no power_refl=0.98 refl_phase_in_pi=1.3 power_loss=0
```

The above statement defines a metal coating with a phase close to 180 degrees so that the wave intensity is nearly zero at the metal.

22.610 rcled_spectrum

parameter	data type	values [defaults]
data_file	char	[void]
top_emission	char	[yes],no
stack_refl	char	[no]
wavelength_range	realx2	
angle	real	[0.] (degree)
broaden	real	(um)

rcled_spectrum is a post-processor statement used to plot the RCLED spectrum at a certain emission angle.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **top_emission** indicates whether emission is towards the top or bottom.
- **stack_refl** would enable the plotting of reflectivity of multiple layer stack viewed from a reference point defined in **stack_view_point** in command **rcled_model**.
- **wavelength_range** is the wavelength range in microns.
- **angle** is the emission angle in degrees.
- **broaden** is the spectrum broadenning width in microns. It is the standard deviation of the gaussian broadening function.

Example(s)

```
rcled_spectrum top_emission=yes broaden=0.002
```

22.611 rcled_spectrum_angle

parameter	data type	values [defaults]
data_file	char	void
top_emission	char	yes
stack_refl	char	[no]
wavelength_range	realx2	(-9999 9999) (μm)
theta_range	realx2	(0 90)
broaden	real	-9999.
spectrum_range	realx2	(-9999 -9999) (W/eV)
view_xrot	real	0.
view_zrot	real	0.

rcled_spectrum_angle is a post-processor statement used to plot the emitting power as a function of wavelength and emission angle for an RCLED.

Parameters

- **data_file** is used to output the plot data to a text file.
- **top_emission** indicates the main direction of power emission (top or bottom).
- **stack_refl** enables the plotting of reflectivity of a multi-layer stack, as viewed from the reference point defined in the **stack_view_point** parameter of rcled_model.
- **wavelength_range** is the wavelength plotting range (x) in microns.
- **theta_range** is the emission angle plotting range (y) in degrees.
- **broaden** is the spectrum broadening width in microns. It is the standard deviation of the gaussian broadening function.
- **spectrum_range** is the power intensity plotting range (z).
- **view_xrot** rotates the plot around the x-axis (or the wavelength axis).
- **view_zrot** rotates the plot around the z-axis (or the power axis).

Examples

```
rcled_spectrum_angle top_emission=yes broaden=0.002
```

22.612 rcled_surface_plot_xy

parameter	data type	values [defaults]
data_file	char	[void]
variable	char	
xrange	realx2	
yrange	realx2	

plot_surface_plot_xy is a post-processor statement used to plot surface plot of a function of x and y for quantities related to RCLED.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **variable** is the variable to be plotted. It takes one of “relative_power" (power distribution relative to the power source), “real_index” (real refractive index), “imag_index” (imaginary refractive index), or “spon_rate” (spontaneous emission rate).
- **xrange** specifies a range for the x-axis.
- **yrange** specifies a range for the y-axis.

Example(s)

```
rcled_surface_plot_xy variable=relative_power
```

22.613 re_emission

parameter	data type	values [defaults]
shift	char	on
mater	intg	
exci_effi	real	0.7
exci_spec_file	char	void
re_emi_wavelength	real	0.6
re_emi_spec_file	char	void
loop_spec_num	intg	20
absorb_pow_dens_file_prefix	char	my_rec
did_spec_loop	char	yes

The **re_emission** statement is used in the optowizard solution file to activate the re-emission function in the ray tracing model. This allows materials such as a phosphor to absorb LED-emitted light and re-emit at a secondary wavelength.

In order to use this statement, the power absorption profile must have been recorded in a previous ray tracing simulation using **rec_absorb_pow_dens**. Using both statements in the same ray tracing simulation is not allowed in order to avoid conflicts. Therefore, two separate runs of the ray tracing simulation are required to obtain the re-emission profile.

Parameters

- **shift** is used to turn on or off this statement.
- **mater** is used to specify the material number of the phosphor. This must match the material number in APSYS.
- **exci_effi** is to specify a fixed value of the excitation efficiency of the phosphor.
- **exci_spec_file** is the name of the data file containing the excitation spectrum of the phosphor. The data must be in text format where the first column is the wavelength in micrometers and the second column is the excitation efficiency at this wavelength. Using this parameter will override the fixed value of **exci_effi**, if defined.
- **re_emi_wavelength** is used to specifiy a single fixed wavelength for the re-emitted light.
- **re_emi_spec_file** is the name of the data file describing re-emission spectrum. The data must be in text format where the first column is the wavelength in micrometers and the second column is the relative power density at this wavelength. Using this parameter overrides the fixed wavelength of the re-emitted light.
- **loop_spec_num** is used to discretize the re-emission spectrum defined in **re_emi_spec_file**. Since the ray tracing program can only work with fixed wavelengths, it will loop over the spectrum a few times in order to process the entire re-emission spectrum.
- **absorb_pow_dens_file_prefix** is used to specify the files where the recorded power density profile from the previous ray tracing run is stored. This file prefix should match the one in **rec_absorb_pow_dens**

- textbfdid_spec_loop tells the ray tracing program that a spectrum loop was used to record the power density profile in the previous ray tracing run. The software will use this information, in conjunction with the excitation effiency spectrum, to determine the amount of re-emitted light.

Examples

```
re_emission shift=on mater=2 exci_effi=0.7 re_emi_wavelength=0.65 &&
            absorb_pow_dens_file_prefix=red_phos
```

22.614 real_func

The command **real_func** is identical to **loop_real**.

22.615 real_index

parameter	data type	values [defaults]
material_par		

The material statement **real_index** is the real refractive index at the appropriate optical frequency. Please do not confuse this with the dielectric constant of Poisson's equation at DC or low frequencies.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.616 real_index_spec

parameter	data type	values [defaults]
material_par		

This statement is the wavelength dependent version of the **real_index** statement. As the free-style macro becomes available in recent versions, it has been outdated since the wavelength dependence can easily be incorporated into the **real_index** statement function by adding a variable (xlam).

22.617 rec_absorb_pow_dens

parameter	data type	values [defaults]
shift	char	[on]
mater	intg	
file_prefix	char	[my_rec]
gauss_sigma	real	[3.0]
gauss_cutoff_dist	real	[5.0]
mesh_interval_xyz	realx3	[1.0 1.0 1.0] (μm)
min_mesh_num_xyz	intgx3	[3 3 3]
max_mesh_num_xyz	intgx3	[50 50 50]

The **rec_absorb_pow_dens** statement is used in the Optowizard solution file to record the power density of the light for a specific material. This is often a phosphor and this information can later be used in the **re_emission** command to compute the secondary/re-emitted light.

This statement may also be used to turn on photon recycling models in solar cells and LEDs. However, users should note that RCLED models have a built-in photon recycling which is independent from the ray tracing (i.e. Green's function analysis).

Important note: This statement can only be used once per ray tracing simulation and cannot be combined with **re_emission**.

Parameters

- **shift** is used to turn on or off this statement.
- **mater** is used to specify the material number of the phosphor. This must match the material number in APSYS.
- **file_prefix** specifies a file prefix used to save the absorbed power density profile.
- **gauss_sigma** and **gauss_cutoff_dist** describe the lateral extent of the Gaussian beam used to record the power. Mathematically, the rays used in the ray tracing method are infinitely thin so when we record the power we assume that in reality, these rays are closer to Gaussian beams. **gauss_sigma** is the standard deviation of the beam's lateral profile while **gauss_cutoff_dist** is the cut-off distance.
- **mesh_interval_xyz** is the uniform mesh spacing used to record the power. However, these values are also constrained by **min_mesh_num_xyz** and

max_mesh_num_xyz which control the minimum and maximum number points that can be used.

Examples

```
rec_absorb_pow_dens shift=on mater=2 file_prefix=red_phos
```

22.618 regrid

parameter	data type	values [defaults]
variable	char	(see list)
method_line_base	char	[yes]
var_diff	real	[3.0]
xrange1	real	-1.E10 (μm)
xrange2	real	1.E10
yrange1	real	-1.E10
yrange2	real	1.E10
min_density	real	10. (m^{-3})
zseg_num	intg	[1]
max_double_times	intg	[5]
set_edge_split	intg	[0]
max_edge_split	intg	[3]
all_zsegm	char	[no]

regrid, previously known as **refine_mesh**, is a command to adaptively refine an existing mesh based on a specified variable distribution. It should be used in the main solution module (.sol files).

The new mesh is saved to the same filename as the original. A backup of the original mesh file will also be saved.

Parameters

- **variable** is the position-dependent variable according to which the mesh is refined. The following variables are accepted:
  - *donor_conc,acceptor_conc,2_dopants*: the donor and acceptor concentrations (or both).

  - *elec_conc*,*hole_conc*,*2_carriers*: the electron and hole concentrations (or both).
  - *potential*: the internal electrical potential V.
  - *x_distance*,*y_distance*,*x_plus_y_distance* and *x_minus_y_distance* rather than real solver values as above, these variables use the distance from the origin (in um) to determine the mesh refinement.

- **var_diff** is the criterion for mesh refinement. When the variable difference between two adjacent nodes is greater than this criterion, additional nodes are added until the variable difference drops down below the specified criterion. For concentrations, the unit is decades. For potential, the unit is Volt.

- **method_line_base** indicate whether mesh line extension method is used to do mesh-refinement. For CSUPREM imported mesh, only triangle based method is available.

- **xrange1** and **xrange2** are the lower and upper limits, in the x-direction, of the region within which the statement is effective.

- **yrange1** and **yrange2** are the lower and upper limits, in the y-direction, of the region within which the statement is effective.

- **min_density** is the minimum concentration density below which the refinement will not take effect.

- **zseg_num** is the z-segment number. **all_zsegm** can be used to operate on all z-segments.

- **max_double_times** affects the mesh refinement only when **method_line_base**=no. The method for refining triangles is to double the mesh density several times until the criterion is met. This parameter limits the number of mesh doubling actions to avoid getting caught in an infinite loop.

- **set_edge_split** determines how triangle edges are split during the refinement.

- **max_edge_split** is similar to **max_double_times** as it limits the number of times the same triangle edge can be split during the mesh refinement.

Examples

```
regrid variable=acceptor_conc var_diff=3.0
```

22.619 reload_mater

parameter	data type	values [defaults]
zseg_num	intg	[1]

reload_mater can be used to force the solver to re-initialize the material data for device. It may be useful when doing mesh refinement and the solver needs to re-initialize.

22.620 remove_section

parameter	data type	values [defaults]
sec_num	intg	[1]
cavity_num	intg	[2]

This statement tells the software that a given optical section does not exist in a given optical cavity; when the transfer matrix processes this section, an identity matrix will be used.

This statement is only relevant to the multicavity model; see the **begin_cavity** statement for more information.

Parameters

- **sec_num** is the section number to be removed.
- **cavity_num** is the optical cavity number in which the optical section is removed.

22.621 renumber_mater

parameter	data type	values [defaults]
xrange	realx2	[-9999. 9999.] (μm)
yrange	realx2	[-9999. 9999.] (μm)
orig_mater	intg	[1]
new_mater	intg	[1]
zseg_num	intg	[1]

This statement is used to override the material number assigned to a region. It should be used before the **load_macro** statement so that this modified region gets assigned the proper material parameters.

Parameters

- **xrange** and **yrange** describe the spatial extend of the region being modified. By default, it applies to the entire mesh plane in a given z segment.
- **orig_mater** is the original material number.
- **new_mater** is the new material number being assigned.
- **zseg_num** is the new z segment number where this statement operates.

Examples

```
renumber_mater orig_mater=1 new_mater=2
```

22.622 replace_macrofile

parameter	data type	values [defaults]
macro1	char	[void]
macro2	char	[void]

The statement **replace_macrofile** may be used to replace the two default macro files containing material parameter models. This is useful when a user would like to use his/her own macro files without modifying the default macros.

- **macro1** is used to replace the default macro file **crosslight.mac**.
- **macro2** is used to replace the default macro file **more.mac**.

```
Example(s)

replace_macrofile macro1=my_macro.mac
```

22.623 report_node

parameter	data type	values [defaults]
near_xy	realx2	[0. 0.] (μm)
plane	intg	[1]

report_node is a programmer post-processing command which reports the number of the mesh point at a specified coordinate.

22.624 restart

parameter	data type	values [defaults]
data_set	intg	99
stop_at_data_set	intg	[1]

restart may be used to restart a simulation at a previous data steps. The statement itself may be placed anywhere in the .sol file and the simulation starting point is based on the specified data set number.

Note that some parameters in the .sol file may be changed between simulation runs but parameter changes that would invalidate the loading of previous simulation data (e.g. changing the mesh structure), are not allowed.

Parameters

- **data_set** is the data set used to restart the simulation. If this number exceeds the number of available data sets, the highest available data set is used instead.
- **stop_at_data_set** stops the restarted simulation once the specified data set number has been reached. This is equivalent to using a strategically-located **stop** statement in the input file.

Examples

```
restart data_set=10
```

22.625 right_contact

left_contact is the same as **top_contact** except that the contact is placed on the right side.

One other difference is that the column number is irrelevant. The location of the contact is given by the last **layer** statement before the contact definition.

22.626 ring_structure

parameter	data type	values [defaults]
radius	real	[100.] (μm)

This statement is used to define a ring laser waveguide. Instead of left/right propagation, optical modes propagate clockwise and counter-clockwise, like in a fiber laser.

Parameters

- **radius** is the radius of the ring measured in the middle of the waveguide.

Examples

```
$ 500 um waveguide -> ring rad from center of waveguide=79.58
$ waveguide width=1.5  mesh origin=79.58-1.5/2=78.83

z_structure uniform_zseg_from=0 uniform_zseg_to=360  &&
  zseg_num=1 zplanes=5 &&
  cylindrical=yes cylindrical_origin=-78.83

ring_structure radius=79.58
```

22.627 rotation

parameter	data type	values [defaults]
angle	real	[0] (degrees)

This statement is used in a .layer file to rotate a structure. This is mostly used to align the substrate layer along the x axis for use is some optical models.

Parameters

- **angle** specifies how much the .layer is rotated when making the .geo file.

Examples

```
rotation angle=+90.0
```

22.628 rtgain_phase

parameter	data type	values [defaults]
density	real	[3.e24] (m^{-3})
point_x	real	[0.1] (μm)
zseg_num	intg	[1]

rtgain_phase is used to generate round-trip gain spectrum data using tabulated gain values at a specific carrier density so that the user can quickly judge the validity of a device design. It is used in PICS3D after the **equilibrium** statement.

The data generated by this statement is printed in a .rtd file. For VCSELs, this statement will also generate standing wave information in a .stw file.

Parameters

- **density** is the electron density used to compute the round-trip gain. The hole density will be estimated using the p/n ratio at equilibrium unless otherwise indicated.
- **point_x** is the x-position for plotting the standing wave pattern in a VCSEL.
- **zseg_num** is the z-segment number.

Examples

```
rtgain_phase density=1.5e14
```

22.629 rt3d_contact_reflector

parameter	data type	values [defaults]
transp_contact	char	[no]
transp_contact1	char	[void]
transp_contact2	char	[void]
transp_contact3	char	[void]
transp_contact4	char	[void]
contact_refl	real	[0.99]
contact_refl1	real	[0.99]
contact_refl2	real	[0.99]
contact_refl3	real	[0.99]
contact_refl4	real	[0.99]
contact1_compindex	realx2	[2. 5.]
contact1_thick	real	[0.1] (um)
contact2_compindex	realx2	[2. 5.]
contact2_thick	real	[0.1] (um)
contact3_compindex	realx2	[2. 5.]
contact3_thick	real	[0.1] (um)
contact4_compindex	realx2	[2. 5.]
contact4_thick	real	[0.1] (um)
dbr1_box1	realx2	(um)
dbr1_box2	realx2	(um)
dbr1_box3	realx2	(um)
dbr2_box1	realx2	(um)
dbr2_box2	realx2	(um)
dbr2_box3	realx2	(um)
dbr1_layer1	real	[0.1] (um)
dbr1_index1	real	[3.5]
dbr1_layer2	real	[0.1] (um)
dbr1_index2	real	[3.5]
dbr2_layer1	real	[0.1] (um)
dbr2_index1	real	[3.5]
dbr2_layer2	real	[0.1] (um)
dbr2_index2	real	[3.5]
contact1_layer2	realx3	

contact2_layer2	realx3	
contact3_layer2	realx3	
contact1_layer3	realx3	
contact2_layer3	realx3	
contact3_layer3	realx3	

rt3d_contact_reflector is used to define the optical properties of contacts in a ray tracing simulation. This statement is new to the 2009 version of APSYS and duplicates features that were previously available in the **export_raytrace** statement. By moving this definition to the post-processing stage, it is no longer required to re-run the electrical simulation to change the optical properties of contacts.

Parameters

- **transp_contact** indicates whether transparent contact is used. If "yes", a multi-layer transmission matrix model is used to determine how much light leaks through the contact. Otherwise, no light is allowed to escape and a fixed reflectivity value is used for the contact.
- **transp_contactk**, k=1..4, has the same meaning as **transp_contact** except it is used to specify only the kth contact. This parameter overrides the specification of **transp_contact**.
- **contact_refl** is the power reflectivity of an opaque contact. It is used for the model of opaque mirror only. This parameter applies to all contacts, unless otherwise specified by **contact_reflk**, k=1..4.
- **contact_reflk**, k=1..4, has the same meaning as **contact_refl** except it is used to specify only the kth contact. This parameter overrides the specification of **contact_refl**.
- **contactn_compindex** (n=1..4) is the complex refractive index of a transparent contact. It is used for the transparent contact model only. Figure 22.28 presents a list of refractive indexes of a few commonly used materials.
- **contactn_thick** (n=1..4) is the thickness of a transparent contact. It is used for the transparent contact model only.
- **dbrj_boxk** (j=1,2; k=1,2,3) is the x-y coordinates of an inside point of the jth DBR stack consisting of kth material box. The point location can be any place within the material box.

Substance		Wavelength [micrometer]							
		0.400	0.500	0.600	0.700	0.800	1.000	4.000	10.000
Aluminum	n	0.400	0.667	1.043	1.550	1.990	1.991	6.100	26.000
	k	4.450	5.573	6.568	7.000	7.050	9.300	30.400	67.300
Copper	n	-	0.880	0.186	0.150	0.170	0.197	1.940	10.520
	k	-	2.420	2.980	4.050	4.840	6.272	23.100	59.290
Germanium	n	2.300	3.400	4.500	5.150	5.270	5.100	4.350	4.300
	k	2.800	2.250	1.700	1.300	0.900	0.450	-	-
Gold	n	-	0.840	0.200	0.131	0.149	0.179	2.040	11.500
	k	-	1.840	2.897	3.842	4.654	6.044	27.900	67.500
Indium	n	-	1.020	1.290	1.390	1.500	1.760	7.600	23.800
	k	-	2.080	2.590	5.750	6.690	7.690	26.100	51.700
Lead	n	-	-	-	1.680	1.510	1.410	6.580	21.000
	k	-	-	-	3.670	4.240	5.400	20.800	37.400
Mercury	n	0.730	1.040	1.390	1.760	2.140	-	-	-
	k	3.010	3.700	4.320	4.830	5.330	-	-	-
Silver	n	0.075	0.050	0.060	0.075	0.090	0.250	2.340	10.800
	k	1.930	2.870	3.750	4.620	5.450	6.810	26.900	60.700

Figure 22.28: Complex index of refraction of some metals and semiconductors by wavelength

- **dbrj_layerl** (j=1,2; l=1,2) is the lth layer within the jth DBR stack.
- **dbrj_indexl** (j=1,2; l=1,2) is the lth real index within the jth DBR stack.

Examples

```
rt3d_contact_reflector transp_contact=no &&
  contact_refl=0.99
```

This defines all contacts as having a fixed reflection coefficient of 99%.

22.630 scale_radiative_recomb

parameter	data type	values [defaults]
mater_label	char	[]
factor	real	[1.]
mater	intg	[]

The statement **scale_radiative_recomb** may be used to artificially scale the spontaneous emission spectrum from an active region. However, it does not affect the recombination rate in passive regions or in active regions where **analytical_recomb**=*yes*.

This statement should be contrasted with **dip_factor** in **active_reg** and **set_active_reg** which affects the dipole moment and so scales both the gain and spontaneous emission spectra.

Parameters

- **mater** may be used to specify which material is being affected; a previously-defined material label may also be used instead of a number with **mater_label**.

  If no specific material is specified, then **scale_radiative_recomb** affects all active regions.

- **factor** is the scaling factor applied to the spontaneous emission spectra.

22.631 scan

parameter	data type	values [defaults]
hot_electron	char	[no]
hot_hole	char	[no]
var	char	
vari(i=2..9)	char	[void]
function_labeli(i=2..9)	char	[void]
infile_label	char	[void]
outfile_label	char	[void]
auto_finish	char	[void]
auto_condition	char	[within] above below
auto2_finish	char	[void]
auto2_condition	char	[within] above below
stop_thermal	char	[no]
solve_rtg	char	[no] yes
scan_label	char	[void]
solve_real_rtg	char	[no] yes
solve_imag_rtg	char	[no] yes
pump_no_current	char	[no] yes
value_to	real	
valuei_to(i=2..9)	real	[]
print_step	real	[]
init_step	real	[]
min_step	real	[]
max_step	real	[]
auto_until	real	[0.9]
auto_within	real	[0.01]
auto2_until	real	[0.9]
auto2_within	real	[0.01]

The **scan** statement is used by the main simulator engine to activate the equation solver while one or more control variables are being changed (scanned).

The initial state of the simulation is always set with the **equilibrium** statement. This implies that the device is under thermal equilibrium and that no external bias is applied. Therefore, the voltage and current on all electrodes is zero; any external light source or e-beam pump is also zero. The reference clock for transient simulations is also set to zero at this step.

When dealing with contacts, the electrical bias can be defined in one of two ways: voltage or current. These are mutually exclusive: when the voltage is controlled,

the current may be adjusted and when the the current is controlled, the voltage is floating. This may be used in latchup simulations: explicitly forcing an electrode to have zero current allows the voltage to freely float.

In the software, current density is a mesh-dependent vector so there must be a convention to represent the total electrode current or the average current density on the electrode as a scalar. By default, current flowing *into* a contact boundary *from* the semiconductor device is considered positive. This may conflict with other software which take an external circuit view. This convention can be adjusted using the **convention** statement.

Between successive **scan** statements, the rule is always that the value of previous control variables is always conserved unless it is overridden by something else. For example, take a simple device with 2 electrodes that is subjected to an external light pump. As we vary various control variables, we get the following scenario:

- initial state: **equilibrium** sets light=0, V1=0, V2=0, I1=0, I2=0.
- ramp voltage of electrode 1 to +1V: light=0, V1=1, V2=0, I1 and I2 to be solved
- ramp light source to +10: light=10, V1=1, V2=0, I1 and I2 to be solved
- ramp current of electrode 1 to +0.2A: light=10, I1=0.2, V2=0, V1 and I2 to be solved.

The **scan** statement works by slowly modifying a previously known solution and making small updates to the control variable(s). An updated solution is then found using a non-linear Newton algorithm with the previously known solution as the initial guess. After convergence is achieved, the solver continues to update the solution until the target value has been reached.

The size of the update steps is therefore not strictly controlled and depends on the convergence rate of the equations for a particular device and mesh layout. As shown in Figure 22.29, the progress of the scanned variable can vary from a minimum step to a maximum step. For a smoother I-V curve, smaller steps are preferable but this increases the overall simulation time. Larger steps usually result in faster simulations but this can make the I-V curve less smooth. Larger steps can also cause convergence problems: if the real solution changes too much between bias steps, the previously known solution is not a good guess for the Newton solver. See **newton_par** for details on how the bias step is controlled by the convergence rate of the Newton solver.

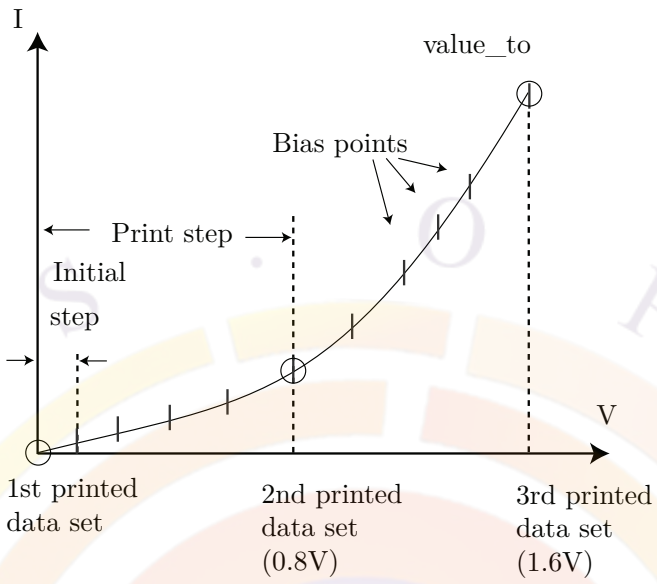


Figure 22.29: Schematics for the **scan** statement.

Parameters

- **var** is the name of the main variable to be scanned. Additional variables can be scanned at the same time using different **vari**(i=2..9) parameters. By default, the additional variables will be linearly proportional to the main variable.

  For details on the available control variables, please see the table at the end of this section.

- **function_label**

- **function_labeli**(i=2..9) is used only if more than one variable is scanned. It is used to declare or label a function relation between the ith variable with the main variable **var**. Such as function relation is further defined in a separate statement called **scan_function**. If this parameter is not used, the program assumes the default linear relation between the ith variable and the main scanned variable **var**.

  The most obvious use of this is to use *time* as the main scan variable. The function labels can then be used to define $f(t)$ relations which is especially useful in transient simulations.

- **infile_label** is the label of the initial guess data file for this **scan** statement. Without the use of this parameter, the program take the previous solution (.out file) as the initial guess. This label may be generated with the **outfile_label** parameter from a previous **equilibrium** or **scan** statement.

- **stop_thermal** instructs the main solver to stop using heat-flow model for the operation of the present statement. This means the temperature distribution will be frozen regardless bias condition.

- **value_to** is the final target value for the main scan variable defined in **var**. The initial value is determined automatically from the previous scan or from a reloaded scan when using **infile_label**.

  valuei_to(i=2..9) may also be used to define the target values of additional scan variables when not using function labels.

- **init_step** is the initial step of the bias scan. This value must be larger than **min_step**. If this parameter is not defined by user, the program defaults to 20 percent of the difference between the initial and final value of the scanned variable.

- **min_step** is the minimum step for this scan statement. If the solver is forced to make a step smaller than this value (e.g. due to convergence difficulties), the solver will automatically terminate the simulation. If this parameter is not defined by user, the program defaults it to small fraction (1×10^{-5}) of the difference between the initial and final value of the scanned variable.

- **max_step** is the maximum step size the solver is allowed to take. This can be used to force the I-V curve to be smoother or to prevent the solver from using a step size which could cause convergence difficulties. If this parameter is not defined by user, the program defaults to 50 percent of the difference between the initial and final value of the scanned variable.

- **print_step** is the interval used to print additional structural data during this bias scan. By default, structural data is automatically output at the end of the scan but **print_step** can be used to print additional data at specific intervals. An example of this can be seen in Fig. 22.29 where a value of **print_step=0.8** is used.

  Note that by necessity, the actual maximum step in the bias scan is the minimum of **print_step** and **max_step**.

- **hot_electron** and **hot_hole** instruct the solver to solve the hydrodynamic equation (energy distribution) for electrons and holes, respectively.

- **auto_finish** is the variable used to make a decision about terminating a bias scan before **value_to** is reached. This must be a variable that is being solved and not one of the control variables for the scan.

  - *current_k* or *voltage_k* (k=1..5) is the current or voltage on electrode #k.
  - *step_current_k* or *step_voltage_k* (k=1..5) is the step size of the current or voltage on electrode #k.

  - *incr_current_k* is the rate of current increase on electrode #k. This rate is measured with respect to the bias units of the main scan variable.
  - *lattice_temp* is the maximum lattice temperature.
  - *rtgain* is the peak-round-trip gain (PICS3D only).
  - *laser_power_front*, *laser_power_back* or *laser_power_total* is the laser power for the front, back or both facets, respectively (LASTIP only).

  Please note that in all cases, the absolute value of the variable is used when deciding to terminate a scan. Therefore, the user does not need to worry about the current convention on the electrode.

- **auto_condition** is condition the variable defined in **auto_finish** must meet to terminate the scan.
  - *above*: the specified variable is larger than **auto_until**
  - *below*: the specified variable is smaller than **auto_until**
  - *within*: the specified variable is approximately equal to **auto_until**, ± **auto_within**.

- **auto2_finish**, **auto2_condition**, **auto2_until** and **auto2_within** are additional termination conditions the solver must meet to prematurely terminate a scan. Both conditions must be met simultaneously: it is not an either/or condition. All statements are otherwise the same as above.

- **solve_rtg** turns on the solution of the complex round-trip gain (RTG) equation and couples the photon density to main drift-diffusion model (PICS3D only). For this parameter to work, a previous **scan** command should use **auto_finish**=*rtgain* so that the RTG equation parameters are initialized.

- **scan_label** labels the output file so it can be easily referred to by plotting tools such as CrosslightView.

  solve_real_rtg and **solve_imag_rtg** may be used to solve only the real or imaginary parts of the complex round-trip gain. However, this feature is for developer use only.

  pump_no_current is used in some nanoscale structures to apply a bias that separates the quasi-Fermi levels without actually calculating the current flow: only the Poisson equation is solved.

Examples

```
scan var=voltage_1 value_to=1.6 print_step=0.8 &&
  init_step=0.02 min_step=1.e-5 max_step=0.1
```

The above statement corresponds to Fig. 22.29.

```
scan var=time var2=voltage_1 value_to=1.e-11 &&
  value2_to=0.5 print_step=1.e-12 &&
  init_step=3.e-13 min_step=1.e-13 max_step=1.e-12  &&
```

This statement defines a transient simulation with a linear voltage ramp.

```
scan var=time value_to=50.e-12 &&
  var2=light function_label2=gs_func max_step=1.e-12
scan_function label=gs_func type=gaussian gsn_dt=4.e-12
```

This statement defines a transient Gaussian light pulse by means of a scan function.

Available Scan Variables

voltage_i (i=1,2,...,9)	Voltage of the ith electrode.
current_i (i=1,2,...,9)	Current of ith electrode in A/m for 2D and Amperes for 3D.
time	Time reference of a transient simulation.
light	A scale factor for the input light defined in commands such as **light_power** and **3d_amplifier_model**.
lighti (i=2...9)	Same as "light" above, defined when there are multiple sources of input light.
new_doping	See **equilibrium**.
bandgap_reduction	See **equilibrium**.
impact_fermi_factor	See **equilibrium**.
mobility_control	See **equilibrium**.
power_loss_scan_i(i=1..9)	Is the power loss between optical sections in PICS3D. Corresponds to the **power_loss** parameter of **couple_next** for section number i.
scale_heat_source	Artificial scaling factor for all heat sources in the device, default value is 1.
scale_impact_cr_field	See **equilibrium**.
scale_negative_stim	See **equilibrium**.
scale_polar_charge	See **equilibrium**.
surf_charge_expo	See **equilibrium**.
temperature	May be used to adjust the isothermal temperature of the device. This should not be confused with self-heating simulations which invoke the **heat_flow** statement.
wavelength_scan	The wavelength of the optical pump in **light_power**.
virtual_time	Artificial quantity used for parameter sweeps in conjunction with the **minispice** command.
xcir_resistance_i(i=1..9)	is the external circuit resistance attached to the ith contact.

22.632 scan_function

parameter	data type	values [defaults]
label	char	[mylabel]
type	char	[void]
user_datafile	char	[void]
gsn_t1	real	[0.]
gsn_dt	real	[4.e-12]
gsn_s1	real	[0.]
gsn_s2	real	[1.]
pulse_t1	real	[0.]
pulse_tr	real	[1.e-12]
pulse_dt	real	[4.e-12]
pulse_tf	real	[1.e-12]
pulse_s1	real	[0.]
pulse_s2	real	[1.]
sin_constant	real	[0.]
sin_amp	real	[1.]
sin_period	real	[1.e-9]
sin_phase	real	[0.]
cos_constant	real	[0.]
cos_amp	real	[1.]
cos_period	real	[1.e-9]
cos_phase	real	[0.]

The statement **scan_function** is used to define relation between different variables in the **scan** statement.

- **label** corresponds to the parameter of **function_labeli**(i=2..9) defining a function between the ith variable and the first (main) variable.
- **type** is the type of function. It currently supports the following types: **gaussian**, **pulse**, **sin** and **cos**. Their meanings are as suggested by their names.
- **user_datafile** if defined, is the user data file containing two columns of data to define a function relation.
- **gsn_t1** , **gsn_dt** , **gsn_s1** and **gsn_s2** are parameters t_1, Δt, s_1 and s_2 respectively, used to describe a Gaussian relation between the 1st and 2nd variable. If we use t as the 1st variable, the Gaussian function is defined as

follows:

$$
\begin{aligned}
if \quad & t_1 \le t \le t_1 + 2\Delta t \\
c_l &= 4ln(2) \\
var2 &= s_1 + \frac{s_2 - s_1}{1 - exp(-c_l)} \left[exp\left(-\left(\frac{t - t_1 - \Delta t}{\Delta t} \right)^2 c_l \right) - exp(-c_l) \right] \\
else & \\
var2 &= s_1
\end{aligned}
\tag{22.121}
$$

- **pulse__t1** , **pulse__tr** , **pulse__dt** , **pulse__tf** , **pulse__s1** and **pulse__s2** are used to define a sqaure pulse of the following shape in the code:

```
if( t>=t_1 && t<=t_1+t_r ) {
  var2=s_1+(s_2-s_1)*(t-t_1)/t_r;
} else if(t.ge.t_1+t_r.and.t.le.t_1+t_r+dt) {
  var2=s_2;
} else if(t.ge.t_1+t_r+dt.and.t.le.t_1+t_r+dt+t_f) {
  var2=s_2+(s_1-s_2)*(t-(t_1+t_r+dt))/t_f;
} else {
  var2=s_1;
}
```

- **sin__constant** , **sin__amp** , **sin__period** , and **sin__phase** are used to define a sines function of the form constant+amp*sin(2*pi/period*var+phase), where phase is in unit of radians.

- **cos__constant** , **cos__amp** , **cos__period** , and **cos__phase** are used to define a cosines function similar to that of sines above.

Example(s)

The following example is to describe a sines function:

```
scan_function label=my_sine_lable type=sin sin_constant=0 sin_amp=0.2 &&
      sin_period=1.e-9 sin_phase=0
```

22.633 section

parameter	data type	values [defaults]
grating_model	char	[kappa], 1layer, klayers (k=2..9)
active	char	[yes] no
2nd_order	char	yes [no]
dbr_period_index_profile	char	
kappa_real	real	$(1/m)$
kappa_imag	real	$(1/m)$
delta_kreal	real	[0.] $(1/m)$
delta_kimag	real	[0.] $(1/m)$
pitch	real	(m)
delta_pitch	real	[0.] (m)
length	real	[300.e-6] (m)
phase_shift	real	[0.] (π)
grating_phase	real	[0.] (π)
mesh_ratio	real	[1.]
layerj (j=1..k)	real	[0.096e-6] (m)
indexj (j=1..k)	real	
h1_real	real	(1/m)
delta_h1_real	real	[0.0] (1/m)
h1_imag	real	(1/m)
delta_h1_imag	real	[0.0] (1/m)
surface_p.c	real	
delta_s.p.c	real	[0.0]
top_refl	real	[0.3]
sec_num	intg	[1]
mesh_points	intg	[8]
dbr_period_mater	intg	
zseg_num	intg	

The statement **section** defines the optical mesh and other properties of a laser in the longitudinal direction (PICS3D only). It should be distinguished from the concept of a z-segment which defines electrical properties for a certain region. In certain cases, a section may not have any electrical mesh defined (e.g. external cavity elements). See Sec. 6.3 for details.

section is often used to directly input the grating properties of a certain region rather than computing the coupling coefficients. In that case, we can also define chirped gratings and write the z-dependent coupling coefficient as:

$$\kappa(z) = \kappa_m(z) exp(j\psi_k(z)) \quad (22.122)$$

$$\kappa_m(z) = \kappa_{m0} + \delta\kappa_m \frac{z - z_1}{z_2 - z_1} \quad (22.123)$$

$$\psi_k(z) = \psi_{k0} + \delta\psi_k \frac{z - z_1}{z_2 - z_1} \quad (22.124)$$

where z_1 and z_2 are the starting and ending points of the section.

The resulting chirp (ζ_{ch}) will be a spatial variation of the propagation constant. In units of π, this can be written as:

$$\zeta_{ch}(z) = \zeta_{ch0} + \delta\zeta_{ch} \frac{z - z_1}{z_2 - z_1} \quad (22.125)$$

Parameters

- **grating_model** is the grating model being used. It may take the following values:
  - 1 *kappa*: the coupling coefficient κ is used to describe the grating structure in an edge laser/waveguide structure (coupled-wave equation). This coupling coefficient can be defined directly or calculated by the software based on the mode profile.
  - 2 *1layer*, *2layers*, ..., klayers: this model uses plane wave transfer matrices through one or more layers and is often used to define DBR layers in VCSELs. In older versions of the software, *layer0* is equivalent to *1layer*; layer is equivalent to *2layers*.

  This parameter should not be confused with the similarly-named **grating_model** statement which is used to define other parameters needed to compute κ in edge-emitting devices.

- **active** is used to indicates that the section contains active layer. This means that this section is matched with an electrical z-segment (or a part thereof) and the optical properties will be obtained from the mesh data. For non-active (non-meshed) sections, a **passive_3d** statement must be used to define the optical properties.

- **2nd_order** turns on the terms for the second order grating (loss/gain coupling).

- **dbr_period_index_profile** can be used to define an index profile in an input file. The data should be in columns: position (in μm) and index.

- **kappa_real** and **kappa_imag** are used to represent the real and imaginary parts of the longitudinal coupling coefficient κ. They are effective only when the above parameter **grating_model**=*kappa*.

  If these values are omitted, PICS3D will compute the coupling coefficient based on the wave overlap and the grating compositions defined in the .layer file. This involves using either the **grating_compos** or **grating_model** statements. The latter should not be confused with the **grating_model** parameter defined above.

- **delta_kreal** and **delta_kimag** are used to represent the change in the real and imaginary parts of index coupling and gain/loss coupling, respectively.

- **pitch** and **delta_pitch** define the grating pitch and the change in the pitch within a section. If this is not specified, the pitch settings from the **longitudinal** statement should be used.

- **length** is the optical length of the section.

- **phase_shift** is the phase shift (in units of π) at the end of the section. For example, in a standard phase shifted DFB laser, we need to create two sections of equal length; the phase shift at the end of the first section will be in the middle of the cavity.

- **grating_phase** is the phase of the grating (Ω):

$$n = \tilde{n} + 2(\Delta n) cos\left(\frac{2\pi}{L_g} z + \Omega\right) \tag{22.126}$$

- **mesh_ratio** is the ratio between adjacent mesh intervals. **shift_center** may be used to shift the centre of the mesh point distribution. See **put_mesh** and Fig. 22.22 for details on both these parameters.

- **layerj** (j=1..k) is the thickness of the jth layer that make up one period in the layer model when parameter **grating_ model**=*klayers*.

- **indexj** (j=1..k) is the refractive index of the jth layer that make up one period in the layer model when **grating_model**=*1layer* or *klayers* (k=2..9). This older method of defining the index profile is not recommended. In newer versions of the software, new index profile capabilities have been added to the **vcsel_section** statement in the .layer file: VCSELs are the most frequent situation where this capability had been used.

  In the event this antiquated feature is required, here are several rules to keep in mind:

- If only one layer is specified, the index may be omitted. In that case, the software will compute the index of the layer based on the material macros, temperature, bias, etc...
- If more than one layer is specified, all the refractive indices must be specified.
- If there are multiple layers, negative values of the index may be defined. This instructs the software to compute changes in the average refractive index of the layers (using their absolute value) while keeping the index contrast between the layers constant.

The most accurate method in all cases is to separate each layer of the DBR into a different section, use the *1layer* model for each and omit the index to let software automatically compute it. However, this may require extra mesh and complications with the electrical solver. Defining the DBR using material macros in **vcsel_section** is also a recommended method.

- **h1_real** and **h1_imag** are the real and imaginary part of the h1 parameter describing the second order grating structure. **delta_h1_real** and **delta_h1_imag** may be used to introduce a chirp in the second order grating parameters.
- **surface_p.c** is the surface power coupling coefficient for second order grating. If not provided, the simulator computes this parameter based on the reflection coefficient at the surface (**top_refl**). As in other parameters, **delta_s.p.c** may be used to chirp this parameter.
- **sec_num** is the section number or section index.
- **mesh_points** is the number of optical mesh points in this section. Material parameters will be interpolated from the electrical mesh at these locations and used in the optical propagation model.
- **dbr_period_mater** may be used to obtain the grating index profile from a certain material number. This parameter should not generally be used as there are better ways of achieving this.
- **zseg_num** may be used to associate this optical section with a specific electrical z-segment. In general, this is not necessary as the length information will suffice to define the alignment between the optical and electrical mesh.

Examples

```
section length=800.e-6 sec_num=1 mesh_points=20
```

22.634 section_air

parameter	data type	values [defaults]
length	real	[300.e-6] (m)
sec_num	intg	[1]
mesh_points	intg	[3]

The statement **section_air** is a special case of **section**. It is used to describe an air gap within a photonic device.

- **length** is the length of the section.
- **sec_num** is the section number or section index.
- **mesh_points** is the subsection number or mesh points used for the purpose of calculation. All physical quantities are assumed to be constant within the subsection.

Example(s)

```
section_air length=200.e-6 sec_num=2 mesh_points=10
```

22.635 section_location

parameter	data type	values [defaults]
start	real	[0] (um)
end	real	[1.] (um)

- **section_location** is used to specify the starting point of the longitudinal mode cavity. If not specified, it will start from zero. For VCSEL, this statement may be automatically generated by the LAYER program.
- **start** is the starting coordinate of the optical cavity.
- **end** is the ending coordinate of the optical cavity.

Example(s)

```
section_location start=2.  end=12.5
```

22.636 self_consistent

parameter	data type	values [defaults]
conditional_off	char	[no]
impose_limit	char	[no]
sheet_charge_model	char	[no]
onetime_update	char	
onetime_resume	char	[no]
loop_num	intg	[9]
electrode_off	intg	[1]
wave_range	real	[0.005] (um)
sample_distance	real	(um)
change_limit	real	[0.1]
voltage_off	real	[-9999.]
current_off	real	[-9999.]
wave_range_bottom	real	(um)
wave_range_top	real	(um)
wave_range_left	real	(um)
wave_range_right	real	(um)
onetime_ref_voltage	real	(V)
onetime_ref_current	real	(2D: A/m, 3D: A)
sample_xcor	real	(um)
sample_ycor	real	(um)
sample_xrange	realx2	(um)
sample_yrange	realx2	(um)
autosample_y	real	(um)
autosample_xrange	realx2	(um)

The **self_consistent** statement is used in the main solution file (.sol) to invoke the self-consistent quantum well carrier density model. It can also be used in the gain preview module (.gain): an external field can be supplied with **gain_module**. This model is required to model the quantum-confined Stark effect (QCSE) that occurs in QW modulators and wurtzite piezoelectric materials.

Without this statement, a flat-band model is used in the Schrödinger solver. See Sections 8.1.2 and 8.2 for more details about the difference between these two models. In general, the self-consistent model is more accurate but more time-consuming.

The self-consistent quantum mechanical solver finds quantized energy states which may appear or disappear when the potential profile is changed. Such appearance or disappearance may cause the global Newton solver to be poorly initialized and thus diverge. In such a case, one may wish to turn off the self-consistent solver conditional

upon the bias being beyond a certain value. This means the density of states (DOS) of the QW/QDOT will be frozen at a value before the conditional turn-off.

Parameters

- **wave_range** is used to define a range for the solution of the quantum mechanical wave equation. Refer to Sec. 8.2.3 for details.

  The wave range value is the same in all directions unless overridden in a particular direction by **wave_range_bottom**, **wave_range_top**, **wave_range_left** or **wave_range_right**.

- **sample_distance**, if specified, gives the position of a 1D cut across the well and barrier layers used to sample the potential. This 1D potential profile is used to solve the quantum levels. By default, a value in the middle of the quantum well region is used.

- **loop_num** is the maximum number of iteration in equilibrium that is need to reach self-consistency for both the potential and quantum confined carrier density profile.

- **conditional_off** is to indicate whether the self-consistent update of quantum mechanical wave solver is to be turned for convergence reason when certain conditions are reached.

- **electrode_off** is used only when **conditional_off**=*yes*. It defines the electrode used as a condition to turn of the self-consistent update solution.

- **sheet_charge_model** is a simplification to the model which can help with convergence in some cases. When turned on, the total sheet charge and the position of the QW levels are calculated self-consistently. However, the carrier density distribution does not follow the self-consistent model.

- **impose_limit** indicates if a limit is imposed on relative change of potential for self-consistent calculation. Too large a change is thought to be harmful to convergence. One may turn on this option if non-convergence due to quantum confinement is encountered.

- **change_limit** is to impose a limit on relative change of potential profile in the quantum solution. The simulator calculates the average applied field and will only allow the profile to be changed by a certain amount.

- **voltage_off** defines a voltage condition under which the self-consistent updates may be stopped.

- **current_off** defines a current condition under which the self-consistent updates may be stopped. The current units (2D or 3D) are the same as the ones used in the **scan** statement.

- **sample_xcor** and **sample_ycor** are (x,y) coordinates used to sample the potential profile in 3D simulations where the QW normal is in the z direction.

- **sample_xrange** and **sample_yrange** may be used in rare cases where a complex MQW region with a weird shape is used. It restricts the use of the self-consistent code to a rectangular region within that complex MQW.

- **onetime_update** turns on a technique which can help improve convergence in cases where a sudden shift in the number of quantum states introduces convergence difficulties: this discontinuity may conflict with the initial guess to the Newton solver, which is extrapolated from the previous solution.

  When using this technique, the quantum states are frozen at a certain reference bias (after turn-on) to improve convergence. Multiple simulation runs must be used to set up this initial state:

  1. Set **onetime_update**=*initial* to save the quantum states for the next run. The solver does not update quantum solutions at this stage.
  2. Set **onetime_update**=*repeat*. The solver imports the potential saved by the previous run and updates the quantum solution at the imported reference bias one time only (hence the name).
  3. Repeat previous step as needed to improve self-consistency.

  We note in passing that the extrapolation for the initial guess to the Newton solver can be turned off in **newton_par**. Users may wish to compare the two methods to see which yields the fastest and most reliable convergence.

- **onetime_ref_voltage** and **onetime_ref_current** set the reference bias for the "onetime" update method described above. **onetime_resume** determines whether the normal update of quantum states is allowed to resume once this reference bias has been reached.

- **autosample_y** and **autosample_xrange** may be used to automatically change the sampling position for self-consistency in a 3D simulation. An example of this would be a 3D Mach-Zehnder modulator with BPM propagation in the Y-branch and where the field under the waveguide better represents the device physics than the field in the middle of the QW region.

  With the auto-sample parameters defined, the software cuts a line at a specific y position and looks for a semiconductor material within a certain x range. The sampling position is set to the middle of the region found within that range: presumably, this is the middle of a ridge and will be close to the peak of the optical mode.

Examples

```
self_consistent loop_num=15
```

Changes the number of self-consistent iterations.

```
self_consistent wave_range=0.05 &&
  electrode_off=1 conditional_off=yes voltage_off=5
```

For convergence reasons, the quantum states are not updated once the voltage on electrode # 1 reaches 5 volts. This may cause some inaccuracy.

22.637 set_3dray_internal_interface

parameter	data type	values [defaults]
model	char	[fixed_index]
import_filei,i=1..9	char	void
thicknessi,i=1..9	real	[0.1] (um)
refr_indexi,i=1..9	realx2	[2.1245, 0.0]
inside_ref_point	realx3	[0.0, 0.0, 0.0] (um)
outside_ref_point	realx3	[0.0, 0.0, 0.0] (um)
nlay	intg	[1]

set_3dray_internal_interface is used to specify special internal interfaces for the 3D raytracing program. For example, this can be used to represent surface coatings that are inside of the structure's geometry. For coatings on the outer surface of a structure, the **surface_model** parameter of **do_raytrace_3d** should be used instead. To define the position of an interface, the user must enter the XYZ coordinates of two points on either side of the interface.

The number of layers in the coating is controlled by **nlay**. The parameters numbered 1 to 9 refer to that particular layer. Layer #1 is defined as the one closest to **inside_ref_point**.

The complex refractive index of each layer is controlled by the **model** parameter. If this is set to *fixed_index*, then the value of the **refr_index** parameter will be used.

If *import_data* is used instead, then the index will be read from a text file specified by the **import_file** parameter. This file must be formatted so that the first column contains the wavelength (in μm), the second column, the real part of the refractive index and the third column, the imaginary part of the refractive index.

22.638 set_3dray_mirror

parameter	data type	values [defaults]
mirror1_side	char	[void] -x,+x,-y,+y,-z,+z
mirror2_side	char	[void] -x,+x,-y,+y,-z,+z
mirror3_side	char	[void] -x,+x,-y,+y,-z,+z

set_3dray_mirror is used to define symmetric boundary conditions for the 3D raytracing program (**do_raytrace_3d**. If this statement has been used in a 3D raytracing simulation in a different .plt file, it must also be used before plotting the results to enforce the correct symmetry of the plots.

22.639 set_active_reg

set_active_reg is used to override active region parameters previously defined in **active_reg**. Most of the material parameters used for the two commands are virtually identical and so are omitted here.

Note that by default, this command applies to all the previously declared active regions so it is often used in the .sol or .gain files to alter the gain model settings that were automatically declared when processing the .layer file.

22.640 set_barrier_width

parameter	data type	values [defaults]
barrier_width_bot	real	[0.1] (um)
barrier_width_top	real	[0.1] (um)
active_mater	intg	[1]

set_barrier_width is used to let the program take into account the barrier composition grading when modeling the carrier confinement of a quantum well.

Note that this command only applies to basic QW macros and is somewhat obsolete: it is strongly recommended to use the complex MQW macros (labeled with a cx- prefix) or the simplified complex QW library scheme from Sec. 3.5.1 instead.

Parameters

- **barrier_width_bot** is the bottom barrier width.
- **barrier_width_top** is the top barrier width.
- **active_mater** is the active layer material number of the quantum well related to the barrier described in this statement.

Examples

```
set_barrier_width barrier_width_bot=0.01 barrier_width_top=0.01 active_mater=3
```

22.641 set_fdtd_interface

parameter	data type	values [defaults]
mater_in	intg	
mater_out	intg	
angle_to_axis	char	[+x],-x,+y,-y,+z,-z
angular_data	char	[fdtd_data.dat]

The **set_fdtd_interface** statement is used in the Optowizard solution file to override the reflection properties of a specified material interface for the purposes of ray tracing. Instead of a Fresnel reflection model, reflection/transmission data from a FDTD solver will be used at the interface: this allows nanoscale effects to be considered inside a larger-scale raytracing model.

Parameters

- **mater_in** and **mater_out** are used to specify the interface affected by this command and the direction of the incoming beam.
- **angle_to_axis** determines a reference axis to match the incident angle of incoming rays to angular data from FDTD solver.
- **angular_data** is the data file containing angular reflection/transmition data from the FDTD solver.

22.642 set_include

set_include defines the relative path for all subsequent **include** statements.

Parameters

relative_path specifies the path for the **include** statements. If a relative path is specified, it is relative to the directory of the input file where **set_include** has been used.

22.643 set_index

parameter	data type	values [defaults]
mater	intg	[-9999]
real_index	real	[3.5]
absorption	real	[200./m]
index_file	char	[void]

The **set_index** statement is used in the optowizard solution file to override the refractive index (n, k) of a specified material for the purposes of ray tracing. If not specified, the index from the electrical simulation in APSYS will be used in the ray tracing.

Exported index data from APSYS is often only for a single wavelength so overriding it is often recommended for more accurate results. It can also be useful to override the index to change something without having to re-run the electrical simulation.

Parameters

- **mater** is used to specify the material number being overridden.
- **real_index** is to specify a fixed real index(n) value for the specified material.
- **absorption** is to specify a fixed absorption coefficient(m^{-1}) for the specified material.
- **index_file** is to specify the file name of the data file describing the index spectrum for the specified material. The data must be in text format where the first column is the wavelength in micrometers, the second column is the real part of the index and the third column is the absorption coefficient (m^{-1}).

Examples

```
set_index mater=1 real_index=2.8 absorption=2000
```

22.644 set_initial_stress

set_initial_stress defines the initial or pre-existing stress in a device. This should be contrasted with external stress applied on the device, which is defined in **set_stress**.

All parameters for this command are the same as in **set_stress**.

22.645 set_lp_mode_index

parameter	data type	values [defaults]
lp_mode_num	intg	[1]
m_index	intg	[0]
n_index	intg	[1]

This statement is used in PICS3D VCSEL simulations to replace the results from the normal mode solver with one of the linearly polarized fiber modes (LP_{mn}). These modes may be difficult to find using normal effective index sorting procedures.

Parameters

- **lp_mode_num** is the number of the lateral/radial mode being specified.
- **m_index** and **n_index** are the indices of the linearly polarized mode.

Examples

```
multimode mode_num=4
set_lp_mode_index lp_mode_num=1 m_index=0 n_index=1
set_lp_mode_index lp_mode_num=2 m_index=1 n_index=1
set_lp_mode_index lp_mode_num=3 m_index=2 n_index=1
set_lp_mode_index lp_mode_num=4 m_index=0 n_index=2
```

22.646 set_minority_carrier

The command **set_minority_carrier** may be used to artificially reduce the bandgap so that larger minority carrier density can be used to overcome possible convergence problems.

The parameter **virtual_eg_kt** can be used to set a reduced bandgap in unit of kT where k is the Boltzman constant.

Example(s)

```
set_minority_carrier virtual_eg_kt=30
```

would set the bandgap to be 30 kT for the purpose of computing minnority carriers.

22.647 set_negative_stim

parameter	data type	values [defaults]
max_absorption	real	[1e4] (m^{-1})
zseg_num	intg	[1]

set_negative_stim artificially caps the the absorption coefficient in a given z-segment.

Note that a similar adjustable scaling coefficient may be found in the **equilibrium** and **scan** statements.

Parameters

- **max_absorption** is the maximum absorption value allowed: values smaller than this parameter will be left unchanged.
- **zseg_num** is the z-segment number where this command is applied.

22.648 set_polarization

Important changes

As of the 2011 version of the software, a newer model based on the **polarization_charge** macro and **polarization_charge_model** statement is available.

This new model offers a great deal more flexibility to the user so **set_polarization** may be obsoleted in a future release.

Note that the two models should not be used in conjunction as this would result in double-counting of the interface charges.

parameter	data type	values [defaults]
xy_plane	char	[no]
alias_macroname_table	char	
screening	real	[0.5]
substrate_latt	real	[3.189] (Å)
hex_lattice_a0	real	[3.189] (Å)
hex_lattice_c0	real	[5.185] (Å)
external_stress_xx	real	[0.] (GPa)
external_stress_yy	real	[0.] (GPa)
ref_column	intg	[1]
growth_plane_miller_index	intgx4	

set_polarization is used in the .layer file to automatically generate **interface** statements and define the piezoelectric charges found in nitride materials.

The polarization formulas[117] used for these calculations are hardcoded. Only the following material macros are recognized: algan, ingan, alinn, gaalinn, and gan.

Parameters

- **xy_plane** would produce interface charge on the x-y plane for a 3D simulation. The interface charge would be defined using the **doping** command.
- **alias_macroname_table** is the name of a data file which can be used to define how certain materials are renamed; the file should contain two columns with the original name in the first column and the new name in the second column. This should be used whenever layer.exe needs to know that a certain material has been renamed and no longer matches the default Crosslight macros.

  Under normal circumstances, the .layer file is agnostic with regards to the names assigned to the various materials in each layer: these names are passed un-changed and are not used until the main simulator attempts to use the **load_macro** statement. In this scenario, macros may be renamed freely and the material macro need not even exist at the time when the .layer file is processed: it only has to exist when the main simulator is launched.

An exception to this rule is when layer.exe calculates the interface polarization because the **set_polarization** command has been used. In that case, layer.exe needs knowledge of the macro names because only a handful of predetermined formulas for GaN-based materials are encoded into the program. Using aliases allows layer.exe to keep doing this job even if those materials have been renamed.

- **screening** is the screening factor used to account for deviations from the theory (e.g. screening by fixed defects, partial relaxation, etc...). A factor of unity means 100 percent of the theoretical interface charge is used.

  This parameter affects the whole device but may be adjusted locally by using **adjust_layer_screening** and **adjust_column_screening**. This may be used to account for surface states or passivation without affecting the whole device.

- **ref_column** is a reference column of the layer structure which represents the complete layer stack. In newer versions of the software, the strain and piezo charge should be determined on a per-column basis so this parameter should not be used.

- **substrate_latt** is the reference lattice used for the strain calculations. By default, this value is set to GaN. Please note that this is not necessarily the lattice of the substrate: in many growth designs, a relaxed buffer is deliberately grown on the actual substrate to improve the material quality of the top layers. In that case, the lattice constant of the buffer should be used here.

  Users working on deep UV devices on AlN substrates should also take care to use the right reference lattice as this is necessary to get the right type of strain.

- **external_stress_xx** and **external_stress_yy** can be used to define externally applied stress on the device.

- **growth_plane_miller_index** can be used to specify the crystal orientation of the growth using the (h k i l) notation. The software will then calculate the direction of the polarization vector based on **hex_lattice_a0** and **hex_lattice_c0** and scale the interface charge with the appropriate cosine factor.

Examples

```
set_polarization screening=0.7
```

This would allow 70 percent of the total theoretical value be used on the layer interfaces.

22.649 set_screening_factor

parameter	data type	values [defaults]
x1_label	char	
x2_label	char	
y1_label	char	
y2_label	char	
at_x_label	char	
at_y_label	char	
screening	real	[0.5]
xrange	realx2	[-1.e9 1.e9] (um)
yrange	realx2	[-1.e9 1.e9] (um)
at_x	real	(um)
at_y	real	(um)
at_point_range	real	[0.0005] (um)
zseg_num	intg	

set_screening_factor is used to locally override the global screening coefficient defined in the **polarization_charge_model** statement. This can be used to account for regions of compensating defects or interface states and alter the amount of fixed piezoelectric charges in a particular region.

Parameters

- **x1_label**,**x2_label**,**y1_label** and **y2_label** are position labels used to define a 2D region where the screening coefficient is modified. Absolute coordinates can also be used directly with **xrange** and **yrange**: this is the default behavior.
- **at_x_label** and **at_y_label** are position labels used to define a specific interface where the screening coefficient is modified. Absolute coordinates can also be used with **at_x** and **at_y**.

  Specifying an interface will override the default behavior of defining a large 2D region for the screening override.
- **screening** is the new locally-defined screening coefficient used to compute the fixed charges.
- **at_point_range** helps locate the interface defined with **at_x** and related commands: it defines a small search range to account for fuzzyness in the mesh point location.

- **zseg_num** is the z-segment number where this statement applies.

Examples

```
set_screening_factor screening=0.0 at_y=0.0
```

22.650 set_stress

parameter	data type	values [defaults]
all_mater	char	[yes]
mater_label	char	
xx	real	[0] (GPa)
yy	real	[0] (GPa)
xy	real	[0] (GPa)
zz	real	[0] (GPa)
xz	real	[0] (GPa)
yz	real	[0] (GPa)
mater	intg	[]

set_stress defines the external stress applied on a device. This should be contrasted with pre-existing stress defined in **set_initial_stress**.

Parameters

- **xx**,**yy**,**zz**,**xy**,**xz** and **yz** define the stress components σ_{ij} applied on the device.
- **all_mater** determines whether the stress is applied to all layers. In not, then a specific material where the stress exists can be specified with **mater**.
- **mater_label** may be used instead of **mater** if a label alias has previously been defined for the material.

22.651 set_temperature

set_temperature resets the isothermal temperature of a simulation. It is used when issuing the **equilibrium** statement several times in the same simulation to define a new thermal equilibrium point.

All parameters are the same as in the **temperature** statement.

22.652 set_wavelength

parameter	data type	values [defaults]
wavelength	real	[0.5] (um)
backg_loss	real	[500] (1/m)

set_wavelength is used to define some critical optical properties of a device. It is used to initialize the optical mode distribution and internal loss.

- **wavelength** is the operating optical wavelength.
- **backg_loss** is the background loss coefficient. By default, regions outside the active region are assumed to have a constant loss background (which can be determined experimentally). Alternatively one can use the material statement **absorption** to set the material loss in different materials. A non-zero **absorption** setting will override the **backg_loss** here.

Example(s)

```
set_wavelength wavelength=0.5 backg_loss=1000.
```

22.653 set_xydata_for_scan

parameter	data type	values [defaults]
scan_var	char	[void]
scan_var_scale	char	[linear]
scan_var_facet	char	[void]
scan_var_scale_lit	real	[2.]
scan_var_scale_curr	real	[2.]
scan_var_scale_horizontal	real	[1.]
scan_var_horizontal_power	real	[-9999.d0]
scan_var_mode_index	intg	[1]
scanline	intg	
scan_num	intg	

set_xydata_for_scan is a post-processor statement used to set plotting options affecting subsequent plots. It is used to convert structure data (xy-data) into bias dependent plots.

- **scanvar** is used to define scan variables to be used in AC analysis plots. The scan variables are the same as used in the **plot_scan** statement. Please refer to **plot_scan** for more details.

  scanvar_xxx, all parameters with prefix scanvar_ are the same as those (without the prefix) defined in **plot_scan** statement. Please refer to **plot_scan** for more details.

- **scanline** is used to denote bias data from a whole **scan** statement. The data from statement **equilibrium** is counted as the scanline=1; those from the next **scan** statement counted as scanline=2, etc. For this parameter to work, the **get_data** command must already include the data sets associated with this parameter.

- **scan_num** is used to denote bias data from a whole **scan** statement. The data from statement **equilibrium** is counted as the scan_num=0; those from the next **scan** statement counted as scanline=1, etc. For this parameter to work, the **get_data** command must already include the data sets associated with this parameter.

```
set_xydata_for_scan scanvar=voltage_1
```

The above statement will cause the program plot small signal quantities (regarded as xy-data or structural data) versus voltage bias at electrode number 1.

22.654 setup_raytrace

parameter	data type	values [defaults]
filebase	char	

setup_raytrace is used in the main Optowizard input file to define the data sets generated by a previous APSYS simulation.

Parameters

- **filebase** is the root filename of the original simulation data, minus the extension (such as .sol or .out).

Examples

```
setup_raytrace filebase=tip
```

22.655 shal_acpt_level

shal_acpt_level is used to define or override the shallow acceptor level of the p-type dopants. The level is measured from the valence band and in the unit of eV. **shal_acpt_level_i** may also be used when several species of dopants are present: "i" is a placeholder for the species number.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.656 shal_acpt_level_i

shal_acpt_level_i is used to define or override the shallow acceptor level of the of the ith p-type dopant. The level is measured from the valence band and in the unit of eV. See also **shal_acpt_level**.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.657 shal_dnr_level

shal_dnr_level is used to define or override the shallow acceptor level of the n-type dopants. The level is measured from the conduction band and in the unit of eV. **shal_dnr_level_i** may also be used when several species of dopants are present: "i" is a placeholder for the species number.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.658 shal_dnr_level_i

shal_dnr_level_i is used to define or override the shallow acceptor level of the of the ith n-type dopant. The level is measured from the conduction band and in the unit of eV. See also **shal_drn_level**.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.659 shear_modulus

shear_modulus defines the shear modulus for the acoustic wave propagation model in SAWAVE.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.660 shift_affinity

parameter	data type	values [defaults]
delta_affinity	real	[0.0] (eV)
mater	intg	[1]

shift_affinity is used to shift the electron affinity for the whole material. This is very convenient for fused junction devices where two completely different material systems are forced together. This statement may be used to control the band offset easily instead of defining a new affinity for each material.

- **delta_affinity** is the amount of shift in electron affinity.
- **mater** is the material number being affected.

Example(s)

```
shift_affinity delta_affinity=0.05 mater=3
shift_affinity delta_affinity=0.05 mater=4
shift_affinity delta_affinity=0.05 mater=5
```

The above example shifts the conduction band downwards by 0.05 eV for materials with material numbers of 3, 4, and 5.

22.661 solve_lateral_wave

parameter	data type	values [defaults]
export_data	char	[void]
cond_valley	intg	[1]
val_valley	intg	[1]

solve_lateral_wave activates the eigen equation solver to solve either the Maxwell wave equation or the Schrödinger quantum mechanical wave equations. This statement is similar to statements like **equilibrium** and **scan** in that it is the main action of the solver.

- **export_data** is used to export the solution of eigen equation solver. If several of the same statements are used in sequence, only the first statement needs to define the exported file. All other statements can share the same data file.
- **cond_valley** and **band_valley** define the conduction and valence band valleys being solved.

Examples

```
solve_lateral_wave cond_valley=1 val_valley=1 &&
  export_data=cone.qdd
```

The above statement solves Gamma and HH bands and exports the data to file cone.qdd.

22.662 sp.rate_wavel

parameter	data type	values [defaults]
data_file	char	
auto_pn_ratio	char	[no]
field_dep_curves	char	[no]
wavel_range	realx2	(μm)
conc_range	realx2	[1.e23 1.e24] (m^{-3})
pn_ratio	real	[1]
av_index	real	[3.3]
field_range	realx2 [1.e5 1.e7]	
curve_number	intg	
data_point	intg	

sp.rate_wavel plots the spontaneous emission rate versus wavelength.

- **data_file** is the file to which the graphic data is written in ASCII format.

- **auto_pn_ratio** is used to indicate if automatic setting of hole/electron density ratio is used in PICS3D simulation. If positive, averaged hole/electron density from the drift-diffusion solver is used. This parameter, if positive, will override **pn_ratio**.
- **field_dep_curves** indicates whether ot not the curves are generated with different applied electrical field intensities. If this is set to no, the curves are plotted with different carrier concentrations while the field is set to zero. If this is set to yes, the carrier concentration is set to the initial value in **conc_range** while the field is varied.
- **av_index** is the estimated average refractive index.
- **wavel_range** is the wavelength range.
- **conc_range** is the electron concentration range in the well.
- **pn_ratio** is the ratio of hole over electron concentrations. Note that this ratio can be set to an arbitrary number in the gain preview. In the main solver, this ratio is automatically determined by the simulator according to the local Fermi levels.
- **field_range** is the range of the applied field for producing the different curves. This parameter is active only if **field_dep_curves** is set to yes.
- **curve_number** is the number of spectral curves to be plotted for different carrier concentrations.
- **data_point** is the number of data points in one curve.

Example(s)

```
sp.rate_wavel wavel_range=(1.0 1.4) &&
conc_range=(5.e23 5.e24)  curve_number=5 data_point=100
```

22.663 sparse_eigen_solver

parameter	data type	values [defaults]
use_mf	char	[yes],no
fast_dynamic_search	char	[no],yes

sparse_eigen_solver is used to control the sparse eigen solver used in the .sol file. These include lateral mode solver for wave guide modes and quantum mechanical wave solver for quantum wells.

Parameters

- **use_mf** controls the use of a multi-frontal sparse matrix solver in the eigenvalue search.
- **fast_dynamic_search** indicates whether a fast dynamic mode search technique is used.

Examples

```
sparse_eigen_solver use_mf=yes
```

22.664 special_suprem_contact

parameter	data type	values [defaults]
type	char	[sym_polygon], surrounding, polygon
polygon_datafile	char	
mater_label	char	
insul_semi_interf	char	[no],yes
polygon_center_x	real	[1.0] (um)
polygon_center_y	real	[1.0] (um)
radius	real	[1.0] (um)
inner_radius	real	[0.0] (um)
num	intg	[1]
touch_mater	intg	
zseg_num	intg	
polygon_edge	intg	[6]

special_suprem_contact is a variation of **suprem_contact** which allows for more complex contact shapes when importing structures from CSUPREM.

Parameters

Many of the parameters in this statement are the same as in **sym_polygon_for_semicrafter** and **contact**.

- **type** Defines the shape of the contact:

  - *sym_polygon* defines a symmetric polygon.
  - *surrounding* indicates that the entire perimeter of the mesh plane is the contact boundary.
  - *polygon* is a user-specified polygon.

- **polygon_datafile** is a text file containing information about a user-specified polygon.
- **polygon_center_x** and **polygon_center_y** are the (x,y) coordinates of the center of the symmetric polygon.
- **radius** and **inner_radius** are used when **type**=*sym_polygon*. Radius is defined as the distance between the center of the polygon and the corners, i.e. the radius of the circle which encompasses the polygon. The contact area is the space between the two polygons; it forms a sort of "ring" if the inner radius is larger than zero.
- **num** is the contact number.
- **touch_mater** determines the material number which serves as the reference for the contact. If a label has previously been defined as an alias for this material, **mater_label** may be used instead.
- **zseg_num** is the z-segment number where the contact is located.
- **polygon_edge** is the number of edges of the symmetric polygon.
- **insul_semi_interf** instructs the software to look for a semiconductor/insulator interface within the specified range and define this interface as the contact boundary. This method overrides the shape of the contact region that would otherwise be produced by this command.

Examples

```
contact num=1
special_suprem_contact type=sym_polygon &&
  polygon_center_x=0.6 polygon_center_y=0.6 &&
  inner_radius=0.301 &&
  radius=0.34 num=1 touch_mater=1 zseg_num=1 polygon_edge=6
```

22.665 spec_heat

parameter	data type	values [defaults]
(see) material_par		

The material statement **spec_heat** is used to define the specific heat (in J/kg/K) of the semiconductor material.

The use of this parameter and related examples are given under **material_par** in section 22.456.

22.666 splot_xyz

parameter	data type	values [defaults]
variable	char	
data_file	char	
wave_option	char	[right],total,left
xy_from	realx2	
xy_to	realx2	
view_xrot	real	[0.]
view_zrot	real	[0.]
xrange	realx2	
yrange	realx2	
zrange	realx2	
z_min	real	
z_max	real	
mode_index	intg	[1]
grid_sizes	intgx2	[20, 20]
trap_index	intg	[1]

splot_xyz is a post-processor statement used to plot structural data on a 2D plane. Unlike **plot_2d**, this command can only generate 3D surface plots of scalar variables.

Other related commands exist and should be used depending on the 2D/3D nature of the original simulation results. The following rules apply:

- 2D simulations: use **plot_2d**
- 3D cylindrical simulations with one mesh plane: use **plot_2d**
- xy plane from a 3D simulation: use **cplot_xy** for contour plots of scalar variables, **splot_xy** for 3D surface plots of scalar variables or **vplot_xy** for vector variables.

- xyz plane from a 3D simulation: use **cplot_xyz** for contour plots of scalar variables, **splot_xyz** for 3D surface plots of scalar variables or **vplot_xyz** for vector variables.

Parameters

With the exception of the z-position and vector plot-related parameters that do not apply, all parameters are the same as in **plot_2d**.

- **wave_option** controls the plotting of certain wave-related variables in PICS3D.
- **xy_from** and **xy_to** define the (x,y) corners of the plotting plane.
- **view_xrot** and **view_zrot** rotate the 3D plot for viewing purposes.
- **xrange**,**yrange** and **zrange** respectively control the display ranges of the x,y and z axes.
- **z_min** and **z_max** define the z coordinates of the corners of the plotting plane.

Examples

```
splot_xyz variable=elec_conc xy_from=(0.5, 1.4970) xy_to=(0.5 1.5030) &&
  grid_sizes=(60, 20) view_zrot=20.
```

22.667 splot_xy

parameter	data type	values [defaults]
variable	char	(see list)
data_file	char	
integration	char	[no], yes
math_oper	char	[void],log10,power10
point_ll	realx2	(μm)
point_ur	realx2	(μm)
view_xrot	real	[0.]
view_zrot	real	[0.]
xrange	realx2	
yrange	realx2	
zrange	realx2	
z	realx2	
integration_xrange	realx2	[-1.e4 1.e4] (μm)
integration_yrange	realx2	[-1.e4 1.e4] (μm)
grid_sizes	intgx2	[20, 20]
mode_index	intg	[1]
trap_index	intg	[1]

splot_xy is a post-processor statement used to plot structural data on a 2D plane. Unlike **plot_2d**, this command can only generate 3D surface plots of scalar variables.

Other related commands exist and should be used depending on the 2D/3D nature of the original simulation results. The following rules apply:

- 2D simulations: use **plot_2d**
- 3D cylindrical simulations with one mesh plane: use **plot_2d**
- xy plane from a 3D simulation: use **cplot_xy** for contour plots of scalar variables, **splot_xy** for 3D surface plots of scalar variables or **vplot_xy** for vector variables.
- xyz plane from a 3D simulation: use **cplot_xyz** for contour plots of scalar variables, **splot_xyz** for 3D surface plots of scalar variables or **vplot_xyz** for vector variables.

Parameters

With the exception of the z-position and vector plot-related parameters that do not apply, all parameters are the same as in **plot_2d**.

- **integration** tells the software to integrate the variable being plotted over **integration_xrange** and **integration_yrange** and display the results in the plotting log (.plt.msg).
- **view_xrot** and **view_zrot** rotate the 3D plot for viewing purposes.
- **xrange**,**yrange** and **zrange** respectively control the display ranges of the x,y and z axes.
- **z** is the position on the z-axis for the 2D plot. If necessary, the variable data will be interpolated from neighboring mesh planes.

Examples

```
splot_xy variable=elec_conc z=50. grid_sizes=(35, 35) &&
  xrange=(0. 1.5) yrange=(0.0 3.0) view_xrot=0 view_zrot=30
```

22.668 spont_charge

This statement is used in the material macro files to define the spontaneous part of the polarization vector ($P_{sp}(z)$ in the literature) which leads to interface charges in GaN materials. This is similar to the charges created through the use of the **polarization_charge** macro command except that with **spont_charge**, the strain-induced component of the polarization vector is calculated automatically using piezoelectric tensor elements such as **e15_bulk**.

Note that the piezoelectric tensor elements MUST be defined in the macro when using this statement; otherwise they will default to zero which will lead to an incorrect interface charge.

This macro parameter must be enabled by using the **polarization_charge_model** command in the .sol file; **input_piezo_param**=*yes* must be used to enable this model instead of **polarization_charge**.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.669 start_loop

parameter	data type	values [defaults]
symbol	char	[void]
value_from	intg	[1]
value_to	intg	[2]
step	intg	[1]

start_loop is a statement used to repeat a sequence of commands. A symbolic variable is defined as part of the loop and serves to control the number of iterations.

The sequence to be repeated must be terminated by a **end_loop** statement.

Parameters

- **symbol** is the symbol used for the loop variable.
- **value_from** and **value_to** are the starting and ending values, respectively, of the loop variable.
- **step** is increase of the loop variable at each iteration.

Examples

```
start_loop symbol=%j value_from=1 value_to=3  step=1
scan var=voltage_%j value_to=0.5
end_loop
```

The above commands cause the symbol %j to take values of 1,2 and 3 and repeat the **scan** command accordingly. The equivalent commands without the loop are as follows:

```
scan var=voltage_1 value_to=0.5
scan var=voltage_2 value_to=0.5
scan var=voltage_3 value_to=0.5
```

22.670 start_qwire_complex

parameter	data type	values [defaults]
column_start	intg	[1]
column_end	intg	[3]

start_qwire_complex is used in the .layer file to define the xy cross-section of a 2D quantum-confined region (i.e. a quantum wire). All the layers and columns between this statement and **end_qwire_complex** form part of the quantum wire region. Note that in order to use this statement, complex MQW macros (normally labeled starting with "cx-") must be used to define the materials in the wire cross-section.

When the .layer file is processed after using this statement, specialized commands including **begin_qwire_complex** and **qwire_complex_region** will be defined in the .mater file to define the quantum wire. Without this statement, these complex MQW regions would merely form a quantum well region with 1D confinement.

Parameters

- **column_start** and **column_end** are the column numbers where the quantum wire respectively begins and ends.

22.671 start_same_complex

parameter	data type	values [defaults]
tag	char	[void]

start_same_complex is used in the .layer file to define a smaller complex MQW chunk of a larger superlattice.

The quantum mechanical solver in the software does not handle periodic boundary conditions. Instead, an infinite potential well is used on the outer edges of the MQW region. Since it is numerically unfeasible of solving the QW states of a superlattice consisting of dozens of periods with this kind of boundary condition, a common approximation is to define a smaller chunk of 5-10 periods and use these QW states to represent the entire superlattice.

Using **start_same_complex** allows the smaller superlattice chunk to be repeated many times in the input files but lets the software know that all these complex MQW regions are the same which can help save on computation time

start_same_complex affects all complex MQW regions defined after this statement; its effects are terminated by the matching **end_same_complex** statement.

Parameters

- **tag** is a user-defined label identifying the complex MQW region.

Examples

```
start_loop symbol=%i value_from=1 value_to=25
start_same_complex  tag=period4well

.....

isolate_complex
end_same_complex

end_loop
```

22.672 stop

This statement forces the solver to stop: it has no parameters.

22.673 strain_bar

strain_bar is an active layer macro statement used to define the strain in the barrier region. It is only used in basic QW macros where the active macro contains parameters for both the barrier and the well

This statement is otherwise identical to **strain_well**.

22.674 strain_well

strain_well is an active layer macro statement used to define the strain in the quantum well. This command is only used for zincblende wells; wurtizte regions

compute the strain based on lattice values using different commands.

By convention, compressive strain is negative and the strain is given as a fraction (e.g., 3 percent strain is 0.03).

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.675 strained_mobility

parameter	data type	values [defaults]
carrier_type	char	[n] p
vsat_factor	real	[0.8]
strained_material	char	[yes],no
import_ref_population	char	[void]
export_ref_population	char	[void]
vsat_factor	real	[0.8]
unstrained_110_enh	real	[1.]
angle_from_100	real	[0.] (degree)

The statement **strained_mobility** is used to specify the strain dependent mobility for strained silicon. The basic idea is to evaluate a strain dependent mobility enhancement factor based on valley/subband population which is then used to scale the unstrained mobility.

The stained mobility is evaluated according to valley/subband averaged effective conduction masses in different directions, as well as valley splitting.

Parameters

- **carrier_type** is the carrier type of interest.
- **strained_material** indicates whether the structure is strained. This setting affects reference population data import/export as detailed below.
- **import_ref_population** is the data file containing reference population data for the unstrained material. It is imported to compute the strain-induced effective mass and mobility values when **strained_material**=*yes*.
- **export_ref_population** is the data file which is used to store unstrained population data for later use when **strained_material**=*no*.

- **vsat_factor** is the factor used for the saturation velocity. If this factor is unity, both low field and the saturation velocity are enhanced by the same factor.
- **unstrained_110_enh** is the enhancement of unstrained hole mobility in the 110 direction due to anisotropic effective hole mass.
- **angle_from_100** is the angle of crystal orientation with respect to 100.

Examples

```
strained_mobility carrier_type=p vsat_factor=0.5 &&
  import_ref_population=ref_pmos_population strained_material=yes &&
  unstrained_110_enh=0.93 angle_from_100=45
```

The above command imports quantum level reference data from a file placed in the installation directory and sets up the strained mobility model for angle=45 degrees from 100.

22.676 stretch_vertical_line

parameter	data type	values [defaults]
corner	char	[bottom_left], bottom_right, top_right, top_left
delta_x	real	[0.0] (μm)
delta_y	real	[0.0] (μm)

---

stretch_vertical_line is used in the .layer file to alter the position of a vertical line. This statement should be used immediately after a **layer_mater** in order to identify the vertical line which is to be altered.

The stretching algorithm is outlined in Fig. 22.30: point C is moved to C' and other segment lengths are modified to respect the following proportionality rule:

$$\frac{\overline{AB'}}{\overline{B'C'}} = \frac{\overline{AB}}{\overline{BC}}$$

The algorithm can further be understood using a simple children's toy as a reference; it consists of a white board on which vertical rubber bands are anchored in order to draw various shapes. By default, all points on the top/bottom of a vertical line

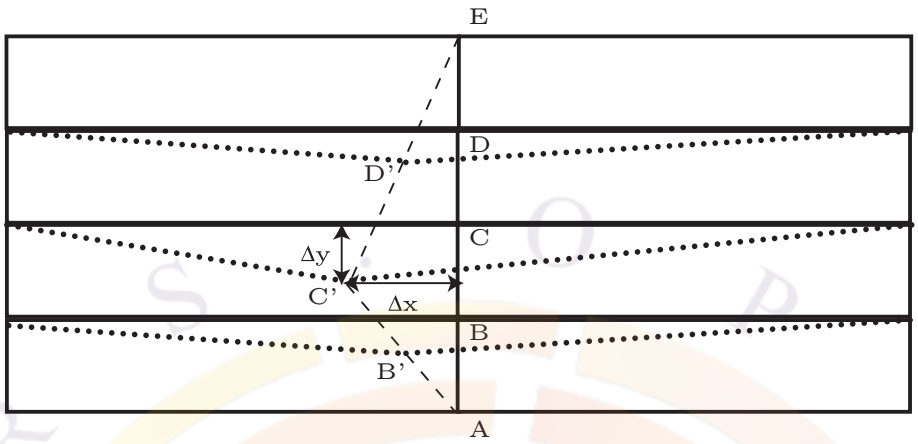


Figure 22.30: Illustration of vertical line stretching.

are anchored on the board, as are all reference points on vertical lines not being stretched (Fig. 22.31 a)). The final grid is formed by drawing horizontal lines on the picture to connect movable reference points: lack of parallelism is simply due to camera positioning and manual positioning of the reference points.

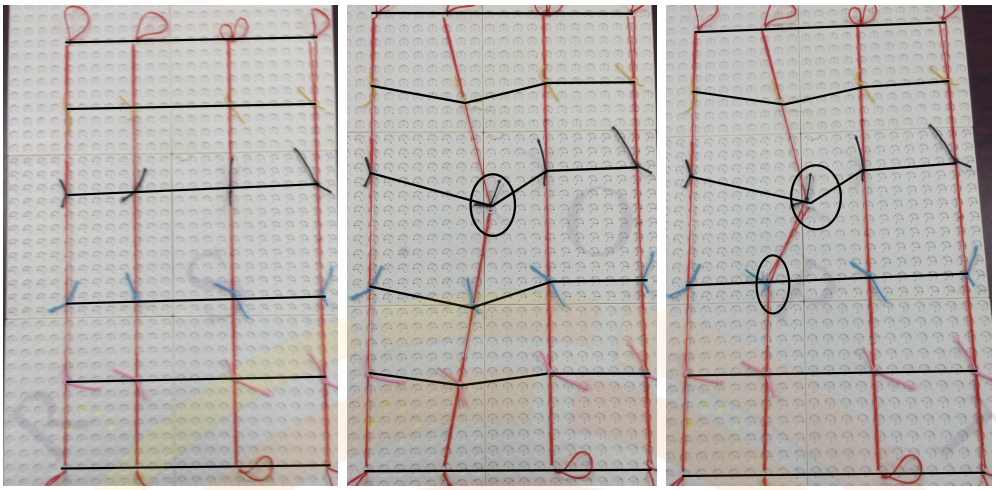
When a single point is moved (Fig. 22.31 b)), the affected vertical line is stretched all other points on this line move in a proportional manner. If a second point on the same line is forced to move with $\Delta = 0$ (Fig. 22.31 c)), it is put back to its original location which affects the stretching of other connected points and also puts them back to their original location. In this way, complex layer shapes may be drawn without resorting to the low-level .geo format: points may be moved in many different ways using different references on the same line or by combining movement on different lines.

This method also allows for layers of varying height rather than simply varying width as per the **upper_w1** parameter of the **layer** statement. In many cases, modifying the height instead of the width results in a better-quality mesh since the ratio of the distortion ($\frac{\Delta_y}{\Delta_x}$) is closer to unity.

Parameters

- **corner** defines the reference point used to pinch and stretch the vertical line. The corner is defined using the polygon of the preceding **layer_mater** statement.

- **delta_x** and **delta_y** are the offset applied to the reference point when stretching the vertical line.



(a) (b) (c)

Figure 22.31: Illustration of vertical line stretching using a child's toy: before stretching (a), with one moved point (b) and with a second point forcibly moved back to its original position (c).

22.677 stress_solution

parameter	data type	values [defaults]
initial_stress_model	char	[stress_set], wurtzite

stress_solution enables the use of a 3D mechanical stress solver; it is meant to be used alongside the **qdot_individual** statement.

Parameters

- **initial_stress_model** controls the initial stress on mesh boundaries. If =*stress_set* the boundary conditions are set by the **set_initial_stress** statement.

  As of v.2016, only =*stress_set* is a valid choice; other parameter values are still under development.

22.678 suprem_contact

parameter	data type	values [defaults]
side	char	[upper],lower,left,right
mater_label	char	
insul_semi_interf	char	[no],yes
xrange	realx2	(μm)
yrange	realx2	(μm)
num	intg	
touch_mater	intg	
zseg_num	intg	

suprem_contact is used to define electrodes on the mesh imported from CSUPREM, a process simulation program typically used to simulate the dopant profile.

Parameters

- **side** specifies the which side the contact is on.
- **num** is the contact number.
- **xrange** and **yrange** define the extent of the contact.
- **touch_mater** is used to help the software determine which material should be used to determine the boundary condition. This can be helpful if a contact touches multiple materials. If a label has previously been defined as an alias, **mater_label** may be used instead.
- **insul_semi_interf** instructs the software to look for a semiconductor/insulator interface within the specified range and define this interface as the contact boundary. This method overrides the shape of the contact region that would otherwise be produced by this command.
- **zseg_num** is the z-segment number where the contact is located.

Examples

```
suprem_contact xrange=(-0.05 0.05) num=2 side=upper touch_mater=1
```

The above defines a contact in the region $x \in [-0.05, 0.05]$ on top of the mesh and which touches material number 1.

22.679 suprem_impurity_define

parameter	data type	values [defaults]
name	char	void
deep_level_trap	char	[no],yes
charge_type	char	[donor], acceptor
level	real	[0.01] (eV)
elec_capture	real	[1.e-21] ($1/m^2$)
hole_capture	real	[1.e-21] ($1/m^2$)

suprem_impurity_define is used control the behavior of impurities loaded from a CSUPREM process simulation.

Parameters

- **name** is the impurity name in CSUPREM. Note that impurities activated manually or through a diffusion step in the process simulation usually have an "_active" suffix: plotting of the CSUPREM structure is recommended to double-check the impurity name.
- **deep_level_trap** controls whether the impurity acts as a shallow dopant or a deep level trap. In the latter case, the trap states have their own Fermi level rather than being in quasi-equilibrium with the carrier density.
- **charge_type** determines if the impurity acts as a donor or an acceptor.
- **level** is the energy position of the impurity.
- **elec_capture** and **elec_capture** are the capture cross-section coefficients used in the trap models.

Examples

```
suprem_impurity_define name=cu deep_level_trap=yes &&
charge_type=donor level=0.8 elec_capture=1.e-21 hole_capture=1.e-21
```

The above loads the raw (non-activated) copper (Cu) ion distribution from a customized CSUPREM implantation step into the device simulation. If a diffusion process step had also been used to activate the impurity, the name parameter should be changed to cu_active in order to load that diffused profile.

22.680 suprem_property

parameter	data type	values [defaults]
generic_impurity	char	[void],donor,acceptor
donor_type2	char	[void], arsenic ,phosphorus ,antimony ,selenium ,silcon ,germanium ,generic
acceptor_type2	char	[void] ,boron ,beryllium ,magnesium ,tin ,zinc ,carbon ,generic
n_doping_level	real	[0.01] (eV)
p_doping_level	real	[0.01] (eV)
n_doping_level2	real	[0.01] (eV)
p_doping_level2	real	[0.01] (eV)
qw_thick	real	[0.01] (um)
qw_xrange	realx2	[-0.05 0.05] (um)
silicon_mater	intg	
poly_mater	intg	
gaas_mater	intg	
aluminum_mater	intg	
oxide_mater	intg	
nitride_mater	intg	
oxynitride_mater	intg	
photoresist_mater	intg	
qw_silicon_mater	intg	

suprem_property is used to define properties related to data imported from the CSuprem, a process simulation program typically used to simulate the dopant profile. Please note that the coordinate system in APSYS is different than SUPREM in that the positive y coordinate points upwards in the former.

- **generic_impurity** defines the type of dopant assigned to generic impurity.
- **donor_type2** assigns a different type (type 2) of donor so that different ionization energy can be defined.
- **acceptor_type2** assigns a different type (type 2) of acceptor so that different ionization energy can be defined.
- **n_doping_level** is the ionization energy of the donor.
- **p_doping_level** is the ionization energy of the acceptor.

- **n_doping_level2** is the ionization energy of the donor type 2.
- **p_doping_level2** is the ionization energy of the acceptor type 2.
- **qw_thick** is the quantum well (qw, if any) thickness.
- **qw_xrange** is the x-range of he quantum well.
- **silicon_mater** defines the material number for silicon.
- **poly_mater** defines the material number for poly silicon.
- **gaas_mater** defines the material number for GaAs.
- **aluminum_mater** defines the material number for aluminum.
- **oxide_mater** defines the material number for SiO2.
- **nitride_mater** defines the material number for silicon notride.
- **oxynitride_mater** defines the material number for oxy-nidride.
- **photoresist_mater** defines the material number for photo-resist.
- **qw_silicon_mater** defines the material number for quantum-well silicon. This is actually the same material as silicon but since quantum well region receives special treatment, a different material number is assigned.
-

Example(s)

```
suprem_property silicon_mater=1 oxide_mater=3 qw_silicon_mater=2 &&
  poly_mater=4 &&
  generic_impurity=acceptor acceptor_type2=generic p_doping_level2=0.156 &&
  qw_xrange=(-0.05 0.05)
```

The above defines a QW from -0.05 to 0.05 microns and also defines acceptor type 2 as having ionization energy of 0.156 eV.

22.681 suprem_to_apsys_material

parameter	data type	values [defaults]
suprem_mater	intg	[1]
apsys_mater	intg	[1]

This command is used when importing simulation data from our process simulator CSUPREM for use in APSYS. It maps material numbers from one software to the other.

Parameters

- **suprem_mater** is the original material number in the CSUPREM file.
- **apsys_mater** is the new material number that will be used in APSYS.

Examples

```
suprem_to_apsys_material suprem_mater=3 apsys_mater=4
```

22.682 sym_polygon_for_semicrafter

parameter	data type	values [defaults]
similar_polygon_data	char	[void]
polygon_center_x	real	[1.0] (um)
polygon_center_y	real	[1.0] (um)
plane_size_x	real	[2.0] (um)
plane_size_y	real	[2.0] (um)
plane_start_x	real	[0.0] (um)
plane_start_y	real	[.0] (um)
polygon_edge	intg	[6]
mesh_per_edge	intg	[5]
plane_mesh_x	intg	[30]
plane_mesh_y	intg	[30]
uniform_below_layer	intg	[1]
outer_column	intg	[]

sym_polygon_for_semicrafter is a .layer pre-processing statement used to generate a set of CSUPREM mesh generation commands. It is meant to be used as quick alternative to the SemiCrafter GUI for simple shapes such as hexagonal nanowires.

When used, this statement changes the normal operation of the .layer file. Instead of defining the structure of an xy mesh plane, the layers now define the vertical stacking in the z direction with each layer corresponding to a z-segment. The top-view of the new structure (xy plane) is first given by a rectangular mesh plane. Etch statements from CSUPREM are then used to define a symmetric polygon centered somewhere on this mesh plane.

See also **layers_for_semicrafter** for a simplified version of this command which allows rectangular shapes in the xy plane.

Parameters

- **similar_polygon_data** is a text file containing information on the polygon. It can be used as an alternative to the other parameters below.
- **polygon_center_x** and **polygon_center_y** are the (x,y) center coordinates of the new polygon.
- **plane_start_x** and **plane_start_y** determine the position of the lower left corner of the plane containing the polygon in the xy plane.
- **plane_size_x** and **plane_size_y** determine the extent of the plane containing the polygon in the xy plane.
- **polygon_edge** is the number of edges of the symmetric polygon.
- **mesh_per_edge** determines the number of mesh points to assign to each edge during the CSUPREM etch process. This works in conjunction with **plane_mesh_x** and **plane_mesh_y** which define the regular mesh for the plane containing the polygon.
- **uniform_below_layer** specifies the depth of the etch used to create the symmetric polygon. Below this layer, the original mesh plane is not etched by CSUPREM and the rectangular plane (substrate) is preserved.
- **outer_column** is the right-most column in the substrate. All columns to the right of this one are not included in the simulation.

Examples

```
sym_polygon_for_semicrafter polygon_center_x=0.6 polygon_center_y=0.6 &&
     plane_size_x=1.2 plane_size_y=1.2 polygon_edge=6 mesh_per_edge=5 &&
     plane_mesh_x=30 plane_mesh_y=30
```

22.683 sym_polygon_quantum_well

parameter	data type	values [defaults]
polygon_center_x	real	[1.0] (um)
polygon_center_y	real	[1.0] (um)
polygon_edge	intg	[6]

sym_polygon_quantum_well is a .sol statement used to define a region with quantum confinement in the area between two symmetric polygons. Based on the input, the software will look to neighboring materials to identify the QW and barrier regions. The main application of this is nanowire LEDs.

See also **sym_polygon_for_semicrafter** which is used to define the initial structure in most problems where **sym_polygon_quantum_well** would be used.

Parameters

- **polygon_center_x** and **polygon_center_y** are the (x,y) coordinates of the center of the polygon.
- **polygon_edge** is the number of edges of the symmetric polygon.

Examples

```
sym_polygon_quantum_well polygon_center_x=0.6 polygon_center_y=0.6 &&
  polygon_edge=6
```

22.684 sym_polygon_taper

parameter	data type	values [defaults]
taper_info_file	char	[sym_polygon_taper_info.txt]

sym_polygon_taper imports a file which defines the taper profile between mesh planes. This statement is generated automatically by layer.exe when using commands such as **sym_polygon_for_semicrafter**. The taper profile is usually determined automatically by the polygon shapes in the original .layer structure.

22.685 symbol_variable

parameter	data type	values [defaults]
variation	char	[function] table
returned	char	
num	intg	[1]

symbolic_variable is used to define functional relationship between variables in the command system. The supported syntax and mathematical expressions are identical to those in the material macro library file. Please refer to the header information in the crosslight.mac file that comes with the software installation.

- **variation** decides if the relationship is defined by a function or a look up table. For the former, a math function of the form should follow

  function(var1,var2,...) end_function

  For a table the following form should be used:

  table(var1,var2,...) end_table

- **returned** is the returned symbol which may be used in any of the states following it.

- **num** is a number labeling the relation definition in case there are more than one such definitions.

Example(s)

```
symbolic_variable returned=%y
function(%x)
s2=%x**2+sin(%x);
s2**2+%x-5.
end_function
```

The above function is used to define a relation between symbolic variable %x and %y so that

```
symbolic_variable variation=table returned=%ai
table(%volt)
0.3  10.
0.8  18.
1.8  28.
end_talble
```

This statement uses a table to define the relation. Linear interpolation will be used is the argument does not fall on the tabulated values.

22.686 taper_between_segments

parameter	data type	values [defaults]
y1_from_label	char	
y2_from_label	char	
y1_to_label	char	
y2_to_label	char	
layer_data_file	char	
xpoint_from	real	(um)
top_xpoint_from	real	(um)
xpoint_to	real	(um)
top_xpoint_to	real	(um)
yrange_from	realx2	[-1.e9 1.e9] (um)
yrange_to	realx2	[-1.e9 1.e9] (um)
from_segment	intg	[1]
to_segment	intg	

taper_between_segments establishes a taper connection between the first and last plane of two neighboring z-segments. This statement replaces many parameters that were previously part of the **z_structure** statement. See also **taper_outer_boundary**.

Note that a common misconception about tapers is that they allow sampling of the region between segments. This is not so: actual mesh planes must be used to sample longitudinal variations. Tapers only control how the mesh points couple to each other between planes.

A more complete discussion on the 3D modeling scheme used by Crosslight software tools can be found in Sec. 6.3.

Parameters

- **y1_from_label** and **y2_from_label** are user labels defining the [y1,y2] range of the taper line at the beginning of the taper. The matching points of the taper line at the end of the taper are given by **y1_to_label** and **y2_to_label**.

  Instead of labels, absolute coordinates can also be specified using **yrange_from** and **yrange_to**.

- **xpoint_from** and **xpoint_to** are the x-coordinates of the taper line at the beginning and end of the taper, respectively. If the taper line is vertical, then only one value is required. Otherwise, **xpoint_from** and **xpoint_to** correspond to the minimum x value while **top_xpoint_from** and **top_xpoint_to** are the maximum x value of the taper line on each side of the taper.

- **from_segment** is the z-segment number immediately before the taper. The next segment (after the taper) is usually determined automatically but can also be specified using **to_segment**.

- **layer_data_file** is the name of a file describing variations in layer thicknesses along the taper. This file is usually generated automatically by the process simulator CSUPREM.

- **layer** is a layer number used to help the software connect taper lines to the right layer if there is a thickness variation along the taper.

22.687 taper_outer_boundary

parameter	data type	values [defaults]
left_taper	char	[no]
right_taper	char	[no]
bottom_taper	char	[no]
top_taper	char	[no]
from_segment	intg	[1]

---

taper_outer boundary essentially serves the same purpose as **taper_between_segments**. but instead of using point coordinates, taper lines generated by this command are based on the outer boundaries of the mesh planes on each side of the taper.

Parameters

- **left_taper**, **right_taper**, **top_taper** and **bottom_taper** uses a straight edge on the respective edge of the device as a taper line.
- **from_segment** is the z-segment number immediately before the taper.

22.688 tau_density

parameter	data type	values [defaults]
hh_or_lh	char	[hh] lh ch
gamma_detail	char	[no] yes
data_file	char	[void]
density_start	real	[2.e23] (m^{-3})
density_end	real	[2.e24] (m^{-3})
pn_ratio	real	[1.]
data_point	intg	[10]
elec_level	intg	[1]
hole_level	intg	[1]

tau_density plots the intraband relaxation time (ie., **tau_scat**) in 10^{-13} sec. versus carrier density. This statement is only used in the gain preview module.

- **hh_or_lh** indicates whether heavy hole or light hole subband is involved.
- **gamma_detail** may be used to plot individual contribution to gain broadening factor (Γ) from different scattering mechanisms such as electron-electron, electron-hole, etc..
- **data_file** is the file to which the graphic data is written in ASCII format.
- **density_start** is the starting point of the carrier density at which the intraband relaxation time is calculated.
- **density_end** is the ending point of the carrier density at which the intraband relaxation time is calculated.
- **pn_ratio** is the ratio of hole over electron concentrations. Note that this ratio can be set to an arbitrary number in the gain preview. In the main solver, this ratio is determined automatically by the simulator according to the local Fermi levels.

- **data_point** is the number of data points in data curve.
- **elec_level** and **hole_level** are the electron and hole subband levels involved in the optical transition.

Example(s)

```
tau_density gamma_detail=yes density_end=5.e24
```

22.689 tau_energy

parameter	data type	values [defaults]
(see) material_par		

tau_energy is the energy relaxation time of hot electrons in seconds. It is used only when hot electron model is activated.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.690 tau_model

parameter	data type	values [defaults]
char	char	
phonon_omega_LO	real	[34.E-3] (eV)
epsilon_infinity	real	[11.4]
mater	intg	1

tau_model is used to turn on the intraband scattering model discussed in Sec. 10.2. When this model is activated, the constant $\frac{\hbar}{\tau}$ normally defined in **active_reg** is overridden and replaced with a temperature-dependent calculated value.

This statement may be used in both the gain preview file (.gain) and the main input file (.sol).

Parameters

- **phonon_omega_LO** is the longitudinal optical phonon energy in eV.
- **epsilon_infinity** is the optical (or at infinite frequency) dielectric constant.
- **mater** is the material number of the active region under consideration. If a label has previously been defined, **mater_label** may be used instead.

Examples

```
tau_model mater=2 phonon_omega_LO=46.E-3 epsilon_infinity=10.4
```

22.691 tau_temperature

parameter	data type	values [defaults]
hh_or_lh	char	[hh] lh ch
gamma_detail	char	[no] yes
data_file	char	[void]
temp_start	real	[200.]
temp_end	real	[400.]
density	real	[2.e24] (m^{-3})
pn_ratio	real	[1.]
data_point	intg	[10]
elec_level	intg	[1]
hole_level	intg	[1]

tau_temperature plots the intraband relaxation time (ie., **tau_scat**) in 10^{-13} sec. versus temperature. This statement is only used in the gain preview module.

- **hh_or_lh** indicates whether heavy hole or light hole subband is involved.
- **gamma_detail** may be used to plot individual contribution to gain broadening factor (Γ) from different scattering mechanisms such as electron-electron, electron-hole, etc..
- **data_file** is the file to which the graphic data is written in ASCII format.
- **temp_start** is the starting point of temperature range.
- **temp_end** is the ending point of temperature range.

- **density** is the carrier density at which the intraband relaxation time is calculated.
- **pn__ratio** is the ratio of hole over electron concentrations. Note that this ratio can be set to an arbitrary number in the gain preview. In the main solver, this ratio is determined automatically by the simulator according to the local Fermi levels.
- **data__point** is the number of data points in data curve.
- **elec__level** and **hole__level** are the electron and hole subband levels involved in the optical transition.

Example(s)

```
tau_temperature temp_start=200.  temp_end=400.  density=2.e24  &&
  data_point=10 elec_level=1 hole_level=1 gamma_detail=yes
```

22.692 tax__mass__bar

tax__mass__bar is an active layer macro statement used to define the anisotropic effective mass of the conduction band for silicon quantum MOS. The band structure model has a ellipsoidal shaped constant-energy surface of cylindrical symmetry around a symmetry axis. For example if the ellipsoid is symmetric around the z-axis, the energy dispersion becomes:

$$E(k_x, k_y, k_z) = E_0 + \frac{\hbar^2}{2m_0}\left(\frac{k_z^2}{\text{lax\_mass}} + \frac{k_x^2}{\text{tax\_mass}} + \frac{k_y^2}{\text{tax\_mass}}\right) \qquad (22.127)$$

tax__mass__barr thus defines the transverse mass perpendicular to the symmetry axis in the quantum barrier region.

The parameters for this statement are the same as for all other material statements. See **material__par** in section 22.456 for examples and further details.

22.693 tax__mass__well

tax__mass__well is identical to **tax__mass__bar** but defines the transverse mass in the well region rather than in the barrier.

The parameters for this statement are the same as for all other material statements. See **material__par** in section 22.456 for examples and further details.

22.694 temp_dep_macro_table

parameter	data type	values [defaults]
doping_logscale	char	[yes]
disable_table_method	char	[no]
temper_points	intg	[60]
doping_points	intg	[40]
wavelength_points	intg	[60]
other_points	intg	[40]

This statement controls the behavior of temperature-dependent macro parameters in the software. At the beginning of the simulation, macros are evaluated at fixed temperature values and a multi-dimensional table is constructed. Later, material parameters are obtained as needed by interpolation over this table.

The temperature range of the table is governed by the parameters of the **heat_flow** statement.

Parameters

- **doping_logscale** controls whether the doping points are spaced on a linear or logarithmic scale.
- **disable_table_method** disables the temperature interpolation table and reverts to an older method based on a polynomial fit.
- **temper_points** is the number of temperature points in the interpolation table.
- **doping_points** is the number of doping points in the interpolation table. It used as an extra dimension for the interpolation (.e.g graded doping region).
- **wavelength_points** is the number of wavelength points in the interpolation table. It used as an extra dimension for the interpolation (e.g. spectrum-dependent quantities).
- **other_points** is the number of points used for other graded properties (e.g. composition). Again, this is an extra dimension used for interpolation.

Examples

```
temp_dep_macro_table temper_points=100
```

22.695 temperature

parameter	data type	values [defaults]
temp	real	[300] (K)

temperature sets the isothermal temperature in the device.

Parameters

- **temp** is the temperature in K.

Examples

```
temperature temp=350
```

22.696 thermal_interf

parameter	data type	values [defaults]
x_label	char	[void]
y_label	char	[void]
within_x1_label	char	[void]
within_x2_label	char	[void]
within_y1_label	char	[void]
within_y2_label	char	[void]
thm_num	intg	
thm_type	intg	[1]
thm_with_x	realx2	[-1.1e19 1.1e19](um)
thm_with_y	realx2	[-1.1e19 1.1e19](um)
thm_x	real	(um)
thm_y	real	(um)
thm_lat_temp	real	[300.] (Kelvin)
thm_ht_flow	real	[100.] (2D: W/m, 3D:W)
thm_cond	real	[10.] (2D: (W/K)/m, 3D: W/K)
thm_ext_temp	real	[300.] (K)

The statement **thermal_interf** defines the thermal properties of a non-electrode thermal boundary.

Parameters

The parameters of this command relating to the positioning of the thermal interface are the same as those in the **interface** statement. The parameters related to the thermal properties are the same as those of the **contact** statement, with some minor syntax variations. For the sake of brevity, these parameters will be omitted here.

- **thm_num** is the thermal interface number. This number is distinct from the contact numbers even though contacts also serve as thermal boundaries.

Examples

```
Example(s)

thermal_interf thm_num = 1 thm_type = 1 thm_lat_temp = 77. &&
thm_with_x = (2. 2.) thm_with_y = (2.2 3.)

thermal_interf thm_num = 2 thm_type = 2 thm_ht_flow = 0. &&
thm_y = 3.
```

22.697 thermal_kappa

thermal_kappa defines the thermal conductivity κ (in W/(m*K)) for a given material. This is used to solve the heat flow equation and is otherwise ignored.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.698 thermal_kappa_xy

parameter	data type	values [defaults]
dir	char	[y],x
mater_label	char	
factor	real	[1.]
mater	intg	[1]

This statement modifies the thermal conductivity defined by **thermal_kappa** to create an anisotropic κ on the x-y plane. This may be useful for the effective medium approximation of multiple layers (such as DBR mirrors in a VCSEL).

Parameters

- **dir** is the direction in which the κ is scaled.
- **factor** is the scaling factor applied to κ.
- **mater** is the number of the material affected by this statement. If a label has previously been defined, **mater_label** may be used instead.

Examples

```
thermal\_kappa_xy dir=y factor=0.6 mater=3
```

The above statement decreases κ in the y-direction by a factor of 0.6 in material number 3.

22.699 tmass_gamma_bar

parameter	data type	values [defaults]
(see) material_par		

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

It is the barrier conduction band effective mass in direction perpendicular to the c-axis. This statement is applicable for wurtzite structure active region [1]. The mass in c-axis direction is specified by **mass_gamma_bar**.

[1] S. L. Chuang, “Optical Gain of Strained Wurtzite GaN Quantum Well Lasers", IEEE J. Quantum Electron., VOL. 32, NO. 10, OCTOBER 1996, p. 1791

22.700 tmass_gamma_bulk

parameter	data type	values [defaults]
(see) material_par		

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

It is the conduction bane effective mass in direction perpendicular to the c-axis. This statement is applicable for wurtzite structure bulk region [1].

[1] S. L. Chuang, “Optical Gain of Strained Wurtzite GaN Quantum Well Lasers", IEEE J. Quantum Electron., VOL. 32, NO. 10, OCTOBER 1996, p. 1791

22.701 tmass_gamma_well

parameter	data type	values [defaults]
(see) material_par		

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

It is the well conduction band effective mass in direction perpendicular to the c-axis. This statement is applicable for wurtzite structure active region [1]. The mass in c-axis direction is specified by **mass_gamma_well**.

[1] S. L. Chuang, “Optical Gain of Strained Wurtzite GaN Quantum Well Lasers", IEEE J. Quantum Electron., VOL. 32, NO. 10, OCTOBER 1996, p. 1791

22.702 top_contact

parameter	data type	values [defaults]
contact_type	char	[ohmic] schottky
use_schottky_xp	char	[yes] no
column_num	intg	[1]
from	real	
to	real	
contact_num	intg	
barrier	real	(volt)
work_function	real	(volt)
intern_xp_size	real	[0.0002] (um)

top_contact defines the placement of the top contact in the .layer file. When the .layer file is processed, it defines boundary conditions for polygons in the .geo file and creates a matching **contact** statement in the .mater file.

Parameters

- **contact_type** indicates the type of contact and can take either *ohmic* or *schottky*.
- **use_schottky_xp** is used to control the use of extra point near the contact in case the contact is of Schottky type.
- **column_num** describes the column where this contact is found.
- **to** and **from** set the x-range where the contact lies. Note that the origin is at the upper left-hand corner of each polygon and that units are in μm.
- **contact_num** defines the index number of the contact.
- **barrier** is the barrier height of the Schottky contact.
- **work_function** is the work function of the Schottky contact.
- **intern_xp_size** is the internal extra point size around a Schottky contact.

Examples

```
top_contact column_num=3 from=0 to=1.8 contact_num=1
```

22.703 trap_assisted_tunnel_junc

parameter	data type	values [defaults]
level_from_cond	real	[0.6] (eV)
tau_n	real	[1.e-2] (s)
tau_p	real	[1.e-2] (s)
pf_expo_factor	real	[1.]
set_tj_num	intg	

trap_assisted_tunnel_junc defines a trap-assisted (interband) tunneling junction model[1]. This command must be used in conjunction with **tunnel_junc** with **use_physical_model**=*yes*.

For trap-assisted intraband tunneling, please see **trap_assisted_tunneling**.

Parameters

- **level_from_cond** is the energy level of the trap, measured from the conduction band.
- **tau_n** and **tau_p** are the trap carrier lifetimes.
- **pf_expo_factor** is the Poole-Frenkel shift of the trap level due to the local field. The same formula used for the incomplete ionization of traps (Sec. 5.1.4) is used here.
- **set_tj_num**, when specified, limits the trap settings to a specific tunnel junction (numbered according to the order of the **tunnel_junc** statements). If this parameter is omitted, all tunnel junctions share the same settings.

Examples

```
tunnel_junc use_physical_model=yes yrange=(0.9 1.1)
trap_assisted_tunnel_junc level_from_cond=0.8 tau_n=1.e-2 tau_p=1.e-2
```

22.704 trap_assisted_tunneling

parameter	data type	values [defaults]
direction	char	x,[y],z
carrier	char	[electron], hole
x1_label	char	[void]
y1_label	char	[void]
x2_label	char	[void]
y2_label	char	[void]
zplane_from_label	char	
zplane_to_label	char	
zdir_x1_label	char	
zdir_x2_label	char	
zdir_y1_label	char	
zdir_y2_label	char	
model	char	[linear] , poole_frenkel, hopping
trap_profile	char	
point_ll	realx2	[μm]
point_ur	realx2	[μm]
scale_tunnel_coef	real	[1.]
zdir_xrange	realx2	[μm]
zdir_yrange	realx2	[μm]
trap_density	real	[1.e20] (m^{-3})
linear_rate1	real	[1.e8] (m/(V s))
tau0_pf	real	[1.e7] (s)
trap_size_hopping	real	[1.e-3] (μm)
trap_level_depth	real	[1.2] (eV)
trap_profile_start	real	[0.] (μm)
tau0_hopping	real	[1.e-9] (s)
trap_level_ref	real	[1.2] (eV)
zplane_from	intg	[1]
zplane_to	intg	[5]
exclude_materi(i=1..5)	intg	
fly_to_contacts	intgx2	
zseg_num	intg	

trap_assisted_tunneling defines a trap-assisted intraband tunneling model. Prior to the 2015 version, this command referred to an interband tunnel junction model which can now be accessed using **trap_assisted_tunnel_junc**.

Theory

The trap-assisted tunneling (TAT) model is based on the assumption that traps in a barrier (insulator or wide bandgap material) are able to emit carriers with a rate of $\frac{1}{\tau}$ and thus generate a current flux with the following form[143]:

$$J = q \int_0^{t_{bar}} \frac{N_{trap}}{\tau} dx \tag{22.128}$$

where the integral is along the thickness of the barrier layer and N_{trap} is the bulk trap density.

To convert the above into the familiar field-dependent mobility form, the rate must be written in a field-dependent form:

$$\frac{1}{\tau} = S_{tat} f_T \tag{22.129}$$

where S_{tat} is the emission rate, F is the electrical field and f_T is a correction factor due to temperature, related to thermal activation of trapped carriers at energy level E_t:

$$f_T = A e^{-\frac{E_t}{k_B T}} \tag{22.130}$$

$$A = e^{\frac{E_{t0}}{k_B \times 300}} \tag{22.131}$$

Using a first-order approximation of the above rate, the mobility form can be obtained:

$$S_{tat} f_T = S_{tat} f_T \mid_{F=0} + \frac{d}{dF} \left(S_{tat} f_T \right) F \tag{22.132}$$

and by imposing the boundary condition of zero net current in the absence of applied bias, the current flux can be obtained as:

$$J = q N_{trap} \frac{d}{dF} \left(S_{tat} f_T \right) \Delta V \tag{22.133}$$

It is interesting to note that if the emission rate is linear in field, the current flux is only dependent on applied voltage and not on the field or the thickness of the barrier.

Various models of field dependence are implemented:

Linear model

The field dependence comes from a linear $\frac{dS_{tat}}{dF}$ term. This model is easy to converge since there is no field-dependent mobility. Temperature dependence still included via the f_T term which has no field dependence.

Poole-Frenkel model

The field dependence is introduced via f_T and comes from a shift in the trap level's position, assuming the presence of a Coulomb potential[143]:

$$\Delta E_t = \sqrt{\frac{qF}{\pi \epsilon_0 \epsilon}} \tag{22.134}$$

This model has a strong field dependence and current saturates strongly.

Hopping model

The field dependence is introduced via f_T and comes from a shift in the trap level's position, assuming the presence of a rectangular potential well of size d_{hop}:

$$\Delta E_t = F d_{hop} \tag{22.135}$$

Parameters

- **direction** is the direction of the tunneling current.
- **carrier** is the type of carrier concerned.
- **point_ll**, **point_ur** are upper left and lower right corner points of the area being considered for tunneling. The area must be large enough to cover the potential barrier and must be at least a few times the mean free path of the tunneling carrier. However, it should not be so large as to affect carrier transport in areas unrelated to quantum tunneling.
- **x1_label**,**y1_label**,**x2_label**,**y2_label** are position labels equivalent to **point_ll** and **point_ur**.
- **zplane_from zplane_to** allow the tunneling to take place between two z-mesh planes. Instead of plane numbers, labels may be used in **zplane_from_label** and **zplane_to_label**.

Optionally, the lateral range of the tunneling between these two planes may be restricted using the absolute coordinates of **zdir_xrange** and **zdir_yrange** or the position labels in **zdir_x1_label**, **zdir_x2_label**,**zdir_y1_label** and **zdir_y2_label**.

- **model** switches between the available trap-assisted tunneling models.
- **trap_density** is the trap density (N_{trap}).

  Spatial dependence of the trap profile can be included via a text file defined in **trap_profile**; this spatial profile is relative to the position given in **trap_profile_start**.
- **linear_rate1** defines the value of $\frac{dS_{tat}}{dF}$.
- **tau0_pf** and **tau0_hopping** define the value of τ at 300K in the Poole-Frenkel and hopping models, respectively. They serve to define the constant value of S_{tat} in the 1st-order expansion of the field dependence.
- **trap_size_hopping** is the value of d_{hop}.
- **trap_level_depth** defines the trap position E_t.
- **trap_level_ref** defines the trap reference E_{t0}.
- **exclude_materi(i=1..5)** may be used to tell the software to omit quantum tunneling from certain materials in the simulation.
- **fly_to_contacts**, when used, takes the tunneling current and applies it directly to the electrode current of the specified pair, bypassing the local mesh. This may be used to represent hot carriers flying over a depleted area where thermalized carriers would be unable to support the needed current flow.
- **zseg_num** is the z-segment number in which the tunneling takes place (if on the x-y plane).

22.705 trap_conc_1

trap_conc_i,i=1..9 is a passive macro statement used when traps are defined using the **material** statement rather than **doping**. It defines the concentration (in m^{-3}) of traps with the label "i".

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.706 trap_conc_2

See trap_conc_1.

22.707 trap_conc_3

See trap_conc_1.

22.708 trap_conc_4

See trap_conc_1.

22.709 trap_conc_5

See trap_conc_1.

22.710 trap_conc_6

See trap_conc_1.

22.711 trap_conc_7

See trap_conc_1.

22.712 trap_conc_8

See trap_conc_1.

22.713 trap_conc_9

See trap_conc_1.

22.714 trap_excitation

parameter	data type	values [defaults]
emit_carrier	char	[electron] hole
mater_label	char	
cross_section	real	[1.e-19] (m^2)
power_depend_coef	real	[0] ($m^4/Watt$)
trap_index	intg	[1]
mater	intg	[1]

The statement **trap_excitation** is used to specify the photo properties of a deep level trap. The theory is detailed in Sec. 5.5.

22.714.1 Parameters

- **emit_carrier** specifies the type of carriers emitted upon light excitation.
- **cross_section** is the cross section of light absorption. When it is multiplied by the density of traps responding to light, an equivalent absorption coefficient may be obtained.

  power_depend_coef is the power dependent cross section coefficient. When multiplied by the optical power density in units of $Watt/m^2$, it adds to the light absorption cross section of the trap.
- **trap_index** is the index (or label) of the deep trap.
- **mater** is the index of the material being affected. If a label has previously been defined for this material, **mater_label** may be used instead.

22.714.2 Examples

```
trap_excitation emit_carrier=electron  &&
  cross_section=1.e-19 trap_index=1 mater=2
```

22.715 trap_level_i

trap_level_i is used to define or override the energy level of deep traps with the label "i" (i=1..9). This level is measured from the conduction band and in the unit of eV. See Sec. 5.5 for more details.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.716 trap_ncap_i

The material statements **trap_ncap_i** and **trap_pcap_i** are the electron and hole capture cross section in m^2. i=1..9 is a label identifying the trap species affected by this statement. the trap_i (i=1,2,...).

The relationship between capture cross sections and minority carrier lifetime is explained under the statements **lifetime_n** and **lifetime_p**. See also Sec. 5.5 for more details.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.717 trap_pcap_i

See **trap_ncap_i**.

22.718 trap_type_1

parameter	data type	values [defaults]
type	char	[donor]
mater_label	char	
mater	intg	[1]

trap_type_i,i=1..9 is a passive macro statement used when traps are defined using the **material** statement rather than **doping**. It defines the trap type (donor or acceptor) for trap species #i.

The material number is only necessary when used to override macro parameters in the .sol file. In that case, **mater_label** may be used instead if a label has previously been defined for the material.

22.719 trap_type_2

See **trap_type_1**.

22.720 trap_type_3

See **trap_type_1**.

22.721 trap_type_4

See **trap_type_1**.

22.722 trap_type_5

See **trap_type_1**.

22.723 trap_type_6

See **trap_type_1**.

22.724 trap_type_7

See **trap_type_1**.

22.725 trap_type_8

See **trap_type_1**.

22.726 trap_type_9

See **trap_type_1**.

22.727 traplevel_stddev_1

traplevel_stddev_i,i=1..9 is a passive macro statement used when traps are defined using the **material** statement rather than **doping**. It defines the standard

deviation when the trap levels follow a gaussian model. This statement applies to traps with the label “i”.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further destddevs.

22.728 traplevel_stddev_2

See **traplevel_stddev_1**.

22.729 traplevel_stddev_3

See **traplevel_stddev_1**.

22.730 traplevel_stddev_4

See **traplevel_stddev_1**.

22.731 traplevel_stddev_5

See **traplevel_stddev_1**.

22.732 traplevel_stddev_6

See **traplevel_stddev_1**.

22.733 traplevel_stddev_7

See **traplevel_stddev_1**.

22.734 traplevel_stddev_8

See **traplevel_stddev_1**.

22.735 traplevel_stddev_9

See **traplevel_stddev_1**.

22.736 traplevel_tail_1

traplevel_tail_i,i=1..9 is a passive macro statement used when traps are defined using the **material** statement rather than **doping**. It defines the characteristic decay constant (L in $e^{-E/L}$) when the trap levels follow an exponential decay model. This statement applies to traps with the label "i".

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.737 traplevel_tail_2

See **traplevel_tail_1**.

22.738 traplevel_tail_3

See **traplevel_tail_1**.

22.739 traplevel_tail_4

See **traplevel_tail_1**.

22.740 traplevel_tail_5

See **traplevel_tail_1**.

22.741 traplevel_tail_6

See **traplevel_tail_1**.

22.742 traplevel_tail_7

See **traplevel_tail_1**.

22.743 traplevel_tail_8

See **traplevel_tail_1**.

22.744 traplevel_tail_9

See **traplevel_tail_1**.

22.745 tunneling

parameter	data type	values [defaults]
direction	char	x,[y],z
carrier	char	[electron], hole
barrier_type	char	[propagation_matrix]
trans_broad	char	[no], rectangle, lorentzian
x1_label	char	[void]
y1_label	char	[void]
x2_label	char	[void]
y2_label	char	[void]
local_correction	char	[yes],no
miniband_model	char	yes,[no]
miniband_period_y1_label	char	
miniband_period_y2_label	char	
save_miniband	char	[miniband_data.txt]
adjust_emission_point	char	[yes]
zplane_from_label	char	
zplane_to_label	char	
zdir_x1_label	char	
zdir_x2_label	char	
zdir_y1_label	char	
zdir_y2_label	char	
point_ll	realx2	[μm]
point_ur	realx2	[μm]

broad_width	real	[1.] (k_BT)
scale_tunnel_coef	real	[1.]
adjust_barrier	real	[0.] (eV)
miniband_upper_range	real	[0.5](eV)
upper_range_above_peak	real	[0.2] (eV)
zdir_xrange	realx2	[μm]
zdir_yrange	realx2	[μm]
miniband_period_y1	real	[μm]
miniband_period_y2	real	[μm]
mesh_multiply	intg	[2]
energy_points	intg	[3000]
miniband_number	intg	[2]
miniband_fd_mesh	intg	[100]
miniband_k_points	intg	[20]
zplane_from	intg	[1]
zplane_to	intg	[5]
exclude_materi(i=1..5)	intg	
fly_to_contacts	intgx2	

tunneling enables the intraband quantum tunneling transport mechanism. This covers a variety of models including the standard theories of Sec. 9.2 and miniband transport[144, 145]. As of the 2013 version, this command is also used to enable non-equilibrium Green's function (NEGF) transport in conjunction with the **negf_model** command.

Parameters

- **direction** is the direction of the tunneling current.
- **carrier** is the type of carrier concerned.
- **barrier_type** is the type of barrier being tunneled through. Several models are available depending on the shape of the potential profile:
  - "rectangle" (default): barrier profile should have steep edges on both sides.
  - "triangle": one side of the barrier should be much steeper than the other.
  - "smooth_wkb": smooth barrier profile

  - "propagation_matrix": general-purpose model, more accurate but slower than the others.

- **local_correction** is used to indicate whether a mesh-dependent quantum tunneling correction is applied to all local mesh points. If set to no, the tunneling current is calculated at the top of the barrier and is assumed to be uniform across the whole rectangle region. The local mesh correction method is recommended for relatively smooth potential profiles while non-local correction works better for thin and high tunneling barriers.

- **trans_broad** is used to indicate if and what type of energy level broadening is used when using the "propagation_matrix" method. The profile of the broadened level may in take rectangle or Lorentzian shape.

- **point_ll**, **point_ur** are upper left and lower right corner points of the area being considered for tunneling. The area must be large enough to cover the potential barrier and must be at least a few times the mean free path of the tunneling carrier. However, it should not be so large as to affect carrier transport in areas unrelated to quantum tunneling.

- **x1_label**,**y1_label**,**x2_label**,**y2_label** are position labels equivalent to **point_ll** and **point_ur**.

- **adjust_emission_point** triggers a downhill search for the bottom of the potential profile in order to set the tunneling energy range; if disabled, the energy range is set using the start/end points of the tunneling region. It is recommended to disable this setting if dealing with a potential profile consisting mostly of quantum wells.

  This parameter is new as of the 2013 version and replaces the use of manual "barrier" labels.

- **broad_width** is the broadening of tunneling energy level in units of k_bT, where k_b is the Boltzmann constant and T is the temperature.

- **scale_tunnel_coef** provides a way to artificially scale the tunneling coefficient.

- **adjust_barrier** works when **local_correction=no** and is used to adjust the barrier height used to compute the carrier density at the top of the barrier. Please note that this parameter only affects the carrier concentration at the top of the barrier (thus the magnitude of tunneling current) but does not affect the potential profile used to calculate the tunneling transparency.

- **mesh_multiply** is a multiplication factor used to over-sample the electrical mesh along the path of the tunneling current. It is used in the propagation matrix method to get more accurate results.

- **energy_points** is the number of energy divisions used in evaluation of tunneling current when using the propagation matrix method.
- **fly_to_contacts**, when used, takes the tunneling current and applies it directly to the electrode current of the specified pair, bypassing the local mesh. This may be used to represent hot carriers flying over a depleted area where thermalized carriers would be unable to support the needed current flow.
- **miniband_model** would enable the mini-band transport model[144, 145] through the tunneling region.
- **miniband_period_y1_label** and **miniband_period_y2_label** are position labels equivalent to **miniband_period_y1** and **miniband_period_y2**.
- **save_miniband** saves the miniband data to a file.
- **miniband_period_y1** and **miniband_period_y2** are positions used to define the beginning and ending of a period for the miniband model.
- **miniband_upper_range** defines an upper range within which we search the energy solution of the miniband.
- **miniband_number** is the number of minibands to be solved and used in optical/transport modeling.
- **miniband_fd_mesh** is the finite difference mesh number within a period of the miniband.
- **miniband_k_points** is the number of k-points to be solved for the miniband structure.
- **upper_range_above_peak** extends the tunneling energy range some distance above the peak of the potential profile. Note that the default non-zero value automatically double-counts some carriers since thermionic emission can occur within this range; use a zero value to go back to the pre-2013 behavior.
- **zplane_from zplane_to** allow the tunneling to take place between two z-mesh planes. Instead of plane numbers, labels may be used in **zplane_from_label** and **zplane_to_label**.

  Optionally, the lateral range of the tunneling between these two planes may be restricted using the absolute coordinates of **zdir_xrange** and **zdir_yrange** or the position labels in **zdir_x1_label**, **zdir_x2_label**,**zdir_y1_label** and **zdir_y2_label**.
- **exclude_materi(i=1..5)** may be used to tell the software to omit quantum tunneling from certain materials in the simulation.

Position Labels

In this statement, various absolute coordinates defining boundaries can be replaced by equivalent position labels for the convenience of the user. These labels must be predefined using the **x_position**, **y_position** or **define_vertical_position** commands.

It is also possible to use higher-level commands such as **layer_position** and **column_position** in the .layer file. These will automatically generate the correct position labels when the file is processed so the user never needs to figure out the exact coordinate of a position label. Also, any changes in the .layer file will automatically result in matching changes to the position labels.

Examples

```
tunneling point_ll=(0., 2.49) point_ur=(0.5 2.6) &&
  barrier_type=triangle
tunneling point_ll=(0., 4.7) point_ur=(0.5 4.95) &&
  barrier_type=triangle carrier=hole
```

22.746 tunnel_junc

parameter	data type	values [defaults]
x_start_label	char	
x_end_label	char	
y_start_label	char	
y_end_label	char	
set_n_side	char	[void],up,down,left,right
use_physical_model	char	[no]
region_label	char	
direction	char	[xy],z
zplane_start_label	char	
zplane_end_label	char	
type2_quantum_well	char	[no]
ignore_forward_current	char	[no]
physical_model	char	[zener], silicon_pat, kane_direct, kane_indirect
xrange	realx2	[-1.e9 1.e9] (um)
yrange	realx2	[-1.e9 1.e9] (um)
equiv_mobility	real	[0.01] (m^2/V/s)
tun_mass	real	
type2_qw_mob_scale	real	[1.0]
silicon_holemass_fac	real	[1.]
scale_tj_current	real	[1.]
zseg_num	intg	[1]
silicon_dir	intg	[001], 011 111

tunnel_junc is used to define an interband tunneling region. Unlike the **zener** model which converts the tunneling current into a local generation term, this model is a non-local approach and the tunneling current goes out from one mesh point and into a remote point. This model therefore bypasses the normal "fluid-like" approach of Drift-Diffusion theory.

The default model for the tunneling current simply assumes some equivalent mobility coefficient which is used to exchange electron and holes on either side of the junction. This method is very numerically stable but may miss some important features of the device such as the peak current and negative resistance of the tunnel junction.

More accurate models (**use_physical_model** and **physical_model**) may also be used to compute the equivalent mobility. This is most often used in multi-junction devices such as solar cells where the individual tunneling junctions are forward-biased. However, this physical model may have trouble in reverse-biased junctions

and calculate a zero mobility which would completely block the current across the tunnel junction. In such a case, the default equivalent mobility model may be used to restore the current flow.

Note that since the tunneling is defined between two mesh points only, it is very important to define the tunneling range properly so that it is representative of the whole junction. Ideally, the end points will be some distance away from the junction itself where the carrier density is high but the band profile is relatively flat. However, the points should also not be too far apart since tunneling is a short-range process: inverting some exponentials can cause NaN (not a number) or divide by zero errors if the wave function decays too much. Selecting points that do not meet this criteria may result in an inaccurate evaluation of the overall tunneling current in the device.

Parameters

- **x_start_label**, **x_end_label**, **y_start_label** and **y_end_label** are position labels which define the tunneling range in the xy plane. When not used, the software will default to the absolute coordinates defined in **xrange** and **yrange**.

  For tunneling in the z direction, **zplane_start_label** and **zplane_end_label** should be used.

- **set_n_side** is used to select the direction of the electron-to-hole tunneling. If void, the software will examine the doping profile to determine the most likely choice. With a properly defined tunneling range and doping profile, the default setting is usually sufficient. However, this may need to be set to define the position of the electron well in type II QW structures.

- **use_physical_model** tells the software to use one of the physical models to estimate the equivalent tunneling mobility. The exact model used is set by **physical_model** and can take one of the following values:

  - *zener* defines the Zener tunneling model described in Sec. 9.3.
  - *silicon_pat* defines a phonon-assisted tunneling model for silicon derived by Schenk[146, 147].
  - *kane_direct* is similar to the Zener model above but uses a slightly different field dependence model[148].
  - *kane_indirect* is also similar to the Zener model above and uses yet another slightly different field dependence model[148].

  Trap-assisted tunneling may also be defined using **trap_assisted_tunneling**. This will supplement the tunneling current calculated by the above physical models.

- **region_label** is a user-defined label defining the tunneling region. It must be accompanied by a matching **st:nonlocal_transp_region** statement.

- **direction** determines whether the tunneling direction is within a mesh plane (xy) or across multiple mesh planes (z).

- **equiv_mobility** is the equivalent mobility for the transport of carriers across the junction when the physical models are not used. This will determine the slope of the I-V curve of the junction.

- **type2_quantum_well** is used to activate a special tunneling model for MQW/superlattice regions with type II band alignment. See **type2_qw_setting** for additional details regarding this model.

- **ignore_forward_current** can be used to artificially ignore the forward current to improve convergence in cases where the tunnel junction is reverse-biased (e.g. power applications).

- **tun_mass** is the Zener interband tunneling mass to be set by the user. If it is not specified, the combined effective mass is used as in the following formula: $2m_e m_h/(m_e + m_h)$.

- **type2_qw_mob_scale** can be used when **type2_quantum_well**=*yes* to scale the effective mobility on a per-junction basis. An additional parameter also exists in **type2_qw_setting** to scale the mobility for the whole simulation project.

- **silicon_holemass_fac** can be used to artificially scale the hole mass in the phonon-assisted silicon model.

- **scale_tj_current** can be used to artificially scale the tunneling current.

- **zseg_num** is the z-segment number in a 3D structure.

- **silicon_dir** is the silicon crystal orientation in the phonon-assisted model.

Examples

```
tunnel_junc y_start_label=y1 y_end_label=y2 equiv_mobility=0.02
$ The labels y1 and y2 are predefined positions.
```

22.747 two_photon_carr

This material statement is used to defined the optical loss due to free carriers generated by the two-photon absorption process[149]. This statement is used in passive regions and an alternate method of defining this term is also found in **passive_carr_loss**. For active regions, equivalent parameters are available in **active_reg** and **set_active_reg**.

The **two_photon_carr** statement defines the propagation power loss coefficient γ in Eq. 22.137. This term should not be seen as a separate single-step three-photon loss term but as a side effect of the two-photon absorption process described by β in the same equation[149].

In Eq. 22.137, the units of $\gamma * S^2$ must be in m^-1 so that γ is given in m^5. However, if one follows the form of Eq. 22.137 given in [149], γ is sometimes given in units of m^3/W^2 so that $\frac{d\phi}{dz}$ and $\gamma\phi^3$ are both in W/m^3. Following the same approach used for the β coefficient, a conversion factor of $(\hbar\omega v_g)^2$ may be used to convert this kind of value to the Crosslight convention.

Alternatively, we can also rewrite the relationship between β and γ given in [149] to reflect our choice of units:

$$\gamma = \frac{\sigma}{2}\tau\beta v_g \tag{22.136}$$

where σ is the total free-carrier absorption cross-section (usually defined using **elec_carr_loss** and **hole_carr_loss**) in m^2, τ is the carrier lifetime , v_g is the group velocity and β is in m^2.

Eq. 22.136 shows that if both free carrier losses and **two_photon_loss** are defined for a material, then a consistent value of **two_photon_carr** should also be defined for this material. Conversely, **two_photon_carr** should be zero if either of these terms is neglected in the model.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.748 two_photon_loss

This material statement is used to defined the optical loss due to the two-photon absorption process[149] in passive regions and an alternate method of defining this term is also found in **passive_carr_loss**. For active regions, equivalent parameters are available in **active_reg** and **set_active_reg**.

The **two_photon_loss** statement defines the propagation power loss coefficient β in the following equation:

$$\begin{aligned}\frac{dS}{dz} &= -\alpha S - \beta S^2 - \gamma S^3 \\ &= (-\alpha - \beta S - \gamma S^2)S \\ &= -\alpha'(S)S \end{aligned} \tag{22.137}$$

where S is the local photon density in m^{-3}. For the γ term, see **two_photon_carr**.

In the software, β is implemented as an absorption coefficient that depends on the local photon density. $\alpha'(S)$ is thus used to compute the stimulated recombination and generation terms in the drift-diffusion equation and additional derivatives in $\frac{\partial \alpha}{\partial S}$ are used to define the Jacobian. Since α and βS must have the same units and α is in m^{-1}, β must be given in m^2.

In reference [149], an alternate form of Eq. 22.137 which depends on the photon flux ϕ in W/m^2 is used so in that reference, β is given in m/W. These units can be converted to the convention used by PICS3D by noting that by definition, the photon flux and photon density are related by the group velocity and photon energy:

$$\phi = \hbar \omega v_g S \tag{22.138}$$

Using the above relationship in Eq. 22.137, it is trivial to show that a scaling factor of $\hbar \omega v_g$ exists between the two definitions of β. From reference [149] $\beta \approx 1.5 \times 10^{-10} m/W$. Assuming a group refractive index of 3.6 and an emission wavelength of 1.3 μm, the equivalent value β used in PICS3D should therefore be on the order of $1.9 \times 10^{-21} m^2$.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.749 type2_qw_setting

parameter	data type	values [defaults]
use_average_field	char	yes, [no]
mean_free_path	real	[0.1] (um)
bulk_radiative_coef	real	[1e-19] (m^3/s)
tau_trap	real	[1e-9] (s)
mobility_scale	real	[1.0]
qw_period	intg	
dos_scale_ref_period	intg	

This statement sets the parameters for the tunneling model when used in a type II QW photodetector It is used in conjunction with **tunnel_junc** with the parameter type2_quantum_well=yes.

The model is derived from [150] which deals with quantum cascade lasers with a tunneling-assisted transition. This paper presents a rate equation in (2) and a conversion to current density in (10). APSYS uses a similar equation except that the transitions considered are between conduction and valence subbands rather then between different conduction subbands.

Parameters

- **use_average_field** uses the average field to compute the long-range tunneling rather than local values which may quickly flucutate.
- **mean_free_path** is the mean free path of the carriers in μm.
- **bulk_radiative_coef** is an effective spontaneous emission coefficient for the superlattice.
- **tau_trap** is an effective trap lifetime for the superlattice.
- **mobility_scale** is a global scaling factor applied to the effective mobility of all type II tunneling regions.
- **qw_period** is the number of quantum well periods in the project. It is used in conjunction with **dos_scale_ref_period** to scale the 1D DOS. This allows more carriers per energy unit to participate in the tunneling in larger devices.
- **dos_scale_ref_period** is the reference number of periods used to scale the 1D DOS. If this parameter is used, **qw_period** must also be defined; otherwise, no scaling is done. This parameter can be used to account for the change in responsivity with increased number of periods.

Examples

```
$ mean-free-path mainly affect longer range tunneling current.
type2_qw_setting mean_free_path=0.4 bulk_radiative_coef=3.e-19
```

22.750 unified_schottky_local_tunneling

parameter	data type	values [defaults]
scale_tunnel_mass	real	[1.]
max_range	real	[0.01]
scale_current	real	[1.]

unified_schottky_local_tunneling converts the non-local tunneling integral into a local carrier generation rate[151, 152].

This simplified approach has the advantages of being convenient and easy to define but the artificial generation of holes may have unphysical effects.

Parameters

- **scale_tunnel_mass** is an artificial scaling coefficient for the tunneling mass.
- **max_range** defines a region around the top of the barrier that is used for the carrier generation.
- **scale_current** is an artificial scaling coefficient for the tunneling current.

22.751 use_bulk_affinity

parameter	data type	values [defaults]
all_material	char	[yes]
macro_name	char	
mater_label	char	
mater	intg	[1]

use_bulk_affinity is an active layer macro statement used to define the band offset of the quantum well. It directs the program to use bulk material affinity to define band offsets of the active layers.

This statement overrides all other band offset declarations including **band_offset** and **band_discont**.

Parameters

- **all_material** lets all active layer materials to use bulk affinity as band offsets.
- **macro_name** indicates all materials with this macro name are affected.
- **mater** is the material number affected. It is not used if **macro_name** is already set.
- **mater_label** may be used instead of **mater** if a label has previously been defined as an alias.

Examples

```
use_bulk_affinity all_material=yes
```

22.752 use_bulk_bandgap

This command would force the program to take the bandgap of the bulk macro if there is a conflict between active and bulk macros.

22.753 use_bulk_property

This command would force the program to take the material parameters of the bulk macro if there is a conflict between active and bulk macros.

This command affects bandgap, affinity and effective masses.

22.754 use_macrofile

parameter	data type	values [defaults]
macroj (j=1,2,...)	char	[void]

The statement **use_macrofile** is used to override or to add to the default macro files crosslight.mac and more.mac. The simulator will check macro files loaded by this statement before going to the default macros. More than one macro files can be loaded this way

- **macroj** (j=1,2,...) is the user-defined macro file.

```
Example(s):
use_macrofile macro1=mydata.mac
```

22.755 user_defined_mobility

parameter	data type	values [defaults]
carrier_type	char	[electron], hole
dependence	char	[parallel_field], total_field
curve_control	char	[vertical_field] total_field, electrode_voltage
data_type	char	[mobility], velocity, current
filek (k=1...9)	char	[my_mob_data_file]
spline_smooth	char	[no]
mater_label	char	
scale_column1	real	[1.]
scale_column2	real	[1.]
scale_curve_para	real	[1.]
curve_parak (k=1...9)	real	
mater	intg	[1]
electrode	intg	[2]

The **user_defined_mobility** command is used to import user-defined mobility into Crosslight simulation software. Whenever this command is used for a material, it will override whatever default mobility model or previously defined mobility model it may have had.

The user supplied mobility model is imported as a 2-column data file to define the mobility versus field relationship. Other forms such as velocity and current can also be used; the software will convert the data into mobility before using it. One may use several data files to define several regimes of the mobility.

Parameters

- **carrier_type** is the type of carrier being defined.
- **dependence** defines whether the mobility depends on the field component parallel to the current flow, or on the total local field magnitude.

- **curve_control** is the parameter used to switch between data files. The program automatically linearly interpolate between different data files as necessary.

- **data_type** is the type of data being imported in the 2nd column (mobility, velocity, or current) of the data files.

- **filek (k=1...9)** are the file names for the imported data. The data should be in SI units or scaled back into SI units using **scale_column1** and **scale_column2**.

- **spline_smooth** determines whether spline smoothing is applied to the user-defined data. By default, a simple linear interpolation method is used.

- **scale_curve_para** is a scaling factor for **curve_parak** below.

- **curve_parak** is the value of the control parameter used to switch between data files.

- **mater** is the number assigned to the material affected by this command. If a label alias has previously been assigned to this material, **mater_label** may be used instead.

- **electrode** is the electrode number if control parameter is based on electrode-voltage.

Examples

```
user_defined_mobility carrier_type=electron &&
  curve_control=electrode_voltage data_type=velocity &&
  curve_para1=4.5 &&
  file1=mobdata1.txt &&
  curve_para2=9. &&
  file2=mobdata2.txt &&
  curve_para3=13.5 &&
  file3=mobdata3.txt &&
  mater=1 electrode=2
```

The above example allows the import of velocity versus parallel field data in 3 different data files. These data files are used for 3 different voltages (4.5, 9 and 13.5 volts) of electrode number 2. The program would interpolate if the electrode voltage falls between these curve parameters.

The content of each file should be similar to the following:

```
0.0       0.1
1.e5     0.081
1.e6     0.02
1.e7     0.02
```

22.756 use_sor

parameter	data type	values [defaults]
print_sor	char	[noprint]
wave_tol	real	[1.e-7]
beta_tol	real	[1.e-7]
omega	real	[1.5]
omegab	real	[1.5]
max_iter	intg	[5000]
update	intg	[15]

use_sor instructs the software to use the SOR iterative method to find the solution of lateral modes. This statement was previously defined as **sor_par**.

The initial guess to the SOR method is either a uniform field profile based on the mode solver limits in **init_wave** or the result of the **optical_field** command.

Application Notes

The SOR method is considered obsolete and the user is *strongly* encouraged to migrate to either **optical_field** or **direct_eigen**, depending on the structure.

Parameters

- **print_sor** controls the printing of the SOR iterations.
- **wave_tol** is the tolerance for the wave-amplitude. Note that the wave amplitude is normalized to the order of 1 inside the SOR method.
- **beta_tol** is the tolerance for the eigenvalue, which is also normalized to the order of 1 inside the SOR method.
- **omega** is the initial relaxation parameter used for the wave amplitude iteration. This parameter is adaptive during the iteration procedure.

- **omegab** is the initial relaxation parameter used for the eigenvalue. This parameter is also adaptive during iteration.

- **max_iter** is the maximum number of iterations. If it is zero, the initial guess is not refined.

- **update** is the interval of iterations at which the eigenvalue is updated inside the SOR method.

Examples

```
use_sor max_iter=0000

use_sor max_iter=5000 print_sor=noprint
```

22.757 valj_mass_para (j=1..3)

parameter	data type	values [defaults]
(see) material_par		

This statement is used in active macros to define the relative valence band valley mass in a direction parallel to the quantum well. It is used for general complex strained macros; the most common applications are for strained silicon, SiGe and II-VI lead salt materials.

The mass defined in this statement can also be modified by non-parabolic terms with the **valj_para_e_dep_mass1** and **valj_para_e_dep_mass2** statements:

$$\begin{aligned} m(E) &= a + bE + cE^2 \\ E(k) &= \frac{\hbar^2 k^2}{2m_0 m(E)} \end{aligned}$$

j is a placeholder value that must be the same in all three statements: it refers to the number ($j = 1..3$) of the valley being defined.

Parameters

The parameters for this statement are the same as for all other material statements and are discussed under **material_par** in Sec. 22.456.

Examples

```
layer_type type=general_cx_strain valley_c1=4 valley_v1=4 &&
  optic_trans_valley_pair1=(1 1)

$ Define band gap as a bulk_xfunc1, to be referred to later by other functions
ext_func1 variation=function
function(x,temper)
0.17+0.057*x-0.095*x**2+sqrt(4.e-4+2.56e-7*temper**2)
end_function

val1_mass_para value=0.05

val1_para_e_dep_mass1 variation=function
function(x)
em_parab=0.05;
egt=ext_func1;
2.*em_parab/egt
end_function
```

Which is equivalent to $m(E) = 0.05 + 0.1\frac{E}{E_g}$ in the notation above.

References

- Khodr & al., "Effects of band nonparabolicity ...", IEEE JQE, VOL. 32, NO. 2, Feb. 1996

22.758 valj_mass_perp (j=1..3)

parameter	data type	values [defaults]
(see) material_par		

This statement is used in active macros to define the relative valence band valley mass in a direction perpendicular to the quantum well. It is used for general complex

strained macros; the most common applications are for strained silicon, SiGe and II-VI lead salt materials.

The mass defined in this statement can also be modified by non-perpbolic terms with the **valj__perp__e__dep__mass1** and **valj__perp__e__dep__mass2** statements:

$$\begin{aligned} m(E) &= a + bE + cE^2 \\ E(k) &= \frac{\hbar^2 k^2}{2m_0 m(E)} \end{aligned}$$

j is a placeholder value that must be the same in all three statements: it refers to the number ($j = 1..3$) of the valley being defined.

Parameters

The parameters for this statement are the same as for all other material statements and are discussed under **material__par** in Sec. 22.456.

Examples

```
layer_type type=general_cx_strain valley_c1=4 valley_v1=4 &&
  optic_trans_valley_pair1=(1 1)

$ Define band gap as a bulk_xfunc1, to be referred to later by other functions
ext_func1 variation=function
function(x,temper)
0.17+0.057*x-0.095*x**2+sqrt(4.e-4+2.56e-7*temper**2)
end_function

val1_mass_perp value=0.05

val1_perp_e_dep_mass1 variation=function
function(x)
em_parab=0.05;
egt=ext_func1;
2.*em_parab/egt
end_function
```

Which is equivalent to $m(E) = 0.05 + 0.1\frac{E}{E_g}$ in the notation above.

References

- Khodr & al., “Effects of band nonparabolicity ...”, IEEE JQE, VOL. 32, NO. 2, Feb. 1996

22.759 valj_para_e_dep_mass1 (j=1..3)

parameter	data type	values [defaults]
(see) material_par		

This statement modifies **valj_mass_para** and adds a linear energy dependent term to the electron mass. j is a placeholder used to identify the valley ($j = 1..3$).

22.760 valj_para_e_dep_mass2 (j=1..3)

parameter	data type	values [defaults]
(see) material_par		

This statement modifies **valj_mass_para** and adds a quadratic energy dependent term to the electron mass. j is a placeholder used to identify the valley ($j = 1..3$).

22.761 valj_perp_e_dep_mass1 (j=1..3)

parameter	data type	values [defaults]
(see) material_par		

This statement modifies **valj_mass_perp** and adds a linear energy dependent term to the electron mass. j is a placeholder used to identify the valley ($j = 1..3$).

22.762 valj_perp_e_dep_mass2 (j=1..3)

parameter	data type	values [defaults]
(see) material_par		

This statement modifies **valj_mass_perp** and adds a quadratic energy dependent term to the electron mass. j is a placeholder used to identify the valley (j = 1..3).

22.763 val1_valley_prop1

valj_valley_propk are a set of active layer macro statements: j and k are placeholder values that indicate the valley subband valley number (j=1..3) and the property number (k). It is otherwise the same as **cond1_valley_prop1**.

22.764 val2_valley_prop1

See **val1_valley_prop1**.

22.765 val2_valley_prop1

See **val1_valley_prop1**.

22.766 val_bandj_edge (j=2,3)

parameter	data type	values [defaults]
(see) material_par		

The material statement **val_bandj_edge** is an active layer macro statement used to define the 2nd or 3rd valence band valley with respect to the highest valence band valley, in unit of eV. It is used for general_cx_strain type of macro, or general complex strain macro.

The parameters for this statement are the same as for all other material statements. The use of these parameters and related examples are given under **material_par** in section 22.456.

22.767 vcsel_cavity_region

parameter	data type	values [defaults]
radius_from	real	[-1.e89] (μm)
radius_to	real	[1.e89] (μm)
gaussian_tail	real	[2.] (μm)
gain_reduc_range	real	[0.0] (μm)
gain_gaussian_tail	real	[0.5] (μm)

vcsel_cavity_region defines a new independent optical region in a VCSEL structure. It is used in conjunction with **begin_cavity**.

Parameters

- **radius_from** and **radius_to** partition the optical mode (which is calculated over the entire VCSEL) and assigns it to the new cavity.
- **gaussian_tail** extends the partitioned mode past the **radius_from** and **radius_to** coordinates with a Gaussian tail to prevent a sharp cut-off; the value of this parameter is equal to the standard deviation of the Gaussian distribution.
- **gain_reduc_range** reduces the optical gain at the edges of the partition to account for the influence of the Gaussian tail from neighboring cavities; the reduction is fixed over this range.
- **gain_gaussian_tail** serves the same purpose as **gain_reduc_range** but reduces the optical gain following a Gaussian tail distribution instead of a fixed amount. This tail points towards the inside of the partition, like the **gaussian_tail** from a neighboring cavity; the value of this parameter is equal to the standard deviation of the Gaussian distribution.

Examples

```
begin_cavity cavity_num=1
vcsel_model index_core=3.2 index_cladding=1.0 &&
  core_radius =9.5  bessel_order=0
init_wave backg_loss=500 init_wavel=0.83  wavel_range=(0.75, 0.90) &&
      photon_fac=1.e9
vcsel_cavity_region  radius_from=0 radius_to=5
```

```
multimode mode_num=2
end_cavity

begin_cavity cavity_num=2
vcsel_model index_core=3.2 index_cladding=1.0 &&
  core_radius =9.5  bessel_order=0
init_wave backg_loss=500 init_wavel=0.83  wavel_range=(0.75, 0.90) &&
      photon_fac=1.e9
vcsel_cavity_region  radius_from=5 radius_to=9.5
multimode mode_num=2
end_cavity
```

22.768 vcsel_model

parameter	data type	values [defaults]
use_eim	char	[no],yes
rectangle_system	char	[no]
rect_symetric_x	char	[yes]
propagation_dir	char	[y],z
rect_lateral_model	char	[eim], bessel
independent_stw	char	yes, [no]
eim_stw_method	char	[min_loss], max_gain, min_abs_loss
index_core	real	[3.5]
index_cladding	real	[2.0]
core_radius	real	
photon_fac	real	[1.e-3]
eim_zero	real	(um)
rect_core_xrange	real	(um)
rect_core_zrange	real	(um)
rect_core_origin	realx2	[0. 0.](um)
zdir_x_ref	real	[0.0] (um)
zdir_y_ref	real	[0.0] (um)
ydir_x_center	real	[0.0] (um)
ydir_z_center	real	[0.0] (um)
bessel_order	intg	[0]
add_r_division	intg	[1]
rect_x_order	intg	[1]
rect_y_order	intg	[1]
long_ref_zseg	intg	

vcsel_model is used to activate the Vertical Cavity Surface Emitting Laser (VCSEL) model in the .sol file.

Parameters

- **use_eim** activates the effective index method model for the VCSEL simulation environment. By default, this parameter is set to *no* and the fiber-like EIM model is used.
- **rectangle_system** turns on the rectangle/cartersian coordinate system for VCSEL simulation.
- **rect_symetric_x** indicates whether only half of a symmetric device is simulated to save mesh. If so, the symmetric axis is assumed to be located at x=0.
- **propagation_dir** defines the optical propagation (longitudinal) direction in 3D simulations with multiple mesh planes. In almost all cases, this should be left to the default setting of y. As of the 2016 version of the software, convergence using multiple mesh planes stacked along the propagation direction of z still has not been demonstrated and should be considered experimental.
- **index_core** and **index_cladding** are the refractive indices of the core and cladding regions in the fiber-like EIM method. These parameters are effective only if **use_eim**=*no*.
- **core_radius** is the radius of the core region in the fiber-like EIM method. This parameter is effective only if **use_eim**=*no*. Please note that this parameter should not be confused with the identically-named parameter in the **vcsel_section** and **outer_section** commands: the radius defined in those two commands are used only for the EIM model (**use_eim**=*yes*).
- **photon_fac** is the photon factor used to normalize the cavity photon number for VCSELs for the round-trip gain equation. It multiplies the parameter of the same name defined in **init_wave** so that the default scaling of the cavity photon number is different for edge-emitting and VCSEL cavities. As such, either of these parameters can be used if the user wishes to modify the default scaling.
- **eim_zero** is the radius outside of which the EIM wave function is forced to be zero. It is used only when the EIM model is activated by the parameter **use_eim** above.

- **rect_lateral_model** is used in 3D simulations that use a cartesian coordinate system but which may also have bent shapes to form a cylindrical shape. It specifies wether the software will use the EIM model or Bessel functions for the modes.

- **rect_core_xrange** and **rect_core_zrange** specify the dimensions of the high index core in the x and z directions, respectively when rectangular/cartesian coordinates are used in a 3D simulation and **propagation_dir**=*y*. **rect_core_origin** is similar and defines the origin (lower left corner) of the rectangular high index core.

- **ydir_x_center** and **ydir_z_center** are used to define the center of the x-z plane (i.e. the origin of the Bessel functions) when **rect_lateral_model**=*bessel*. **zdir_x_ref** and **zdir_y_ref** serve the same function when **propagation_dir**=*z* (experimental).

- **bessel_order** is the order of the Bessel function used to model the wave function in the modified effective index method. Higher order means higher order lateral modes.

- **add_r_division** is used for the EIM model (**use_eim**=*yes*). When this model is activated, **add_r_division** controls the number of divisions in the lateral direction used to evaluate the effective index. More precisely, the software will use a 1D transfer matrix at various vertical cut lines to establish $n(r, z)$ profiles, each line representing a local standing wave profile. At each r value, a single standing wave effective index is selected to provide a $n(r)$ profile which is then used to establish the lateral mode profile.

  The position of these cut lines is determined by an initial rough division of the device into columns, as seen in Fig. 22.32; these are normally defined in the .layer file through the **vcsel_section** command and the resulting **outer_section** statement. These divisions may be further refined if they exceed the value of ΔR:

$$\Delta R = \frac{R_m - R_0}{\text{add\_r\_division}} \tag{22.139}$$

- **eim_stw_method** controls how the local standing wave profile is determined in the EIM model described above. For each r position, an 1D eigenvalue problem is solved for z and a single standing wave profile is selected based on the following effective index criteria:

  - *min_loss* picks the mode with the smallest loss coefficient
  - *max_loss* picks the mode with the largest gain coefficient

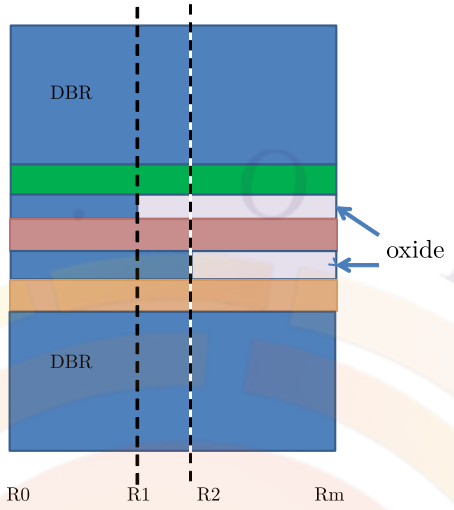


Figure 22.32: Schematics of the cut lines in the VCSEL EIM model.

  - *min_abs_loss* picks the mode with the smallest absolute gain/loss coefficient

- **rect_x_order** and **rect_y_order** define the number of modes solved for in each direction for 3D simulations with rectangular/cartersian coordinates.
- **long_ref_zseg** is used in 3D simulations with multiple mesh planes. It defines which z-segment serves as a reference for the optical **section** statements. All other z-segments are mapped onto this reference segment.
- **independent_stw** works with **add_r_division** and the EIM model. By default, the software will calculate the standing wave (STW) pattern only once to save on time; enabling this option will evaluate the STW for every cut line which results in a slower but more accurate simulation. It is recommended to enable this option in devices where there is a strong lateral variation of the index profile (e.g. oxide-confined VCSELs).

Examples

Simple example using fiber-like model:

```
vcsel_model index_core=3.2 index_cladding=1.0 &&
  core_radius =7.5  bessel_order=0
```

Simple example using EIM model settings inherited from .layer file:

```
vcsel_model bessel_order=0 use_eim=yes
```

A rectangular-shaped VCSEL:

```
vcsel_model index_core=3.2 index_cladding=1.0  rectangle_system=yes
```

22.769 vcsel_section

parameter	data type	values [defaults]
vcsel_type	char	
grating_model	char	[kappa]layer,layer0
active	char	[no],yes
dbr_period_index_profile	char	
dbr_period_from_macro	char	[no],yes
layerk,k=1..9	real	[0.096e-6] (m)
indexj,j=1..9	real	
core_radius	real	(um)
mesh_points	intg	[10]
remove_from_cavity	intg	[]
section_index	real	
outer_indexi,i=1..9	real	[]

vcsel_section is used to describe physical properties of a layer region used in a VCSEL. While it used exclusively in the .layer file, this statement defines optical properties which are different than the electrical/geometrical settings defined in other .layer commands. When the .layer file is processed, the information included in this statement will be included in a .vcsel file which must be included in the main .sol input file.

Parameters

- **vcsel_type** defines a label for the VCSEL section which is used in other statements such as **layer**.
- **active** is used to indicate if a VCSEL section is active (i.e. has an optical gain). This parameter is also used to locate the active region and compute the standing wave enhancement factor.
- **dbr_period_index_profile** may be used to give the name of a text file that contains the refractive index profile for a single DBR period instead of directly

using the **layerk** and **indexj** values. This method makes it simple to define a graded index profile for the DRB period.

- **dbr_period_from_macro** instructs the software to adjust the thickness of the overall DBR for the optical mesh (c.f. **section** statement in the .vcsel output file) to match the refractive index values of the individual materials inside the DBR period.

  Note that this parameter requires defining the individual components of the DBR using **vertical_dbr_layer_mater** instead of directly using the **layerk** and **indexj** values. Graded composition profiles may be defined using the latter statement.

  We also note that the overall electrical thickness for that layer must match the previously-defined optical thickness; this is set in the **layer** statement.

- **grating_model** defines the grating model for the optical propagation in that section; see **section** and the parameter of the same name for more information.

- **layerk**,k=1..9 are the different DBR layer thicknesses (in meters) used in this section, with the appropriate **grating_model** setting. The refractive index of the DBR layers is likewise defined using **indexj**,j=1..9. See **section** for more information on DBR settings.

- **outer_indexi**,i=1..9 is used to determine the refractive index outside the VCSEL in the fiber-like effective index model. If the electrical layer is subdivided into multiple optical sub-sections (e.g. a DBR mesa in the first column, with air outside), a separate value must be provided for each of those sub-sections.

  When the .layer file is processed, the outer index settings are transferred to the **outer_section** statement for use in the .sol file.

- **core_radius** is used as part of the VCSEL EIM model (see **vcsel_model**). It defines the radius of a particular region for mode calculations and tells the simulator that this section is terminated by an outside insulator layer (e.g. air). In complex devices, multiple VCSEL section declarations with different core radius settings may be used to define different effective index profiles in different columns of the device; in that case, **core_radius** should be set to the same value as the column width, as seen in Fig. 22.32.

  When the .layer file is processed, the core radius settings are transferred to the **outer_section** statement for use in the .sol file.

- **mesh_points** is the number of longitudinal mesh points used in the optical propagation model for this section.

- **section_index** is identical to **index1** above.

- **remove_from_cavity** removes that particular section from existence for the given optical cavity number.

  This statement may be used in the multicavity model (see **begin_cavity** statement)and produces a **remove_section** statement for used in the .sol file.

Examples

A standard definition using older syntax. Under this model, the refractive index values for the DBR are fixed for the entire simulation.

```
vcsel_section vcsel_type=n-dbr &&
  grating_model=2layers active=no &&
  layer1 =0.122e-6 layer2 =0.112e-6 &&
  index1 = 3.17 index2 = 3.46 &&
  mesh_points=15
```

A similar definition using negative index numbers. Under this model, only the absolute difference in refractive index values is used for the DBR index step, with the average refractive index in the layer taken from the macros according to injection/thermal conditions. For more explanation of negative index numbers, please see the **section** statement.

```
vcsel_section vcsel_type=n-dbr &&
  grating_model=2layers active=no &&
  layer1 =0.122e-6 layer2 =0.112e-6 &&
  index1 = -3.17 index2 = -3.46 &&
  mesh_points=15
```

A more modern declaration style, with all index values taken from the material macros. The electrical mesh still consists of an "average" material.

```
vcsel_section vcsel_type=n-dbr &&
  dbr_period_from_macro=yes &&
  active=no mesh_points=10

$ this is the effective medium
layer_mater macro_name=algaas var1=0.625 column_num=1 var_symbol1=x

$ let's define a DBR period using macro like this (use column 1 only)
$ (also possible to define grading within a DBR period)
```

```
vertical_dbr_layer_mater macro_name=algaas var_symbol1=x var1=0.25 &&
  thick=0.0595
vertical_dbr_layer_mater macro_name=algaas var_symbol1=x var1=1. &&
  thick=0.0706

layer d=3.7729  n=15  r=0.9 &&
  n_doping1=2.e24  vcsel_type=n-dbr
```

22.770 vectorial_wave

parameter	data type	values [defaults]
mode	char	[te] tm

The statement **vectorial_wave** is used to turn on the vectorial wave solver. Please note that this statement only affects the wave distribution. It does not affect the optical evaluation. You should also set the polarization in **active_reg** for the TE or TM mode.

- **mode** is the polarization mode and may take te or tm.

Example:

```
vectorial_wave mode=te
```

22.771 vertical_dbr_layer_mater

parameter	data type	values [defaults]
macro_name	char	
var_symbol1	char	
var_symbol2	char	
var_symbol3	char	
var_symbol4	char	
var_symbol5	char	
var_symbol6	char	
var_symbol7	char	
var_symbol8	char	
var_symbol9	char	
mater_lib	char	
var1	real	[-9999.]
var2	real	[-9999.]
var3	real	[-9999.]
var4	real	[-9999.]
var5	real	[-9999.]
var6	real	[-9999.]
var7	real	[-9999.]
var8	real	[-9999.]
var9	real	[-9999.]
grade_from	real	[-9999.]
grade_to	real	[-9999.]
thick	real	[-9999.]
grade_var	intg	[0]
grade_points	intg	[5]

This command is used to define VCSEL DBR layers using material macros instead of index values. It only defines the optical part of the model (VCSEL sections).

Parameters

Most of the parameters are the same as in the **layer_mater** statement as both commands define material compositions. Size information for the DBR is given by:

- **thick** is the thickness of the DBR layer.

- **grade_points** is the number of points used to sample the grading within the DBR layer.

Examples

```
vcsel_section vcsel_type=n-dbr &&
  dbr_period_from_macro=yes &&
  active=no mesh_points=10

$ this is the effective medium
layer_mater macro_name=algaas var1=0.625 column_num=1 var_symbol1=x

$ let's define a DBR period using macro like this (use column 1 only)
$ (also possible to define grading within a DBR period)
vertical_dbr_layer_mater macro_name=algaas var_symbol1=x var1=0.25 &&
  thick=0.0595
vertical_dbr_layer_mater macro_name=algaas var_symbol1=x var1=1. &&
  thick=0.0706
layer d=3.7729  n=15  r=0.9 &&
  n_doping1=2.e24  vcsel_type=n-dbr
```

22.772 view_dipole

parameter	data type	values [defaults]
data_file	char	[void]
mode	char	[te]tm
gamma_subband	intg	[1]

view_dipole plots the dipole moments in a k.p theory.

- **data_file** is the file to which the graphic data is written in ASCII format.
- **mode** denotes the polarization.
- **conc_range** is the electron concentration range in the well.
- **gamma_subband** is the Gamma subband used in the optical transition.

Example(s)

```
view_dipole gamma_subband=1
```

22.773 view_ganvalence

parameter	data type	values [defaults]
data_file	char	[void]
data_file2	char	[void]
matrix_6x6	char	yes,[no]
direction	char	[x],y,z
region	char	[well],barrier,barrier2

view_ganvalence is used in the .gain file to plot the active region basic bulk band structure (ie., HH, LH, CH) for wurtzite GaN-based structure along with parabolic fit

Parameters

- **data_file** is the data file to which the band structure data is to be saved. If **plot_device**=*data_file* in **plot_data**, **data_file2** may be used to specify a second data file in which to save the data.
- **matrix_6x6** determines if the full 6×6 wurtizte Hamiltonian is used.
- **direction** is the direction of the wave vector used to plot the band structure.
- **region** is used to specify if the band structure for well or barrier is to be plotted.

Examples

```
view_ganvalence region=well
```

22.774 view_zincblende_valence

This statement and its parameters are analogous to **view_ganvalence**. However, it applies to cubic zincblende materials so a different Hamiltonian is solved.

22.775 view_kpsubband

parameter	data type	values [defaults]
data_file	char	[void]
include_data	char	[void]
versus_q	char	[no]
mqw_complex	intg	[1]

view_kpsubband is used in the .gain file to plot the quantum well subbands ($E(k_t)$) from k.p theory.

- **data_file** is a text file which is used to save a copy of the band structure data.
- **include_data** includes data files from other subband calculations for the purpose of comparison. The format is a column-wise list of k_t and energy E; data from different subbands are separated by an empty line.
- **mqw_complex** is used to select which MQW is being plotted if there are multiple quantum-confined regions in the simulation.
- **versus_q** changes the axis of the plot and is used only if periodic boundary conditions have been defined in **modify_qw**. In that case, different energy dispersion relations can be calculated: as normal, vs. the in-plane wave vector k_t or vs. the periodicity coefficient q.

  The periodicity coefficient is defined in the interval $[0, \frac{\pi}{L}]$ and covers the entire range of possible periodic boundaries[66]:

$$\Psi(L) = \Psi(0)e^{iqL} \tag{22.140}$$

22.776 view_kpwave

parameter	data type	values [defaults]
versus_q	char	[no]
kt	char	[1]

view_kpsubband is used in the .gain file to plot wave functions ($E(k_t)$) from k.p theory. As of the 2015 version, this is only available for the newer version of the zincblende k.p solver.

- **versus_q** changes the axis of the plot and is used only if periodic boundary conditions have been defined in **modify_qw**. In that case, different energy dispersion relations can be calculated: as normal, vs. the in-plane wave vector k_t or vs. the periodicity coefficient q.

  The periodicity coefficient is defined in the interval $[0, \frac{\pi}{L}]$ and covers the entire range of possible periodic boundaries[66]:

$$\Psi(L) = \Psi(0)e^{iqL} \tag{22.141}$$

- **kt** controls which wave functions are being plotted. This value ranges from 1 (at zone center) to the number of k.p points specified in **modify_qw**.

22.777 view_macro

view_macro is a programmer-only command used to evaluate macros with external tools.

22.778 view_vtkfile

parameter	data type	values [defaults]
vtkfile	char	
dimension	char	[2d],3d

view_vtkfile is used in the post-processing stage to visualize FDTD data stored in the VTK file format with the help of an external viewer called Paraview.

Parameters

- **vtkfile** is the filename containing the FDTD data.
- **dimension** tells the viewer to expect 2D or 3D data.

22.779 virtual_time_setting

parameter	data type	values [defaults]
element_name	char	[], Vcc, I1
scan_num	intg	[2]

virtual_time_setting modifies a particular **scan** statement so that when the *virtual_time* variable is used, only one particular current or voltage source in the **minispice** external circuit is modified.

If this statement is omitted, *virtual_time* modifies all SPICE source devices simultaneously during the scan.

Parameters

- **element_name** is the SPICE device name that will be modified during the scan
- **scan_num** sets which **scan** statement is altered by this command

22.780 vplot_xy

parameter	data type	values [defaults]
variable	char	(see list)
data_file	char	
mater_boundary	char	[yes]no
point_ll	realx2	(μm)
point_ur	realx2	(μm)
xrange	realx2	
yrange	realx2	
z	real	(micron)
maxvector_scale	real	[1.]
grid_sizes	intgx2	[20, 20]

vplot_xy is a post-processor statement used to plot structural data on a 2D plane. Unlike **plot_2d**, this command can only plot vector fields.

Other related commands exist and should be used depending on the 2D/3D nature of the original simulation results. The following rules apply:

- 2D simulations: use **plot_2d**
- 3D cylindrical simulations with one mesh plane: use **plot_2d**

- xy plane from a 3D simulation: use **cplot_xy** for contour plots of scalar variables, **splot_xy** for 3D surface plots of scalar variables or **vplot_xy** for vector variables.
- xyz plane from a 3D simulation: use **cplot_xyz** for contour plots of scalar variables, **splot_xyz** for 3D surface plots of scalar variables or **vplot_xyz** for vector variables.

Parameters

With the exception of the z-position and contour plot-related parameters that do not apply, all parameters are the same as in **plot_2d**.

- **z** is the position on the z-axis for the 2D plot. If necessary, the variable data will be interpolated from neighboring mesh planes.

Examples

```
vplot_xy variable=elec_curr z=50. grid_sizes=(35, 35) &&
  xrange=(0. 1.5) yrange=(1.1 1.9)
```

22.781 vplot_xyz

parameter	data type	values [defaults]
variable	char	(see list)
data_file	char	
xy_from	realx2	(μm)
xy_to	realx2	(μm)
xrange	realx2	
yrange	realx2	
z_min	real	(micron)
z_max	real	(micron)
maxvector_scale	real	[1.]
grid_sizes	intgx2	[35, 35]

---

vplot_xyz is a post-processor statement used to plot structural data on a 2D plane. Unlike **plot_2d**, this command can only plot vector fields.

Other related commands exist and should be used depending on the 2D/3D nature of the original simulation results. The following rules apply:

- 2D simulations: use **plot_2d**
- 3D cylindrical simulations with one mesh plane: use **plot_2d**
- xy plane from a 3D simulation: use **cplot_xy** for contour plots of scalar variables, **splot_xy** for 3D surface plots of scalar variables or **vplot_xy** for vector variables.
- xyz plane from a 3D simulation: use **cplot_xyz** for contour plots of scalar variables, **splot_xyz** for 3D surface plots of scalar variables or **vplot_xyz**for vector variables.

Parameters

This statement is similar to **vplot_xy** and varies only in the way it defines the plotting plane. As such, most of the parameters are also similar to those in **plot_2d**.

- **xy_from** and **xy_to** define the (x,y) corners of the plotting plane.
- **z_min** and **z_max** define the z coordinates of the corners of the plotting plane.

Examples

```
vplot_xyz variable=elec_curr  &&
  xy_from=(0.5 0.) xy_to=(0.5 3.)
```

22.782 wave_boundary

parameter	data type	values [defaults]
x1_label	char	[void]
y1_label	char	[void]
x2_label	char	[void]
y2_label	char	[void]
point_ll	realx2	[0. 0.](μm)
point_ur	realx2	[1. 1.](μm)
fld_center	realx2	(μm)
n_ubdata	realxn	
n_lbdata	realxn	
ubdata_num	intg	[0]
lbdata_num	intg	[0]

The statement **wave_boundary** sets the boundary condition for the optical wave equations. Use of this statement overrides the corresponding parameters in in statement **init_wave**. This statement can be generated by the graphic user interface and appears in the .doping file.

- **x1_label y1_label x2_label y2_label** user pre-defined labels to specify **point_ll**, **point_ur** parameters, respectively. The labels must be predefined using **x_position**, **y_position** or **define_vertical_position** command. They may be predefined in the .layer file relative to a specific layer so that there is no need to find out the absolute coordinates.

- **point_ll** and **point_ur** are lower-left and upper-right points of the window within which the wave equation is solved (see also Fig. 22.13). This window must be less than or equal to the full device size defined in the .geo input file. In some cases, this window must be reduced in size because only a limited number of lateral modes are needed. In gain guided devices, the lasing mode is not the highest index mode. Since PICS3D searches the optical mode starting from the highest mode index, there may be many non-lasing modes within the window. In the current version, PICS3D can only solve up to 10 modes. Since the number of non-lasing, high index modes increases with size of the window, the user should carefully adjust the window size for gain guided structures so that the lasing mode is included within the 10 modes.

- **fld_center** is the estimated optical mode center. Note that this information is only used to initialize the wave function and may or may not coincide with the optical field center in the final solution.

- **n_ubdata** and **n_lbdata** are window specifications. By default, the window for the wave equation is specified by "point_ll" and "point_ur" to be a rectangular window. Now it is possible to override the upper and lower borders with the parameters **n_ubdata**, **n_lbdata**, **ubdata_num** and **lbdata_num** where **n_ubdata** (for upper boundary data) is used to specify the upper boundary points (xi, yi) in the format

  n_ubdata=(x1 y1 x2 y2 x3 y3 ...)
  where the prefix n_ should be equal to the number of data values in the bracket (or half of the points in 2D space). The related parameter for this statement, **ubdata_num**, must match the number of data value in **n_ubdata**.

```
init_wave ... &&
  ubdata_num=8 8_ubdata=(0. 3.16 0.773 3.16 1.5 2.16 6. 2.16)
```

is used to describe a non-planar metal boundary for the wave equation. Similarly, **n_lbdata** and **lbdata_num** are used to describe the lower boundary.

Example(s)

```
wave_boundary point_ll=(0., 0.0) point_ur=(3.5, 3.0) &&
  fld_center=(0.5, 1.5)
```

22.783 waveguide_input

This command is identical to **3d_amplifier_model**. It exists only for historical reasons where separate models in PICS3D were used for semiconductor optical amplifiers (SOAs) and electro-absorbing modulators (EAMs). This is no longer necessary now that the equations are fully coupled with the round-trip gain method and the same model can handle both kinds of devices.

Note that for completely passive photo-absorbing waveguides, an additional flag is needed to properly use the bulk absorption coefficient in the simulation. Please see **3d_amplifier_model** for details.

22.784 wurtzite_offset_model

parameter	data type	values [defaults]
use_strained_bandgap	char	[yes]
bulk_strain_exist	char	[no]
use_hh_as_bandgap	char	[yes]

The this command is used to control various aspects of the band offset model for wurtzite materials; to understand it properly, a few key aspects of the band alignment rules in Crosslight should be reviewed:

- 1 Bulk layers (active and passive) directly define the conduction band position using the **affinity** statement. No strain shifts are applied to this value unless they are deliberately included the affinity declaration (macro file).

  For passive layers, only a single band is ever used for transport so all valence bands are combined, with a reduced mass accounting for the HH, LH and CH bands. The valence band edge is thus positioned as $E_v = E_c + E_g$.

For bulk active layers, all of the valence band edges are explicitly considered so using the effective bandgap for each valley, the valence bands are positioned as $E_{v,i} = E_c + E_{g,i}$.

- 2 Quantum wells position the conduction band relative to the barrier based on the **band_offset** value; this can be overridden by certain commands such as **use_bulk_affinity** and **band_discont**. Various authors define the band offset in different ways which may or may not include the strain; see Sec. 10.1 for more information on this topic.

  For a quantum well, each band valley is considered separately and the valence band edges are given by $E_{v,i} = E_c + E_{g,i}$. Valence band mixing through $k \cdot p$ is available as an option but only affects momentum matrix elements and dispersion relations.

Parameters

- **use_strained_bandgap** determines whether or not the strained bandgap is used inside the band offset calculations for quantum wells.
- **bulk_strain_exist** uses the strained bandgap in bulk and bulk active layers to position the valence band(s).
- **use_hh_as_bandgap** controls whether the HH band is used as the single band edge for transport in passive layers. If this is turned off, the smallest bandgap will be used as the band edge. This parameter will affect hole transport by changing the effective barrier height in the thermionic emission model.

Examples

Recommended setting to model a strained electron blocking layer of AlGaN in a LED; adjustment of the affinity macro to include the hydrostatic shift is also required.

```
wurtzite_offset_model bulk_strain_exist=yes use_hh_as_bandgap=no
```

22.785 x_position

parameter	data type	values [defaults]
label	char	[void]
x	real	

x_position is used by the program to refer to a x-coordinate using a label.

Parameters

- **label** is a position label at a specific x-coordinate in microns.
- **x** is the x-coordinate of the point.

Examples

```
x_position label=mid_mqw x=0.0051
```

This example labels a x-position 0.0051 μm as “id_mqw" for use in other statements.

22.786 y_position

parameter	data type	values [defaults]
label	char	[void]
y	real	

y_position is used by the program to refer to a y-coordinate using a label.

Parameters

- **label** is a position label at a specific y-coordinate in microns.
- **y** is the y-coordinate of the point.

Examples

```
y_position label=mid_mqw y=0.0051
```

This example labels a y-position 0.0051 μm as “id_mqw" for use in other statements.

22.787 young_modulus

young_modulus defines Young's modulus for the acoustic wave propagation model in SAWAVE.

The parameters for this statement are the same as for all other material statements. See **material_par** in section 22.456 for examples and further details.

22.788 z_structure

parameter	data type	values [defaults]
link_taper_contact	char	[yes] (no)
cylindrical	char	[no] yes
uniform_zseg_from	real	(um)
uniform_zseg_to	real	(um)
uniform_length	real	(um)
taper_length	real	[0.] (um)
xp1_size	real	[0.] (um)
xp2_size	real	[0.] (um)
mesh_ratio	real	[1.]
shift_center	real	[0.] (um)
cylindrical_origin	real	[0.](um)
single_plane_zdim	real	(um)
zseg_num	intg	[1]
zplanes	intg	[1]

The statement **z_structure** is used to define the electrical mesh structure in the z-direction for a particular region (z-segment) with uniform material composition and which share the same material parameters.

Note that many taper commands previously defined here have been relocated to the **taper_between_segments** statement.

We also note that electrical current may or may not flow between mesh planes: this setting is controlled in **3d_solution_method**. This setting may be of interest in edge-emitting lasers where the mesh planes are coupled by the round-trip propagation of the light and longitudinal current flow is negligible.

A more complete discussion on the 3D modeling scheme used by Crosslight software tools may also be found in Sec. 6.3.

Parameters

- **link_taper_contact** indicates whether contacts in different segments are joined together if there is tapering between segments. This statement takes effect only when two electrodes have the same contact number.

  This parameter only takes effect when the current is not allowed to flow between xy planes. For details, see **3d_solution_method**. If the current is allowed to flow between planes, then the connection is mandatory.

- **cylindrical** would set the current segment to be in cylindrical system.

- **uniform_zseg_from** and **uniform_zseg_to** are used to specify the positions of the uniform segment along the longitudinal direction. These two positions can be the same, in which case the length of the segment is zero. This is not uncommon when using tapers.

  Note that taper connections are made in the space between two uniform segments: there may therefore be gaps between the "from" and "to" values of two successive segments. However, the first and last segments of the device must always be uniform segments, even if their length is zero.

- **zseg_num** is the z-segment number.

- **uniform_length** defines the length of the uniform segment; this will override the **uniform_zseg_from** and **uniform_zseg_to** if need be. In general, only one declaration style for segment lengths should be used, in order to simplify the input.

- **xp1_size** and **xp2_size** are small offsets applied to the mesh plane position at the beginning and end, respectively, of the segment. This is needed to properly define heterojunctions since mesh points can only belong to one material number in the Crosslight discretization scheme.

- **mesh_ratio** controls the distribution of mesh planes in the z-segment. See Fig. 22.22 for details.

  shift_center is a related parameter which applies for negative mesh ratios and symmetric mesh distributions. This parameter is used to adjust the center of the distribution.

- **cylindrical_origin** is the origin of the cylindrical system, if applicable.

- **zplanes** is the number of mesh planes (longitudinal mesh) in the z-segment; to properly define a 3D volume, at least 2 mesh planes must normally be present.

  A single mesh plane inside a segment may be used in only 2 cases:

- in the case of tapers, where a single mesh plane with zero thickness is used to sample the longitudinal properties of the device at a specific position. In this case, multiple segments are expected, in order to define a 3D volume using multiple mesh planes from these different segments.
- if **single_plane_zdim** or **taper_length** are used to define the segment length. In this case, the material properties are uniform in the segment and the 3D volume is obtained by extruding the 2D mesh plane with this length.

Examples

The first example is a standard edge-emitting DFB laser, electrically uniform with 7 mesh planes:

```
3d_solution_method 3d_flow=yes z_connect=no
z_structure uniform_length=500.0 zplanes=7 zseg_num=1
load_mesh mesh_inf=inp13.msh
```

However, this particular DFB laser has a $\frac{\lambda}{4}$ phase shift in the middle so optically, this electrical segment is split into two optical sections:

```
$ Quarter-wave phase shift: 90 degrees=0.5*pi
section length=250e-6 kappa_real=2e3 &&
  sec_num=1 mesh_points=10 phase_shift=0.5
section length=250e-6 kappa_real=2e3 &&
  sec_num=2 mesh_points=10
```

The second example defines a taper between z=0 and z=200m followed by a uniform segment. There is no mesh inside the taper region but mesh triangles are projected along taper lines inside the finite volume discretization of the Drift-Diffusion equations. Additional segments (with their own .layer, .geo and .msh files) inside the taper regions would enhance the accuracy of this simulation by more accurately sampling the lateral variation in the structure.

```
3d_solution_method 3d_flow=yes z_connect=yes
z_structure uniform_zseg_from=0. uniform_zseg_to=0. &&
  zseg_num=1 zplanes=1

taper_between_segments from_segment=1 xpoint_from=1 xpoint_to=6

z_structure uniform_zseg_from=200. uniform_zseg_to=400. &&
```

```
  zseg_num=2 zplanes=2

load_mesh mesh_inf=gaas2e1.msh zseg_num=1
load_mesh mesh_inf=gaas2e2.msh zseg_num=2
```

22.789 zdir_cx

parameter	data type	values [defaults]
type	char	[well], barrier
xy_ref	realx2	[1. 2.] (μm)
zseg	intg	
mater	intg	1

zdir_cx is used to define a complex MQW region in the z direction.

Parameters

- **type** identifies the current z-segment as being either the barrier or well of the complex MQW region.
- **mater** is the material number for the quantum-coupled region on this plane. If this statement is used in the .layer file where materials is not available, the quantum coupled region is identified with the (x,y) coordinates of **xy_ref** instead.
- **zseg** is the z-segment number where this commands applies; it may be generated automatically for the .sol if this statement is originally defined in the .layer file.

Examples

```
begin_zdir_complex num_segment=   9
zdir_cx zseg=  16 type=barrier mater=   3
zdir_cx zseg=  17 type=barrier mater=   3
zdir_cx zseg=  18 type=well xy_ref=  0.1000E+01  0.5000E-01 mater=   4
zdir_cx zseg=  19 type=well xy_ref=  0.1000E+01  0.5000E-01 mater=   4
zdir_cx zseg=  20 type=well xy_ref=  0.1000E+01  0.5000E-01 mater=   4
zdir_cx zseg=  21 type=well xy_ref=  0.1000E+01  0.5000E-01 mater=   4
zdir_cx zseg=  22 type=well xy_ref=  0.1000E+01  0.5000E-01 mater=   4
```

```
zdir_cx zseg=  23 type=barrier mater=   3
zdir_cx zseg=  24 type=barrier mater=   3
end_zdir_complex
```

22.790 zdir_light_source

parameter	data type	values [defaults]
shape	char	[rectangle], circle
xrange	realx2	[-1.e8 1.e8] (μm)
yrange	realx2	[-1.e8 1.e8] (μm)
gaussian_tail	real	[0.1] (μm)
circle_center	realx2	[0.0 0.0] (μm)
circle_radius	real	[1.0] (μm)

This statement is used in conjunction with **light_power** to define an optical pumping source that is roughly perpendicular to the xy mesh planes in a 3D simulation.

zdir_light_source light source specifies only the profile of the input light. All other properties are controlled by **light_power** just like in a 2D simulation.

Parameters

- **shape** is the shape of the light spot on the device. Within that spot, the power density is uniform.
- **xrange** and **yrange** determine the extend of the light spot in the rectangle model.
- **gaussian_tail** describes how quickly the light decays when outside the light spot. This is modeled in the same way as the decay in the **doping** statement.
- **circle_center** and **circle_radius** determine the extend of the light spot in the circle model.

Examples

```
light_power  spectrum_file=solar.am0  light_dir=+z
zdir_light_source shape=circle gaussian_tail=0.1 &&
      circle_center=(65. 65.) circle_radius=60.
```

22.791 zener

parameter	data type	values [defaults]
balance_zero_bias	char	[yes]
tun_mass	real	
xrange	realx2	
yrange	realx2	

zener is used to activate the interband tunneling (or Zener tunneling as in Zener diodes) model. Since this statement is useful for heavily doped reverse junctions, it may be useful to use this statement together with the impact ionization model.

Note that this model converts the tunneling current into a local generation term: the formula involved is only applicable for a reverse junction and does not model the negative resistance that can occur under forward bias.

For forward-biased tunnel junctions, see the **tunnel_junc** statement. A full discussion on theory can also be found in Sec. 9.3.

Parameters

- **balance_zero_bias** balances out the tunneling current with thermal current at zero bias. This may be set to "no" in some cases for the purpose of achieving convergence at low bias voltages.

- **tun_mass** is the Zener interband tunneling mass to be set by the user. If it is not specified, the combined effective mass is used as in the following formula: $2m_e m_h/(m_e + m_h)$.

- **xrange** and **yrange** define the tunneling range using absolute coordinates.

Examples

```
$ Just one word:
zener
$ This will enable the interband tunneling model to
$   use the standard combined effective mass.
```

22.792 zero_doping

parameter	data type	values [defaults]
mater_label	char	
mater	intg	[1]
macro_name	char	[void]

The statement **zero_doping** forces the doping to be zero in a particular material.

Parameters

- **mater** is the material number affected by this statement. This parameter is ignored if **macro_name** is used.
- **mater_label** may be used instead of **mater** if a label has previously been defined as an alias.
- **macro_name** indicates all materials using this macro name be set to zero doping impurities.

22.793 zincblende_offset_model

parameter	data type	values [defaults]
use_strained_bandgap	char	[yes]
partition_hydro_term	char	[yes]

The this command is used to control various aspects of the band offset model for zincblende materials; to understand it properly, a few key aspects of the band alignment rules in Crosslight should be reviewed:

- 1 Bulk layers (active and passive) directly define the conduction band position using the **affinity** statement. No strain shifts are applied to this value unless they are deliberately included the affinity declaration (macro file).

  For passive layers, only a single band is ever used for transport so all valence bands are combined, with a reduced mass accounting for the HH and LH bands. The valence band edge is thus positioned as $E_v = E_c + E_g$.

For bulk active layers, all of the valence band edges are explicitly considered so using the effective bandgap for each valley, the valence bands are positioned as $E_{v,i} = E_c + E_{g,i}$.

- 2 Quantum wells position the conduction band relative to the barrier based on the **band_offset** value; this can be overridden by certain commands such as **use_bulk_affinity** and **band_discont**. Various authors define the band offset in different ways which may or may not include the strain; see Sec. 10.1 for more information on this topic.

  For a quantum well, each band valley is considered separately and the valence band edges are given by $E_{v,i} = E_c + E_{g,i}$. Valence band mixing through $k \cdot p$ is available as an option but only affects momentum matrix elements and dispersion relations.

Parameters

- **use_strained_bandgap** determines whether or not the strained bandgap is used inside the band offset calculations for quantum wells.
- **partition_hydro_term** controls whether or not the hydrostatic shift is split between the conduction and valence band. If this option is disabled, the hydrostatic shift is applied on the valence band alone.

22.794 zplane_label

parameter	data type	values [defaults]
label	char	
hline nearest_z	real	(μm)
zplane	intg	

This statement is used to mark a specific z coordinate for later use. Unlike **zplane_position**, the coordinate must explicitly match the position of a mesh plane.

Parameters

- **label** defines a variable name that can be reused in other commands to refer to this position.

- **zplane** is the z-mesh plane number where the labeled is applied. This number can also be found automatically by specifying a numeric value in **nearest_z**; the software will attempt to locate the mesh plane closest to this position.

22.795 zplane_position

parameter	data type	values [defaults]
label	char	
location	char	[top]
delta_z_from_bottom	real	[-9999.] (μm)
delta_z_from_top	real	[-9999.] (μm)

This statement is similar to **layer_position** and is used to mark a specific z coordinate for later use. Unlike **zplane_label**, the position defined by this statement does not need to match that of a z-mesh plane which makes it useful for plotting purposes.

All the parameters from this statement are analogous to those of **layer_position**. The main difference is that all positions are defined relative to the preceding **z_structure** statement.

22.796 zsegment_setting

parameter	data type	values [defaults]
limit_gain	char	[no]
zseg_num	intg	[1]

zsegment_setting is used to set some model parameters for a specific z-segment.

- **limit_gain** indicates whether optical gain is limited by the gain interpolation table and disallow any extrapolation. This parameter affects package PICS3D only.
- **zseg_num** is the z-segment affected by this command.

PARS OPTICS GROUP

Part VI

APPENDICES

PARS OPTICS GROUP

Appendix A

WAVE EQUATION FOR PERFECTLY MATCHED LAYER

We shall derive the wave equation for the PML starting from the Maxwell equations:

$$\begin{aligned} \nabla \cdot \varepsilon[\Lambda]\overrightarrow{E} &= 0 \\ \nabla \cdot \mu[\Lambda]\overrightarrow{H} &= 0 \\ \nabla \times \overrightarrow{E} &= -j\omega\mu[\Lambda]\overrightarrow{H} \\ \nabla \times \overrightarrow{H} &= j\omega\varepsilon[\Lambda]\overrightarrow{E} \end{aligned} \tag{A.1}$$

Based on the 3rd and 4th equations above, we obtain the wave equation in the following form:

$$[\Lambda]^{-1}\nabla \times [\Lambda]^{-1}\nabla \times \overrightarrow{E} - k^2\overrightarrow{E} = 0 \tag{A.2}$$

Using a, b, c to define the $[\Lambda]$ tensor, we can make the following expansion and derivation:

$$\begin{aligned} [\Lambda]^{-1}\nabla \times [\Lambda]^{-1}\nabla \times \overrightarrow{E} &= \frac{1}{a}\left[\frac{1}{c}\left(\frac{\partial^2 E_y}{\partial x \partial y} - \frac{\partial^2 E_x}{\partial y^2}\right) - \frac{1}{b}\left(\frac{\partial^2 E_x}{\partial z^2} - \frac{\partial^2 E_z}{\partial x \partial z}\right)\right]\overrightarrow{i} \\ &+ \frac{1}{b}\left[\frac{1}{a}\left(\frac{\partial^2 E_z}{\partial y \partial z} - \frac{\partial^2 E_y}{\partial z^2}\right) - \frac{1}{c}\left(\frac{\partial^2 E_y}{\partial x^2} - \frac{\partial^2 E_x}{\partial x \partial y}\right)\right]\overrightarrow{j} \\ &+ \frac{1}{c}\left[\frac{1}{b}\left(\frac{\partial^2 E_x}{\partial x \partial z} - \frac{\partial^2 E_z}{\partial x^2}\right) - \frac{1}{a}\left(\frac{\partial^2 E_z}{\partial y^2} - \frac{\partial^2 E_y}{\partial y \partial z}\right)\right]\overrightarrow{k} \end{aligned} \tag{A.3}$$

For anisotropic PML with interface normal to x-axis, the coefficients should be re-

lated by $c = b = 1/a$. We may simplify the above equation as follows:

$$\begin{aligned}
[\Lambda]^{-1}\nabla \times [\Lambda]^{-1}\nabla \times \overrightarrow{E} &= \left[\left(\frac{\partial^2 E_y}{\partial x \partial y} - \frac{\partial^2 E_x}{\partial y^2}\right) - \left(\frac{\partial^2 E_x}{\partial z^2} - \frac{\partial^2 E_z}{\partial x \partial z}\right)\right]\overrightarrow{i} \\
&+ \left[\left(\frac{\partial^2 E_z}{\partial y \partial z} - \frac{\partial^2 E_y}{\partial z^2}\right) - \frac{1}{c^2}\left(\frac{\partial^2 E_y}{\partial x^2} - \frac{\partial^2 E_x}{\partial x \partial y}\right)\right]\overrightarrow{j} \\
&+ \left[\frac{1}{c^2}\left(\frac{\partial^2 E_x}{\partial x \partial z} - \frac{\partial^2 E_z}{\partial x^2}\right) - \left(\frac{\partial^2 E_z}{\partial y^2} - \frac{\partial^2 E_y}{\partial y \partial z}\right)\right]\overrightarrow{k} \quad \text{(A.4)}
\end{aligned}$$

It is useful to consider the following expression:

$$\begin{aligned}
\frac{1}{c}\nabla(\nabla \cdot [\Lambda]\overrightarrow{E}) &= \nabla\left(\frac{1}{c^2}\frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} + \frac{\partial E_x}{\partial z}\right) \\
&= \overrightarrow{i}\frac{\partial}{\partial x}\left(\frac{1}{c^2}\frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} + \frac{\partial E_x}{\partial z}\right) \\
&+ \overrightarrow{j}\frac{\partial}{\partial y}\left(\frac{1}{c^2}\frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} + \frac{\partial E_x}{\partial z}\right) \\
&+ \overrightarrow{k}\frac{\partial}{\partial z}\left(\frac{1}{c^2}\frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} + \frac{\partial E_x}{\partial z}\right) \quad \text{(A.5)}
\end{aligned}$$

Based on the first Maxwell equation, the above equation should be zero. Thus, we may subtract it from Eq. A.4 so that the cross terms may be canceled out. We obtain the following:

$$\begin{aligned}
[\Lambda]^{-1}\nabla \times [\Lambda]^{-1}\nabla \times \overrightarrow{E} &= -\left(\frac{1}{c^2}\frac{\partial^2 E_x}{\partial x^2} + \frac{\partial^2 E_x}{\partial y^2} + \frac{\partial^2 E_x}{\partial z^2}\right)\overrightarrow{i} \\
&- \left(\frac{1}{c^2}\frac{\partial^2 E_y}{\partial x^2} + \frac{\partial^2 E_y}{\partial y^2} + \frac{\partial^2 E_y}{\partial z^2}\right)\overrightarrow{j} \\
&- \left(\frac{1}{c^2}\frac{\partial^2 E_z}{\partial x^2} + \frac{\partial^2 E_z}{\partial y^2} + \frac{\partial^2 E_z}{\partial z^2}\right)\overrightarrow{k} \quad \text{(A.6)}
\end{aligned}$$

Or simply:

$$[\Lambda]^{-1}\nabla \times [\Lambda]^{-1}\nabla \times \overrightarrow{E} = -\left(\frac{1}{c^2}\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\overrightarrow{E} \quad \text{(A.7)}$$

The wave equation in the PML can be thus written in the following form:

$$\left(\frac{1}{c^2}\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\overrightarrow{E} + k^2\overrightarrow{E} = 0 \quad \text{(A.8)}$$

Appendix B

MATERIAL PARAMETERS

B.1 Introduction

Most material parameters implemented in the simulation software are well established data taken from Refs. such as [1], [153] and [154]. They are specified outside the program in the form of input statements. Since there are a large number of semiconductor parameters for a device, these material statements are collected together as macros.

To ensure a rapid update of the most recent material parameters, the material macro library is in a simple editable text format. Should the user have better knowledge of the material parameters than those listed in the macro library, he/she is encouraged to revise the library or override the parameters in the macro by re-issuing an input material statement after the macros have been loaded. It is also possible to define an entirely new macro in a separate text file in the simulation directory: the program can make use of this custom macro file with the **use_macrofile** statement.

In this appendix, we only provide a full listing of a few typical material macro parameters for the purpose of illustration. For a full listing of all macros, consult the "crosslight.mac" and "more.mac" files in the installation directory. These files are the default macro database and should usually not be edited by the user since they affect all simulations. Instead, use custom macro files or override input statements as described above.

A full list of macros is also visible when using the LayerBuilder and GeoEditor GUI programs to define the materials in a particular layer or polygon.

Note that certain macros, especially for ternary/quaternary III-V materials, are designed for the case where the material is lattice-matched to a particular substrate (GaAs or InP). Make sure to read the header section of each macro to ensure you are using the correct macro and material parameters for your simulation.

B.2 Rules for macros

The rules for macros are detailed in the header of the crosslight.mac file. For the sake of convenience, here is a copy.

```
$ IMPORTANT: Please do not use any invisible characters such as tabs in macro
$*****************MATERIAL MACRO*********************************
$
$      Copyright (c) 1995-present Crosslight Software Inc..
$                 All rights reserved.
$
$                  What Is A Macro
$
$  A macro is a collection of input statements (or commands).
$  A statement starts with a keyword followed by parameters
$  specified with "=".
$  A statement for material parameters overrides a
$  previous statement with the same keyword.
$  To make changes of the parameters in a macro, just re-issue
$  the statement with different parameters after the macro call
$  in the input file.  You can also change the parameters in the
$  macro directly.
$
$  For example, if you are unhappy with the bandgap in gaas macro,
$  you can do the following:
$    load_macro name=gaas mater=#m
$    band_gap value=1.425 mater=#m
$
$
$                  Two Types of Macros
$
$  For device simulation, you may need two types of macros:
$
$  1) Bulk material macros.  These are given lower case macro names
$     such as "algaas".  They are referred to by "load_macro" in the
$     input files.   You must specify bulk material macro for
$     all material layers.  They specify general material
$     parameters such as bandgap, mobility, index, etc.
$
$  2) Active layer macros.  These are given mixed case macro names
$     such as "AlGaAs". These are only needed for active layers in a
$     laser diode or quantum well layers in other devices.  Quantum
$     subbands and/or optical processes are considered in these layers.
```

```
$     If there are overlaps with 1), the parameters (such as bandgap)
$     in active layer macros will override those in 1). Active layer
$     macro with prefix cx- (such as cx-InGaAsP/InP)
$     are "complex active layers" which may be used to define
$     active or quantum regions of complex structures.
$
$  You must use both "bulk material macro" and "active layer macro"
$  if you have an active region or a quantum region.
$
$
$                        Syntax
$
$  Comments start with "$".  A line to be continued terminates
$  with "&&".  Symbol definition within a function terminates with ";".
$  A statement starts with a keyword followed by parameters
$  specified with "=".  Space, ",", "(", ")", "[", "]", "{" and
$  "}" with a statement are ignored by the program.  The length
$  of a statement line must not exceed 80 characters.  The tab key is
$  not to be used in any statements.
$
$  A math function is declared using variation=function and
$  followed in the next few lines by
$
$  function(var1,var2,var3,...)
$  for logical_expression_1
$  complete_function_definition_1
$  for logical_expression_2
$  complete_function_definition_2
$  for logical_expression_3
$  complete_function_definition_3
$  ...
$  for else
$  complete_function_definition_N
$  end_function
$
$  The logical_exression takes the format
$  expr1<expr2 or expr1<expr2<expr3
$  It also supports one of | (logical OR) or & (logical AND) operator.
$  to use "for else" as a case indicator. The following lines are legal:
$
$  function(x,y,xlam)
$  for 1.24/xlam<bulk_xfunc1
$  complete_function_definition_1
```

```
$  for 0.5<y
$  complete_function_definition_2
$  for y>0.1&xlam<0.8
$  complete_function_definition_3
$  for x<0.8|xlam<0.8
$  complete_function_definition_4
$  for else
$  complete_function_definition_5
$  end_function
$
$  Please note that ">" really means "greater than or equal".  Similarly
$  "<" means "less than or equal".  For continuous
$  numerical functions, there is no difference between "greater than or equal"
$  and simply "greater than".
$
$  The case indicator above separates the function into branches
$  of complete function definitions (like having different functions).
$
$  Within each branch of complete function definition, a set of
$  new and more convenient logical switches can be used in the following form
$
$  if(expression1) then
$  else if(expression2) then
$  else if(expression3) then
$  ...
$  else
$  endif
$
$  The "then" word is optional, so is the "if" following the "else".
$  The program looks for expressions within the
$  bracket.   The following format is equivalent to the above.
$
$  if(expression1)
$  else (expression2)
$  else (expression3)
$  ...
$  else
$  endif
$
$  It is also possible to use an if-else-endif branching within another
$  if-else-endif.  The hierarchy may reach 99 and within
$  each hierarchy, there can be 29 branches.
$  For example the following complicated logical branches are
```

```
$  supported:
$
$  if(adfa.eq.k) then
$  some_definition_lines
$
$   if(5<x*y)
$   some_definition_lines
$   else if(wavelength>1.9)
$   some_definition_lines
$   endif
$
$  some_definition_lines
$  else if (bulk_xfunc1>2.5) then
$  some_definition_lines
$  endif
$
$
$  Please note that a complete function definition can not
$  exceed 3000 characters.  If you need to construct a function
$  with many logical branches of long math expressions,
$  it is recommended that you use the conventional
$  "for" logical branching method.
$
$  The logical expressions in "for" and "if-then-else" must
$  not exceed 80 characters.  The math expressions follow the
$  same syntax as for function definition detailed as follows.
$
$  For a function definition, there can be more than one
$  math expressions.  The "=" symbol is used to define intermediate
$  variables. Expression without "=" is interpreted as the returned
$  function value and causes the evaluation procedure to exit the function.
$  All expressions should be separated by ";" or "end-of-line".
$  An expression longer than 80 characters will be jointed
$  by the next line regardless of existence of "&&" at the end of the line.
$
$  The mathematical operators +, -, *, /, **
$  follow those of C/FORTRAN [** means pow() or power() in C] languages.
$  All intrinsic mathematic functions appearing in the
$  FORTRAN language, such as sin(), cos(),
$  sqrt(), log(), log10(), exp(), etc., can be used without modification.
$
$                     Tables
$
```

```
$  In addition to analytical functions, parameters can also be specified
$  by numerical tables expressed in the form of an n-dimensional matrix:
$  para(j1,j2,...,jn).  The index in each dimension corresponds to an
$  independent variable (such as composition or temperature).
$  The change of such a variable with the index is not necessarily of equal
$  interval but must be in ascending order.  The table must be presented in
$  a column-wise manner and the left most index changes the fastest.
$  Blank lines may be inserted anywhere inside the table.  For example:
$
$  electron_mass variation=table
$  table(x,y)
$    x(1)  y(1)  matrix(1,1)
$    x(2)  y(1)  matrix(2,1)
$    x(3)  y(1)  matrix(3,1)
$
$    x(1)  y(2)  matrix(1,2)
$    x(2)  y(2)  matrix(2,2)
$    x(3)  y(2)  matrix(3,2)
$
$  end_table
$
$
$                   Units
$
$  Dimensions are in micron meters.  Band parameters and
$  potentials are in eV or volts.  All others are in MKS units.
$  For example mobility is in m^2/(volt*sec), band_gap in eV
$  and dopant concentration in 1/m^3.
$
$               Symbol Definition
$
$  max_electron_mob        Maximum dopant dependent elec. mobility.
$  min_electron_mob        Minimum dopant dependent elec. mobility.
$  electron_ref_dens       Reference impurity density in
$                          Dopant dependent elec. mobility function
$  alpha_n                 exponent in dopant dependent elec. mobility
$                          function.
$  max_hole_mob            Corresponding symbol for holes.
$  min_hole_mob            ...
$  hole_ref_dens           ...
$  alpha_p                 ...
$  tau_energy              Energy relaxation time in hydrodynamic model.
$  lifetime_n              Minority n-carrier (electron) life time.
```

```
$                          Deep trap specification will override this.
$  lifetime_p              Minority p-carrier (hole) life time.
$                          Deep trap specification will override this.
$  radiative_recomb        Radiative recombination constant.
$  other symbols           Obvious.
$
$
$              Special Symbols, Temperature, Doping, etc.
$
$  "temper" is reserved as a variable for the lattice temperature
$  when constructing a material macro.  The unit is degrees Kelvin.
$  If you use any other symbol for temperature within a macro,
$  it may not work for the thermal modeling option.
$  Please note that in a simulation
$  involving setting the environmental temperature, it is not
$  sufficient to set the temperature parameter in a material
$  macro.   You must also use the "temperature" statement.
$  The temperature parameter in a macro is used to evaluate the
$  material properties (such as bandgap) at a particular temperature,
$  while the "temperature" statement is used to model the
$  carrier statistics at that temperature.
$
$  Similarly, "doping_n", "doping_p", and "trap_1", "trap_2", ..., "trap_9"
$  are reserved variable symbols in a function definition.  The units are
$  in 1/m**3.  "wavelength" is another reserved variable with unit of
$  micron meters.
$
$  # is used together with a symbol to denote a numerical value
$  to be supplied by the user. For example, in
$
$  load_macro name=gaas mater=#m
$
$  #m is used to mean a numerical value denoting the material
$  number in a device.  For example, if your device has GaAs bulk
$  material as material number 2, then the above statement should be
$  written as
$
$  load_macro name=gaas mater=2
$
$
$                  Macro Styles and Function Formats
$
$  An advanced feature of Crosslight Software macro is its ability
```

```
$  to define mathematical functions external to the simulation
$  program.  Such functions are evaluated dynamically at run-time.
$
$  Currently, there are two kinds of macros with different formats of
$  functions:  a) the older "fix-arguement-style" macros and
$  b) the more recent "free-arguement-style" macros.  The older macros
$  are included here for compatibility reason and the newer formats are
$  highly recommended.  The two formats are explained as follows.
$
$  a) Fix-Arguement-Style Macros
$
$  The rules for function variables/arguements are rather strict.
$  If a function is used for a physical quantity (presented as
$  a statement), there should be exactly the same number of
$  of function variables as that used in the "load_macro" or
$  "get_active_layer" statement.  For example,  if you use
$  (x,temper) as the variables in "load_macro", all functions
$  within the macro must have these two variables even if
$  all of them do not use the two variables.  When calling such a macro,
$  the following can be used:
$
$  load_macro name=algaas.temp var1=0.3 var2=300
$
$  Since all functions of such a style have the same number of
$  arguments, the program knows which value supplied belongs to which
$  argument in a function.  Please note that when calling these kinds
$  of macros,  the user must supply all values of the arguments even
$  if they are reserved symbols such as "temper", "doping_n", etc.
$
$  b) Free-Argument-Style Macros
$
$  The use of free-style format is more based on the need of variables.
$  A function passes and uses as many function arguments as it needs.
$  For example, the bandgap can pass and use both the composition
$  and temperature as (x,temper).  But for the effective mass, only
$  the composition is needed and passed as (x).  The evaluation of such
$  a macro is based on symbol matching, i.e., the user must specify
$  both the symbol and the value of the symbol when calling the macro,
$  such as in the following call.
$
$  load_macro name=algaas.temp var1=0.3 var2=300 &&
$    var_symbol1=x var_symbol2=temper
$
```

```
$  The user may skip the specification of reserved internal arguments
$  such as "temper", "doping_n", etc., because the program already
$  knows their values.  For example, the following may be used instead of
$  the above.
$
$  load_macro name=algaas.temp var1=0.3 var_symbol1=x
$
$
$                  Searching For a Macro
$
$  The macro files can contain both the older and newer style macros
$  with the same macro name.  The search of the correct version depends
$  on how the macro is called.  If a macro does not require
$  any user-supplied macro variables, the simulator will use the first
$  macro it encounters in the search. For macro requiring user-supplied
$  variables (or function arguments), if "var_symbol" is not specified
$  in a macro call, older style macro is assumed and user must strictly
$  align all variables in a macro call.  The user is encouraged
$  to convert to the newer style because all future new macros will
$  be created and maintained in the newer free-style.
$
$  Please also note that the older style (fix-argument-style) may be
$  regarded as a special case of free-style macro with many un-used
$  function variables listed in all macro functions.  Or in another word
$  the older macro can be loaded also with a complete listing of
$  "var_symbol".  In fact when the program detects a "load_macro" or
$  "get_active_layer" without any "var_symbol", it will search all the
$  macro functions and supply the necessary "var_symbol".  In the case
$  of a macro call fits more than one macros in the macro library, the
$  simulator will always use the first one it encounters.
$
$  For the free-style macros to be recognized by the graphic user
$  interface (GUI) such as LayerBuilder and GeoEditor, it must be labeled
$  as [free-style] in the remarks before the macro.  Otherwise, it will be
$  treated as an older style macro.
$
$
$                  Mobility Parameter Specifications
$
$  In Crosslight device simulator, the dopant dependent low field mobility
$  takes the following form:
$
$  mu_low_field=mu_min+(mu_max-mu_min)/(1+(total_doping/ref_dens)**alpha)
```

```
$
$  The current version of the simulator supports two versions of
$  parameter specifications.   The older version is to specify the
$  following  four parameters for electrons:
$  max_electron_mob, min_electron_mob, electron_ref_dens, and alpha_n.
$  The program computes the above for each mesh point and average them
$  in mid-point between two nodes.  Then the mid-point quantities of the
$  above is used in the dopant dependent formula at every iteration.
$
$  To speed up the computation, a more efficient version of parameter
$  specification is to directly evaluate the dopant dependence for
$  for statement of "electron_mobility", as in the following example.
$
$  electron_mobility variation=function
$  function(doping_n,doping_p,trap_1)
$  mu_max=0.14;
$  mu_min=0.0055;
$  ref_dens=1.07d23;
$  alpha=0.73;
$  total_doping=doping_n+doping_p+trap_1;
$  mu_min+(mu_max-mu_min)/(1+(total_doping/ref_dens)**alpha)
$  end_function
$
$  Speed-up is achieved because we need to evaluate the formula
$  once only at the time of processing the macro data.
$
$  The two versions of parameters specification are equivalent in theory
$  but may result in some difference if the mesh is rough because of the
$  non-linearity of the dopant dependent formula.  Also, difference in
$  number of trap types may also make a difference.  The older version
$  of specification automatically takes into account all trap densities
$  while the user need to explicitly specify the number of traps to
$  be used in the dopant dependent formula.
$
$
$  use_macro_file macro1=my_old_macro1 macro2=my_old_macro2
$
$  The simulation software will first search macros in the above
$  statement before searching the default crosslight.tab/more.tab.
$  Please note that that included macros must be placed in the
$  same directory as the input files.
$
$
```

```
$     Crosslight Quaternary Parameter Interpolation Scheme
$
$ For quaternary material such as In(1-x)Ga(x)As(y)P(1-y), experimental
$ data for arbitrary (x,y) normally do not exist.  However, data are available
$ for ternaries [such as In(1-x)Ga(x)As] and for the lattice matched line:
$ fm(xm,ym)=0.    Therefore, any reasonable interpolation scheme must
$ satisfy experimental data for all the ternaries and at the matched line.
$
$ Crosslight macros use the following interpolation scheme:
$
$ Let us take the bandgap as an example:
$ Given bandgap of matched line (xm,ym): Eg0_match(ym)
$ and measured ternary bandgaps, we interpolate as follows.
$ If (x,y) is within the left side of the matched line (xm,ym):
$
$ Eg0=Eg0_bilin(x,y)+(x/xm)*(Eg0_match(y)-Eg0_bilin(xm,y))
$
$ If (x,y) is within the right side of the matched line (xm,ym):
$
$ Eg0=Eg0_bilin(x,y)+[(1-x)/(1-xm)]*(Eg0_match(y)-Eg0_bilin(xm,y))
$
$ where Eg0_bilin(x,y) is bi-linear interpolation from ternaries as proposed
$ in the paper of Adachi,
$ [Adachi,S.  "Material parameters of In(1-x)Ga(x)As(y)P(1-y) and related
$  binaries." J. Appl. Phys. Vol. 53 No. 12, Dec. 1982, pgs 8775-92.]
$
$ The Crosslight formula ensures that the unstrained bandgap
$ agrees with experiment at ternaries and at the matched line (xm,ym).
$ For any other composition, we draw horizontal line across
$ (x,y) and linearly interpolate given the three known points:
$ (0,y), (xm,y), (1,y)
$
$         Supported Intrinsic Math Functions
$
$ Math functions are directly taken from Fortran90 and accuracy of
$ double precision is already used.  Please note that the following
$ symbols are reserved for math functions and these should not be used
$ for user-defined variables.  Also, version 2009 and later supports
$ multiple variable instrinsic functions, such as atan2(x1,x2)
$ and max(x1,x2,...).
$
$  function-symbol   equivalent double-precision version    remark
$  log                dlog
```

```
$  log10              dlog10
$  exp                dexp
$  asin               dasin
$  acos               dacos
$  atan               datan
$  sin                dsin
$  cos                dcos
$  tan                dtan
$  sinh               dsinh
$  cosh               dcosh
$  tanh               dtanh
$  erf                derf
$  erfc               derfc
$  sqrt               dsqrt
$  abs                dabs
$  atan2              datan2          2-variable function
$  max                                multiple-variable function
$  maxval                             same as max()
$  min                                multiple-variable function
$  minval                             same as min()
$
$         Band Offsets and Alignments
$
$  For bulk macro, the band alignment is solely determined by the
$  affinity parameter.  For active_layer macro, the band alignment
$  is determined by band_offset, band_discont, or band_discont_right.
$  band_discont and band_discont_right override band_offset.   Then
$  band alignment settings in active_layer macro override the affinity
$  setting in bulk macro.
$
$  For quantum well with strain in well and/or barrier, it may be confusing
$  to use band_offset since hydrostatic strain changes the bandgap.  Such
$  change may be partitioned between the conduction and
$  valence band with a certain ratio.
$
$  The shear strain may split the HH/LH/CH valence bands, making the
$  bandgap multi-valued.  Different use of band_offset is controlled by
$  the command zincblende_offset_model or wurtzite_offset_model, for
$  zincblende or wurtzite material, respectively.
$
$
$         Supported Internal and External Functions
$
```

```
$ Crosslight macro system supports user-defined external and internal
$ functions.   User-defined external function uses names like
$ bulk_xfunck (k=1,...,9) for bulk macros and ext_funck (k=1,...,9)
$ for active_layers.
$ They can be defined in the same way as other predefined functions
$ such as for band_gap and affinity, and supports symbolic math expressions
$ and tables.   The advantage is that they can be repeatedly used in
$ defining other quantities.  For example, one may define a temperature
$ dependent external function and repeatedly use it in band_gap, affinity,
$ mobility and other physical variables which may use the same temperature
$ dependent function.
$
$ The other type of function is "internal function", in the form of
$ intern_funck (k=1,...,9).   These are more flexible and are treated
$ in a manner as the math intrinsic functions such as sin() and cos().
$ They are user-defined within the same macro and can use both math
$ expression and tables.   The function arguements are completely independent
$ of variables passed when loading a macro.   The difference between
$ user-defined internal and external functions are as follows:
$ Function arguements for external functions must be passed from macro
$ loading.  For example, the argument "x" is passed through
$ load_macro name=algaas mater=   1 &&
$  var_symbol1=x  var1=  0.7100E+00
$ Also, temperature, doping concetration and other reserved variables are
$ automatically passed to it.  The user has no way of changing the
$ input arguements.   For internal functions, the input arguements
$ are supplied when being used within a function definition.  The input
$ arguements may or may not be related to macro loading.  Also, user
$ may operate on the input variables before passing it as internal
$ a function arguement.
$
$=============================================================
```

B.3 Zincblende passive macro

A typical passive macro for a zincblende material ($Al_xGa_{1-x}As$) is listed here.

```
$**********************************
$ macro algaas
$ for bulk Al(x)Ga(1-x)As
$ Lattice matched to GaAs
$ [free-style]
```

```
$ Temperature dependent version of macro algaas
$ Renamed from algaas.temp
$ Typical use:
$   load_macro name=algaas var1=#x mater=#m &&
$     var_symbol1=x
$ parameter_range x=[0 1]
$ parameter_range temper=[77 600]
$ parameter_range doping_n=[1.e20 1.e26]
$ parameter_range doping_p=[1.e20 1.e26]
$ parameter_range trap_1=[1.e18 1.e24]
$**********************************
begin_macro algaas
material type=semicond band_valleys=(1 1) &&
  el_vel_model=n.gaas hole_vel_model=beta
dielectric_constant variation=function
function(x)
13.1 - 3 * x
end_function

electron_mass variation=function
function(x)
for   0.<x<0.45
0.067 + 0.083 * x
for  0.45<x<1.
0.85 - 0.14 * x
end_function

hole_mass variation=function
function(x)
   (   ( 0.087 + 0.063 * x ) ** (3 / 2)
 + ( 0.62 + 0.14 * x  ) ** (3 / 2)  ) ** (2 / 3)
end_function

band_gap variation=function
function(x,temper)
for 0.<x<0.45
shift=-5.5e-4 * temper**2 /(temper+225)+9.4285712E-02 ;
1.424 + 1.247 * x +shift
for 0.45<x<1.
shift=-5.5e-4 * temper**2 /(temper+225)+9.4285712E-02 ;
1.9 + 0.125 * x + 0.143 *x * x  +shift
end_function
```

```
affinity variation=function
function(x,temper)
for 0<x<0.45
offset=0.6;
shift=-5.5e-4 * temper**2 /(temper+225)+9.4285712E-02 ;
4.07 - 0.748 * x-offset*shift
for 0.45<x<1.
offset=0.6;
shift=-5.5e-4 * temper**2 /(temper+225)+9.4285712E-02 ;
3.7964 - 0.14 * x-offset*shift
end_function

electron_mobility variation=function
function(x,temper,doping_n,doping_p,trap_1)
for 0<x<0.45
fac=(300/temper)**2.3;
mu_max=0.85 * exp(-18.516 * x ** 2 )*fac;
mu_min=0;
ref_dens=1.69d23;
alpha=0.436;
total_doping=doping_n+doping_p+trap_1;
mu_min+(mu_max-mu_min)/(1+(total_doping/ref_dens)**alpha)
for 0.45<x<1.
fac=(300/temper)**2.3;
mu_max=0.02*fac;
mu_min=0;
ref_dens=1.69d23;
alpha=0.436;
total_doping=doping_n+doping_p+trap_1;
mu_min+(mu_max-mu_min)/(1+(total_doping/ref_dens)**alpha)
end_function


hole_mobility variation=function
function(x,temper,doping_n,doping_p,trap_1)
dope=1.e22 ;
tfac1=(temper/300)**2.3 ;
tfac2=(temper/300)**1.5 ;
fac=1/( tfac1 + 1.6e-24*dope*tfac2 );
mu_max=(0.04 - 0.048 * x + 0.02 * x * x)*fac;
mu_min=0;
ref_dens=2.75d23;
alpha=0.395;
```

```
total_doping=doping_n+doping_p+trap_1;
mu_min+(mu_max-mu_min)/(1+(total_doping/ref_dens)**alpha)
end_function

beta_n value=2.
electron_sat_vel variation=function
function(x,temper)
for 0<x<0.45
fac=(300/temper)**2.3;
0.77e5 * (1 - 0.44 * x )*fac
for 0.45<x<1.
fac=(300/temper)**2.3;
8.e4*fac
end_function

beta_p value=1.
hole_sat_vel variation=function
function(temper)
dope=1.e22 ;
tfac1=(temper/300)**2.3 ;
tfac2=(temper/300)**1.5 ;
fac=1/( tfac1 + 1.6e-24*dope*tfac2 );
1.d5*fac
end_function

norm_field value=4.e5
tau_energy value=1.e-13
radiative_recomb value=1.d-16
auger_n value=1.5e-42
auger_p value=1.5e-42
lifetime_n value=1.e-7
lifetime_p value=1.e-7

real_index variation=function
function(x)
3.65-0.73*x
end_function

absorption value=0.
thermal_kappa value=46.
end_macro algaas
```

B.4 Zincblende active macro

A typical active macro for a zincblende material ($Al_xGa_{1-x}As$) is listed here. This is a QW macro with the barrier and well having a different Al%: note how the parameters depend on both the well and barrier compositions. Parameters from this macro will override values defined in the passive macros for the well and barriers if there is a conflict.

```
$ ******************************************************
$ acitve layer macro : Al(xw)Ga(1-xw)As/Al(xb)Ga(1-xb)As
$ [free-style]
$ xw, xb=Al comp. in well,  barrier, temper=well temperature
$ Typical use:
$   get_active_layer name=AlGaAs/AlGaAs  mater=#m &&
$     var1=#xw var2=#xb var_symbol1=xw var_symbol2=xb
$ parameter_range xw=[0 1]
$ parameter_range xb=[0 1]
$ parameter_range temper=[77 600]
$ ******************************************************
$
begin_active_layer AlGaAs/AlGaAs
$
layer_type type=unstrained_well valley_gamma=1 valley_l=4 &&
  valley_hh=1 valley_lh=1
$
eg0_well variation=function
function(xw,temper)
for   0.<xw<0.45
shift0=-5.5d-4*300.**2/(300.+225.) ;
shift=-5.5d-4*temper**2/(temper+225.) ;
1.424+1.247*xw+shift-shift0
for  0.45<xw<1.
shift0=-5.5d-4*300.**2/(300.+225.) ;
shift=-5.5d-4*temper**2/(temper+225.) ;
1.9+.125*xw+0.143*xw**2+shift-shift0
end_function
$
lband_well value=0.28
$
eg0_bar variation=function
function(xb,temper)
for   0.<xb<0.45
shift0=-5.5d-4*300.**2/(300.+225.) ;
```

```
shift=-5.5d-4*temper**2/(temper+225.) ;
1.424+1.247*xb+shift-shift0
for  0.45<xb<1.
shift0=-5.5d-4*300.**2/(300.+225.) ;
shift=-5.5d-4*temper**2/(temper+225.) ;
1.9+.125*xb+0.143*xb**2+shift-shift0
end_function
$
lband_bar value=0.28
$
$ revised jun04
$delta_so_well value=0.366
delta_so_well variation=function
function(xw,xb,temper)
0.343-0.0628*xw
end_function
delta_so_bar variation=function
function(xw,xb,temper)
0.343-0.0628*xb
end_function
$
band_offset value=0.6
$
mass_gamma_well variation=function
function(xw)
0.067+0.083*xw
end_function
$
mass_gamma_bar variation=function
function(xb)
0.067+0.083*xb
end_function
$
mass_l_well variation=function
function(xw)
0.56+0.1*xw
end_function
$
mass_l_bar variation=function
function(xb)
0.56+0.1*xb
end_function
```

```
$
gamma1_well variation=function
function(xw)
g1ga=6.9 ;
g1al=3.45 ;
g1ga*(1.-xw)+g1al*xw
end_function
$
gamma2_well variation=function
function(xw)
g2ga=2.2 ;
g2al=0.68 ;
g2ga*(1.-xw)+g2al*xw
end_function
$
gamma3_well variation=function
function(xw)
g3ga=2.9 ;
g3al=1.29 ;
g3ga*(1.-xw)+g3al*xw
end_function
$
a_well variation=function
function(xw)
dhga=-9.8 ;
dhal=-9.8 ;
dhga* (1. - xw) +dhal*xw
end_function
$
b_well variation=function
function(xw)
duga=-1.76 ;
dual=-1.76 ;
duga*(1.-xw)+dual*xw
end_function
$
c11_well variation=function
function(xw)
c11ga=11.9 ;
c11al=12.02 ;
c11ga*(1.-xw)+c11al*xw
end_function
$
```

```
c12_well variation=function
function(xw)
c12ga=5.38 ;
c12al=5.70 ;
c12ga*(1.-xw)+c12al*xw
end_function
$
gamma1_bar variation=function
function(xb)
g1ga=6.9 ;
g1al=3.45 ;
g1ga*(1.-xb)+g1al*xb
end_function
$
gamma2_bar variation=function
function(xb)
g2ga=2.2 ;
g2al=0.68 ;
g2ga*(1.-xb)+g2al*xb
end_function
$
gamma3_bar variation=function
function(xb)
g3ga=2.9 ;
g3al=1.29 ;
g3ga*(1.-xb)+g3al*xb
end_function
$
a_bar variation=function
function(xb)
dhga=-9.8 ;
dhal=-9.8 ;
dhga*(1.-xb)+dhal*xb
end_function
$
b_bar variation=function
function(xb)
duga=-1.76 ;
dual=-1.76 ;
duga*(1.-xb)+dual*xb
end_function
$
c11_bar variation=function
```

```
function(xb)
c11ga=11.9 ;
c11al=12.02 ;
c11ga*(1.-xb)+c11al*xb
end_function
$
c12_bar variation=function
function(xb)
c12ga=5.38 ;
c12al=5.70 ;
c12ga*(1.-xb)+c12al*xb
end_function
$

kane_para_f_well variation=function
function(xw)
gaas=-1.94;
alas=-0.48;
(1-xw)*gaas+xw*alas
end_function

kane_para_f_bar variation=function
function(xb)
gaas=-1.94;
alas=-0.48;
(1-xb)*gaas+xb*alas
end_function


lattice_constant value=5.65325
$
end_active_layer AlGaAs/AlGaAs
```

B.5 Wurtzite passive macro

This section shows the passive macro of a common wurtzite material ($In_xGa_{1-x}N$). Note how the strain parameters are critical.

```
$*******************************************
$ [free-style]
$ macro ingan
$ Bulk In(x)Ga(1-x)N parameters
```

```
$ Contributors: Peter Mensz, Khalid Shahzad (and maybe) others.
$ Changed by Kehl Sink 6/10/99
$ changed by Joachim Piprek 12/2/99 + 1/7/00
$ revised By zqli 05/2005 for band gap and bowing
$   InN band gap of 0.71 is from T Matsuoka et al.
$   Appl. Phys. Lett., 12, 1246(2002)
$ revised jul07
$ 1) use unstrained offset 2) include temper. 3) reduce lifetime
$ Typical use:
$   load_macro name=ingan var1=#x mater=#m var_symbol1=x
$ parameter_range x=[0 1]
$ parameter_range temper=[77 600]
$ ****************************************
begin_macro ingan
material type=wurtzite band_valleys=(1 1) &&
  el_vel_model=beta  hole_vel_model=beta
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ bowing=3. & band-structure parameters
$ according to JAP Vol 89 No 11 pp 5815-5875
$ modified by George Xiao on Aug. 29, 2006
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$---wurtzite parameters-----------
$---These are the same as in corresponding Active Layer Macro
$---Please see references therein

lattice_base value=3.189
lattice_c_base value=5.185
lattice_bulk variation=function
function(x,temper)
3.189*(1-x)+3.545*x
end_function
lattice_c_bulk variation=function
function(x)
gan=5.185;
inn=5.718;
gan*(1-x)+inn*x
end_function
$
eg0_bulk variation=function
function(x,temper)
eg0gan=3.507-9.09d-4*temper**2/(830+temper);
```

```
eg0inn=0.735-2.45d-4*temper**2/(624+temper);
bowing=3.0;
(1.0-x)*eg0gan+eg0inn*x-bowing*x*(1-x)
end_function
$
delta1_bulk variation=function
function(x,temper)
gan=0.019;
inn=0.041;
gan+(inn-gan)*x
end_function
$
delta2_bulk variation=function
function(x,temper)
gan=0.0047;
inn=0.00033;
gan+(inn-gan)*x
end_function
$
delta3_bulk variation=function
function(x,temper)
gan=0.0047;
inn=0.00033;
gan+(inn-gan)*x
end_function
$
mass_gamma_bulk variation=function
function(x)
gan=0.20;
inn=0.12;
gan+(inn-gan)*x
end_function
$
tmass_gamma_bulk variation=function
function(x)
gan=0.20;
inn=0.12;
gan+(inn-gan)*x
end_function
$
a1_bulk variation=function
function(x,temper)
inn=-8.21;
```

```
gan=-6.56;
gan+(inn-gan)*x
end_function
$
a2_bulk variation=function
function(x,temper)
inn=-0.68;
gan=-0.91;
gan+(inn-gan)*x
end_function
$
a3_bulk variation=function
function(x,temper)
inn=7.57;
gan=5.65;
gan+(inn-gan)*x
end_function
$
a4_bulk variation=function
function(x,temper)
inn=-5.23;
gan=-2.83;
gan+(inn-gan)*x
end_function
$
a5_bulk variation=function
function(x,temper)
inn=-5.11;
gan=-3.13;
gan+(inn-gan)*x
end_function
$
a6_bulk variation=function
function(x,temper)
inn=-5.96;
gan=-4.86;
gan+(inn-gan)*x
end_function
$
c33_bulk variation=function
function(x,temper)
gan=398;
inn=224;
```

```
gan+(inn-gan)*x
end_function
$
c13_bulk variation=function
function(x,temper)
gan=106;
inn=92;
gan+(inn-gan)*x
end_function

c11_bulk variation=function
function(x,temper)
gan=390;
inn=271;
gan+(inn-gan)*x
end_function

c12_bulk variation=function
function(x,temper)
gan=145;
inn=124;
gan+(inn-gan)*x
end_function

c44_bulk variation=function
function(x,temper)
gan=105;
inn=46;
gan+(inn-gan)*x
end_function

$
$ GaN values for def. pot.
$
d1_bulk variation=function
function(x,temper)
gan=-3.0;
inn=-3.0;
gan+(inn-gan)*x
end_function
$
d2_bulk variation=function
function(x,temper)
```

```
gan=3.6;
inn=3.6;
gan+(inn-gan)*x
end_function
$
d3_bulk variation=function
function(x,temper)
gan=8.82;
inn=8.82;
gan+(inn-gan)*x
end_function
$
d4_bulk variation=function
function(x,temper)
gan=-4.41;
inn=-4.41;
gan+(inn-gan)*x
end_function

d5_bulk variation=function
function(x,temper)
gan=-4.00;
inn=-2.33;
gan+(inn-gan)*x
end_function

d6_bulk variation=function
function(x,temper)
gan=-4.00;
inn=-2.33;
d5=gan+(inn-gan)*x;
d3=8.82;
(d3+4*d5)/sqrt(2)
end_function

$
$a_bulk value=-4.08
$ac_bulk value=-8.61
$
a_bulk variation=function
function(x,temper)
gan=-8.16;
inn=-3.5;
```

```
gan+(inn-gan)*x
end_function
$
ac_bulk variation=function
function(x,temper)
gan=-4.08;
inn=-3.5;
gan+(inn-gan)*x
end_function
$
$---end of wurtzite parameters-----------
$
dielectric_constant variation=function
function(x,temper)
gan=9.5;
inn=15;
gan+(inn-gan)*x
end_function
$
$ k.p theory:  unstrained ref. bandgap=eg0+delta1+delta2
$
affinity variation=function
function(x,temper)
gan=4.07;
eg0gan=3.507-9.09d-4*temper**2/(830+temper);
eg0inn=0.735-2.45d-4*temper**2/(624+temper);
bowing=3.0;
egx=(1.0-x)*eg0gan+eg0inn*x-bowing*x*(1-x);
$
delta1_gan=0.019;
delta1_inn=0.041;
delta1_x=delta1_gan+(delta1_inn-delta1_gan)*x;
$
delta2_gan=0.0047;
delta2_inn=0.00033;
delta2_x=delta2_gan+(delta2_inn-delta2_gan)*x;
$
$egx_ref=egx+delta1_x+delta2_x;
$eg0gan_ref=eg0gan+delta1_gan+delta2_gan;
egx_ref=egx;
eg0gan_ref=eg0gan;
off=0.67;
gan-(egx_ref-eg0gan_ref)*off
```

```
end_function

$
max_electron_mob variation=function
function(x,temper)
684.d-4*(300/temper)**1.5
end_function
min_electron_mob variation=function
function(x,temper)
386.d-4*(300/temper)**1.5
end_function
electron_ref_dens value=1.d23
alpha_n value=1.37
beta_n value=1.
electron_sat_vel value=1.d5
norm_field value=2.1e7
$ from [7]
max_hole_mob value=2.0d-4
min_hole_mob value=2.0d-4
hole_ref_dens value=2.75d23
alpha_p value=0.395
beta_p value=1.
hole_sat_vel value=1.d5
$
tau_energy value=1.d-13
$
lifetime_n value=1.e-7
lifetime_p value=1.e-7
$ adjusted 1/7/00
radiative_recomb  value=0.2d-16
auger_n value=1.d-46
auger_p value=1.d-46
$
$real_index value=2.7
$for x<0.3
real_index variation=function
function(x,temper)
gan=2.5067;
inn=3.4167;
gan+(inn-gan)*x
end_function
absorption value=0.
$ rough estimate for thin layers:
```

```
thermal_kappa value=10.
$
end_macro ingan
```

B.6 Wurtzite active macro

This section lists a typical wurtzite active layer: InGaN well grown on an InGaN barrier. The parameters for wurtzite semiconductors are substantially different from those of zincblende structure due to the complex coupling of the three valence bands. Many terms are needed to define the valence band Hamiltonian and various strain effects.

Since the whole structure is often grown on sapphire, it is also unclear exactly what the base lattice size should be. Thus the user is required to investigate this issue based on information of the growth on sapphire. In many cases, a thick GaN buffer layer is also used and the base lattice will be that of GaN.

```
$ ********************************************
$ Version 2005
$ active layer macro for
$ quantum well system of In(xw)Ga(1-xw)N/In(xb)Ga(1-xb)N
$ xw=In composition in well, xb=In composition in barrier,
$ temper=well temperature
$
$ [free-style]
$ revised by J. Piprek 9/9/2001
$ revised By zqli 05/2005 for band gap and bowing
$   InN band gap of 0.71 is from T Matsuoka et al.
$   Appl. Phys. Lett., 12, 1246(2002)
$ Strained quantum well grown on strained quantum barrier.
$ The substrate is assumed to be lattice matched to GaN
$ Typical use:
$   get_active_layer name=InGaN/InGaN  mater=#m &&
$     var1=#xw var2=#yw var_symbol1=xw var_symbol2=yw
$ parameter_range xw=[0 1]
$ parameter_range xb=[0 1]
$ parameter_range temper=[77 600]
$ ********************************************
$
begin_active_layer InGaN/InGaN
layer_type type=wurtzite_well valley_gamma=1 &&
  valley_hh=1 valley_lh=1 valley_ch=1
```

```
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ bowing=3. & band-structure parameters
$ according to JAP Vol 89 No 11 pp 5815-5875
$ modified by George Xiao on Sept. 14, 2006
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Lattice constant of InN is 3.548
$ Lattice constant of GaN is 3.189
$
$ We need to compute the strain based on lattice constant
$   of the substrate
$
lattice_constant value=3.189
lattice_c_constant value=5.185
$
lattice_well variation=function
function(xw,xb)
3.189*(1-xw)+3.545*xw
end_function
$
lattice_bar variation=function
function(xw,xb)
3.189*(1-xb)+3.545*xb
end_function

lattice_c_well variation=function
function(xw,xb)
gan=5.185;
inn=5.718;
gan*(1-xw)+inn*xw
end_function

lattice_c_bar variation=function
function(xw,xb)
gan=5.185;
inn=5.718;
gan*(1-xb)+inn*xb
end_function

$
$ --- energy parameters
$
```

```
$eg0_well variation=function
$function(xw,temper)
$eg0GaN=3.35564150943396;
$bowing=3.8;
$eg300=(1.0-xw)*eg0GaN+1.89*xw-bowing*xw*(1-xw);
$eg300-6.e-4*(temper-300.)
$end_function
$
eg0_well variation=function
function(xw,xb,temper)
eg0GaN=3.507-9.09d-4*temper**2/(830+temper);
eg0InN=0.735-2.45d-4*temper**2/(624+temper);
bowing=3;
$bowing=1.4;
(1.0-xw)*eg0GaN+eg0InN*xw-bowing*xw*(1-xw)
end_function
$
$eg0_bar variation=function
$function(xb,temper)
$eg0GaN=3.35564150943396;
$bowing=3.8;
$eg300=(1.0-xb)*eg0GaN+1.89*xb-bowing*xb*(1-xb);
$eg300-6.e-4*(temper-300.)
$end_function
$
eg0_bar variation=function
function(xw,xb,temper)
eg0GaN=3.507-9.09d-4*temper**2/(830+temper);
eg0InN=0.735-2.45d-4*temper**2/(624+temper);
bowing=3;
$bowing=1.4;
(1.0-xb)*eg0GaN+eg0InN*xb-bowing*xb*(1-xb)
end_function
$
band_offset value=0.67
$
delta_so_well variation=function
function(xw,xb)
gan=0.014;
inn=0.001;
gan+(inn-gan)*xw
end_function
$
```

```
delta_so_bar variation=function
function(xw,xb)
gan=0.014;
inn=0.001;
gan+(inn-gan)*xb
end_function
$
delta1_well variation=function
function(xw,xb)
gan=0.019;
inn=0.041;
gan+(inn-gan)*xw
end_function
$
delta1_bar variation=function
function(xw,xb)
gan=0.019;
inn=0.041;
gan+(inn-gan)*xb
end_function
$
delta2_well variation=function
function(xw,xb)
gan=0.0047;
inn=0.00033;
gan+(inn-gan)*xw
end_function
$
delta2_bar variation=function
function(xw,xb)
gan=0.0047;
inn=0.00033;
gan+(inn-gan)*xb
end_function
$
delta3_well variation=function
function(xw,xb)
gan=0.0047;
inn=0.00033;
gan+(inn-gan)*xw
end_function
$
delta3_bar variation=function
```

```
function(xw,xb)
gan=0.0047;
inn=0.00033;
gan+(inn-gan)*xb
end_function
$
$ ----conduction band masses
$
mass_gamma_well variation=function
function(xw,xb)
gan=0.20;
inn=0.12;
gan+(inn-gan)*xw
end_function
$
mass_gamma_bar variation=function
function(xw,xb)
gan=0.20;
inn=0.12;
gan+(inn-gan)*xb
end_function
$
tmass_gamma_well variation=function
function(xw,xb)
gan=0.20;
inn=0.12;
gan+(inn-gan)*xw
end_function
$
tmass_gamma_bar variation=function
function(xw,xb)
gan=0.20;
inn=0.12;
gan+(inn-gan)*xb
end_function
$
a1_well variation=function
function(xw,xb)
inn=-8.21;
gan=-6.56;
gan+(inn-gan)*xw
end_function
$
```

```
a2_well variation=function
function(xw,xb)
inn=-0.68;
gan=-0.91;
gan+(inn-gan)*xw
end_function
$
a3_well variation=function
function(xw,xb)
inn=7.57;
gan=5.65;
gan+(inn-gan)*xw
end_function
$
a4_well variation=function
function(xw,xb)
inn=-5.23;
gan=-2.83;
gan+(inn-gan)*xw
end_function
$
a5_well variation=function
function(xw,xb)
inn=-5.11;
gan=-3.13;
gan+(inn-gan)*xw
end_function
$
a6_well variation=function
function(xw,xb)
inn=-5.96;
gan=-4.86;
gan+(inn-gan)*xw
end_function
$
a1_bar variation=function
function(xw,xb)
inn=-8.21;
gan=-6.56;
gan+(inn-gan)*xb
end_function
$
a2_bar variation=function
```

```
function(xw,xb)
inn=-0.68;
gan=-0.91;
gan+(inn-gan)*xb
end_function
$
a3_bar variation=function
function(xw,xb)
inn=7.57;
gan=5.65;
gan+(inn-gan)*xb
end_function
$
a4_bar variation=function
function(xw,xb)
inn=-5.23;
gan=-2.83;
gan+(inn-gan)*xb
end_function
$
a5_bar variation=function
function(xw,xb)
inn=-5.11;
gan=-3.13;
gan+(inn-gan)*xb
end_function
$
a6_bar variation=function
function(xw,xb)
inn=-5.96;
gan=-4.86;
gan+(inn-gan)*xb
end_function
$
c33_well variation=function
function(xw,xb)
gan=398;
inn=224;
gan+(inn-gan)*xw
end_function
$
c33_bar variation=function
function(xw,xb)
```

```
gan=398;
inn=224;
gan+(inn-gan)*xb
end_function
$
c13_well variation=function
function(xw,xb)
gan=106;
inn=92;
gan+(inn-gan)*xw
end_function
$
c13_bar variation=function
function(xw,xb)
gan=106;
inn=92;
gan+(inn-gan)*xb
end_function

c11_well variation=function
function(xw,xb)
gan=390;
inn=271;
gan+(inn-gan)*xw
end_function

c11_bar variation=function
function(xw,xb)
gan=390;
inn=271;
gan+(inn-gan)*xb
end_function

c12_well variation=function
function(xw,xb)
gan=145;
inn=124;
gan+(inn-gan)*xw
end_function

c12_bar variation=function
function(xw,xb)
gan=145;
```

```
inn=124;
gan+(inn-gan)*xb
end_function

c44_well variation=function
function(xw,xb)
gan=105;
inn=46;
gan+(inn-gan)*xw
end_function

c44_bar variation=function
function(xw,xb)
gan=105;
inn=46;
gan+(inn-gan)*xb
end_function

$
$a_well value=-8.16
$ac_well value=-4.08
$a_bar value=-8.16
$ac_bar value=-4.08
$d1_well value=-0.89
$d1_bar value=-0.89
$d2_well value=4.27
$d2_bar value=4.27
$d3_well value=5.18
$d3_bar value=5.18
$d4_well value=-2.59
$d4_bar value=-2.59
$a_well value=-3.5
$ac_well value=-4.9
$a_bar value=-3.5
$ac_bar value=-4.9
$
a_well variation=function
function(xw,xb)
gan=-8.16;
inn=-3.5;
gan+(inn-gan)*xw
end_function
$
```

```
ac_well variation=function
function(xw,xb)
gan=-4.08;
inn=-1.75;
gan+(inn-gan)*xw
end_function

az_well variation=function
function(xw,xb)
gan=-8.16;
inn=-3.5;
gan+(inn-gan)*xw
end_function
$
acz_well variation=function
function(xw,xb)
gan=-4.08;
inn=-1.75;
gan+(inn-gan)*xw
end_function



$
a_bar variation=function
function(xw,xb)
gan=-8.16;
inn=-3.5;
gan+(inn-gan)*xb
end_function
$
ac_bar variation=function
function(xw,xb)
gan=-4.08;
inn=-3.5;
gan+(inn-gan)*xb
end_function

az_bar variation=function
function(xw,xb)
gan=-8.16;
inn=-3.5;
gan+(inn-gan)*xb
```

```
end_function
$
acz_bar variation=function
function(xw,xb)
gan=-4.08;
inn=-3.5;
gan+(inn-gan)*xb
end_function

$
$d1_well value=-3.7
$d1_bar value=-3.7
$d2_well value=4.5
$d2_bar value=4.5
$d3_well value=8.2
$d3_bar value=8.2
$d4_well value=-4.1
$d4_bar value=-4.1
$
d1_well variation=function
function(xw,xb)
gan=-3.0;
inn=-3.0;
gan+(inn-gan)*xw
end_function
$
d1_bar variation=function
function(xw,xb)
gan=-3.0;
inn=-3.0;
gan+(inn-gan)*xb
end_function
$
d2_well variation=function
function(xw,xb)
gan=3.6;
inn=3.6;
gan+(inn-gan)*xw
end_function
$
d2_bar variation=function
function(xw,xb)
gan=3.6;
```

```
inn=3.6;
gan+(inn-gan)*xb
end_function
$
d3_well variation=function
function(xw,xb)
gan=8.82;
inn=8.82;
gan+(inn-gan)*xw
end_function
$
d3_bar variation=function
function(xw,xb)
gan=8.82;
inn=8.82;
gan+(inn-gan)*xb
end_function
$
d4_well variation=function
function(xw,xb)
gan=-4.41;
inn=-4.41;
gan+(inn-gan)*xw
end_function
$
d4_bar variation=function
function(xw,xb)
gan=-4.41;
inn=-4.41;
gan+(inn-gan)*xb
end_function

d5_well variation=function
function(xw,xb)
gan=-4.00;
inn=-2.33;
gan+(inn-gan)*xw
end_function

d5_bar variation=function
function(xw,xb)
gan=-4.00;
inn=-2.33;
```

```
gan+(inn-gan)*xb
end_function

d6_well variation=function
function(xw,xb)
d5_gan=-4.00;
d5_inn=-2.33;
d5=d5_gan+(d5_inn-d5_gan)*xw;
d3=8.82;
(d3+4*d5)/sqrt(2)
end_function

d6_bar variation=function
function(xw,xb)
d5_gan=-4.00;
d5_inn=-2.33;
d5=d5_gan+(d5_inn-d5_gan)*xb;
d3=8.82;
(d3+4*d5)/sqrt(2)
end_function
$
end_active_layer InGaN/InGaN
```

PARS OPTICS GROUP

Appendix C

MODIFIED SCHARFETTER-GUMMEL FORMULA FOR HOT ELECTRONS

C.1 Current Flow in Hydrodynamic Model

The expression for hot electron current flow has been derived by Azoff [2][3]. Here we use a very initiative and less rigorous approach to derive the expression for hot electron current flow to gain insight into how it relates to the corresponding model in the conventional drift-diffusion model. We give all the details so that our user can check all the steps.

Our purpose is to convert our expression of current flow so that the spatial variation in the electron temperature (or electron energy) can be taken into account. We start by defining a ratio to indicate the effect of Fermi-Dirac statistics.

$$\gamma_F/kT = ln\left\{\frac{F_{1/2}[(E_{fn}-E_c)/kT]}{exp(E_{fn}-E_c)/kT]}\right\} \tag{C.1}$$

We recall that the electron concentration is given by

$$n = N_c F_{1/2}[(E_{fn}-E_c)/kT] \tag{C.2}$$

or in terms of the γ_F constant we just introduced,

$$n = exp[ln(N_c)]exp(\gamma_F/kT)exp[(E_{fn}-E_c)/kT] \tag{C.3}$$

or

$$kTln(n) = kTln(N_c) + \gamma_F + (E_{fn}-E_c) \tag{C.4}$$

We choose the vacuum as the zero energy and express the conduction band in terms of the potention and the electron affinity.

$$E_c = -\psi - \chi \tag{C.5}$$

Then we have

$$kTln(n) = kTln(N_c) + \gamma_F + E_{fn} + \psi + \chi \tag{C.6}$$

or

$$E_{fn} = kTln(n) - kTln(N_c) - \gamma_F - \psi - \chi \tag{C.7}$$

A well known formula in drift and diffusion theory relates the current flow to the quasi-Fermi level:

$$J_n q/\mu_n = n\nabla E_{fn} \tag{C.8}$$

where

$$E_{fn} = kTln(n) - kTln(N_c) - \gamma_F - \psi - \chi \tag{C.9}$$

We split this into drift and diffusion parts:

$$J_n q/\mu_n[drift] = n\nabla[-kTln(N_c) - \gamma_F - \psi - \chi] \tag{C.10}$$

$$J_n q/\mu_n[diff] = n\nabla[kTln(n)] \tag{C.11}$$

or

$$J_n q/\mu_n[diff] = \nabla[kTn] \tag{C.12}$$

We recall that the density of states is given by

$$N_c = M_x 2.5 \times 10^{25} \left(\frac{m_c kT}{km_0 300.}\right)^{3/2} \tag{C.13}$$

where M_x is the number of band valleys (or degeneracy) in k-space. The drift part under constant electron temperature is rewritten as

$$\begin{aligned} J_n q/\mu_n[drift] &= n\nabla[-kTln(N_c) - \gamma_F - \psi - \chi] && \text{(C.14)} \\ &= -nkT(3/2)\nabla ln(m_c) - n\nabla[\gamma_F + \psi + \chi] && \text{(C.15)} \end{aligned}$$

Here we have neglected the effect of temperature gradient to the drift part. We also assume that the temperature gradient contributes an extra diffusion term $n\nabla(kT)$. Therefore the diffusion current is proportional to the gradient of (kTn) instead of n alone:

$$J_n q/\mu_n[diff] = \nabla[kTn] \tag{C.16}$$

To avoid confusion with lattice temperature, we use T_e to denote electron carrier temperature. Our final results are

$$J_n q/\mu_n = -n\nabla[\gamma_F + \psi + \chi] - nkT_e(3/2)\nabla ln(m_c) + \nabla[kT_e n] \tag{C.17}$$

which is the same as that derived by Azoff [2][3] from Boltsman equation.

We define the electron energy as $w = (3/2)kT_e$ and we have the following final result for current flow:

$$J_n q/\mu_n = -n\nabla[\gamma_F + \psi + \chi] - nw\nabla ln(m_c) + (2/3)\nabla[wn] \tag{C.18}$$

C.2 Discretization for Hot Electron Current

In the conventional drift-diffusion model, the discretization of the current flow on a grid is best handled by the formula derived by Scharfetter and Gummel (SG-formula) in 1969 [11]. For hot electron model, the discretization formula must be modified to relfect the spatial variation of electron temperature. Following Azoff [3], we derive the new discretization as follwos.

We consider two grid points along the x-direction and express the current flow as follows:

$$(3/2)J_n q/\mu_n = -(3/2)n\frac{\partial}{\partial x}[\gamma_F + \psi + \chi] - (3/2)nw\frac{\partial}{\partial x}ln(m_c) + \frac{\partial}{\partial x}[wn] \tag{C.19}$$

We assume that the electric field between mesh points is constant. The electron temperature may be different between the two points, but we assume that the temparture variation is smooth enough so that $\frac{\partial w}{\partial x}$ and $\frac{\partial ln(w)}{\partial x}$ may be regarded as constants. We re-write the current:

$$(3/2)J_n q/\mu_n = \frac{\partial w}{\partial x}n\{-(3/2)\frac{\partial}{\partial x}[\gamma_F + \psi + \chi] - (3/2)w\frac{\partial}{\partial x}ln(m_c)\}\left(\frac{\partial w}{\partial x}\right)^{-1} + \frac{\partial}{\partial x}[wn] \tag{C.20}$$

or

$$(3/2)J_n q/\mu_n = \frac{\partial w}{\partial x}n\alpha + \frac{\partial}{\partial x}[wn], \tag{C.21}$$

where α is a position independent parameter based on our previous assumptions,

$$\alpha = \{(3/2)\frac{\partial}{\partial x}[-\gamma_F - \psi - \chi] - (3/2)w\frac{\partial}{\partial x}ln(m_c)\}\left(\frac{\partial w}{\partial x}\right)^{-1} \tag{C.22}$$

$$= \{(3/2)\frac{\partial}{\partial x}(-\gamma_F - \psi - \chi)\left(\frac{\partial w}{\partial x}\right)^{-1} - (3/2)\frac{\partial}{\partial x}ln(m_c)\left(\frac{\partial}{\partial x}ln(w)\right)^{-1} \tag{C.23}$$

We apply the basic assumption of the original SG-formula that the electron current is constant between two grid piints. Using w^α as an integration factor on Eq. (C.21), we obtain

$$(3/2)J_n q/\mu_n = w^{-\alpha}\left[\frac{\partial w}{\partial x} n\alpha w^{\alpha} + \frac{\partial}{\partial x}(wn)w^{\alpha}\right] \tag{C.24}$$

$$(3/2)J_n q/\mu_n = w^{-\alpha}\left[\frac{\partial w}{\partial x} n\alpha w^{\alpha} + \frac{\partial w}{\partial x} n w^{\alpha} + \frac{\partial n}{\partial x} w^{\alpha+1}\right] \tag{C.25}$$

$$(3/2)J_n q/\mu_n = w^{-\alpha}\left[\frac{\partial w}{\partial x} n(\alpha+1) w^{\alpha} + \frac{\partial n}{\partial x} w^{\alpha+1}\right] \tag{C.26}$$

$$(3/2)J_n q/\mu_n = w^{-\alpha}\left[\frac{\partial}{\partial x} w^{\alpha+1} n + \frac{\partial}{\partial x} n w^{\alpha+1}\right] \tag{C.27}$$

$$(3/2)J_n q/\mu_n = w^{-\alpha}\frac{\partial}{\partial x}(w^{\alpha+1} n) \tag{C.28}$$

We assume w is a linear function between the two grid points so that

$$w = w_a x + w_b \tag{C.29}$$

We re-write the equation as

$$(3/2)J_n q/\mu_n w^{\alpha} dx = d(w^{\alpha+1} n) \tag{C.30}$$

We integrate from point 1 to point 2 and get:

$$(3/2)J_n q/\mu_n \left[\frac{w^{\alpha+1}}{(1+\alpha)w_a}\right]_1^2 = [w^{\alpha+1} n]_1^2 \tag{C.31}$$

$$J_n q/\mu_n = \frac{(2/3)[w^{\alpha+1} n]_1^2}{\left[\frac{w^{\alpha+1}}{(1+\alpha)w_a}\right]_1^2} \tag{C.32}$$

$$= (2/3)(x_2 - x_1)^{-1}\frac{(w_2 - w_1)[w^{\alpha+1} n]_1^2}{\left[\frac{w^{\alpha+1}}{(1+\alpha)}\right]_1^2} \tag{C.33}$$

$$= (2/3)(x_2 - x_1)^{-1}\frac{(w_2 - w_1)(w_2^{\alpha+1} n_2 - w_1^{\alpha+1} n_1)(1+\alpha)}{w_2^{\alpha+1} - w_1^{\alpha+1}} \tag{C.34}$$

$$J_n q/\mu_n = (2/3)(x_2 - x_1)^{-1}(w_2 - w_1)(1+\alpha)\frac{w_2^{\alpha+1} n_2 - w_1^{\alpha+1} n_1}{w_2^{\alpha+1} - w_1^{\alpha+1}} \tag{C.35}$$

$$= (2/3)(x_2 - x_1)^{-1}(w_2 - w_1)(1+\alpha) \left[\frac{w_2^{\alpha+1} n_2}{w_2^{\alpha+1} - w_1^{\alpha+1}} + \frac{w_1^{\alpha+1} n_1}{w_1^{\alpha+1} - w_2^{\alpha+1}}\right] \tag{C.36}$$

We obtain our result:

$$J_n q/\mu_n = (2/3)(x_2 - x_1)^{-1}(w_2 - w_1)(1+\alpha)\left[\frac{n_1}{1 - (w_2/w_1)^{\alpha+1}} + \frac{n_2}{1 - (w_1/w_2)^{\alpha+1}}\right] \tag{C.37}$$

Furthermore, we wish to use the lattice temperature kT_L as the unit for $W = w/kT_L$ so that the equilibrium electron temperature is $W_0 = 3/2$. We obtain:

$$\begin{aligned} J_n &= \mu_n \left(\frac{kT_L}{q}\right)(2/3)(x_2 - x_1)^{-1}(W_2 - W_1)(1+\alpha) \\ &\quad \left[\frac{n_1}{1-(W_2/W_1)^{\alpha+1}} + \frac{n_2}{1-(W_1/W_2)^{\alpha+1}}\right] \qquad \text{(C.38)} \\ &= \mu_n v_t (2/3)(x_2 - x_1)^{-1}(W_2 - W_1)(1+\alpha) \\ &\quad \left[\frac{n_1}{1-(W_2/W_1)^{\alpha+1}} + \frac{n_2}{1-(W_1/W_2)^{\alpha+1}}\right] \qquad \text{(C.39)} \end{aligned}$$

where $v_t = kT_L/q$ is the lattice thermal voltage.

C.3 Hot Electron Current with Equal Energy

When the electron energy at the two points are equal or nearly equal, some approximation can be made. This is important in numerical simulation to avoid floating point problem when the carrier temperature of the adjacent points are nearly equal. We start from our basic equation before any discretization:

$$J_n q/\mu_n = -n\nabla(\gamma_F + \psi + \chi) - nw\nabla ln(m_c) + (2/3)\nabla(wn) \qquad \text{(C.40)}$$

When w can be considered constant (but not necessarily equal to the lattice temperature), we have

$$J_n q/\mu_n = -n\nabla[\gamma_F + \psi + \chi] - n(3/2)kT_e\nabla ln(m_c) + kT_e\nabla n \qquad \text{(C.41)}$$

$$J_n q/(kT_e\mu_n) = -n\nabla[\gamma_F + \psi + \chi + (3/2)kT_e\nabla ln(m_c)]/kT_e + \nabla n \qquad \text{(C.42)}$$

Again, we apply the basic assumption of the SG-formula that the current is constant between two points. Then we obtain a differential equation in the x-direction of the form:

$$a = -bn + \frac{dn}{dx} \qquad \text{(C.43)}$$

where

$$J_n q/(kT_e\mu_n) = -n\nabla[\gamma_F + \psi + \chi + (3/2)kT_e\nabla ln(m_c)]/kT_e + \nabla n \qquad \text{(C.44)}$$

$$a = J_n q/(kT_e\mu_n) \qquad \text{(C.45)}$$

$$b = \nabla[\gamma_F + \psi + \chi + (3/2)kT_e\nabla ln(m_c)]/kT_e \qquad \text{(C.46)}$$

Solving the simple differential equation from x_1 to x_2, we have

$$a = 1/(x_2 - x_1)b(x_2 - x_1)[n_2 - n_1 exp[b(x_2 - x_1)]]/[exp[b(x_2 - x_1)] - 1] \quad \text{(C.47)}$$

$$\begin{aligned} a(x_2 - x_1) &= b(x_2 - x_1)n_2/[exp[b(x_2 - x_1)] - 1] \\ &\quad -b(x_2 - x_1)n_1 exp[b(x_2 - x_1)]/[exp[b(x_2 - x_1)] - 1] \\ &= b(x_2 - x_1)n_2/[exp[b(x_2 - x_1)] - 1] \\ &\quad -b(x_2 - x_1)n_1/[1 - exp[b(x_1 - x_2)]] \\ &= b(x_2 - x_1)n_2/[exp[b(x_2 - x_1)] - 1] \\ &\quad -b(x_1 - x_2)n_1/[exp[b(x_1 - x_2)] - 1] \quad \text{(C.48)} \end{aligned}$$

$$a = \frac{1}{x_2 - x_1}[n_2 B[b(x_2 - x_1)] - n_1 B[b(x_1 - X_2)]] \quad \text{(C.49)}$$

where B() is the Bernouli function. Substituting a and b from Eqs. (C.45) and (C.46)

$$J_n q/(kT_e\mu_n) = \frac{1}{x_2 - x_1}[n_2 B(\eta_2^h - \eta_1^h) - n_1 B(\eta_1^h - \eta_2^h)] \quad \text{(C.50)}$$

$$J_n = V_{te}\frac{\mu_n}{x_2 - x_1}[n_2 B(\eta_2^h - \eta_1^h) - n_1 B(\eta_1^h - \eta_2^h)] \quad \text{(C.51)}$$

This is to be compared with the cool electron expression:

$$J_n = V_t\frac{\mu_n}{x_2 - x_1}[(n_2 B(\eta_2 - \eta_1) - n_1 B(\eta_1 - \eta_2)] \quad \text{(C.52)}$$

It is most convenient to relate the η^h to η

$$\begin{aligned} \eta^h &= [\gamma_F + \psi + \chi + kT_L ln(N_c)]/kT_e \\ &\quad +(3/2)\nabla ln(m_c) - ln(N_c)T_L/T_e \quad \text{(C.53)} \\ &= [\gamma_F + \psi + \chi + kT_L ln(N_c)]/kT_L(T_L/T_e) \\ &\quad +(3/2)\nabla ln(m_c) - ln(N_c)T_L/T_e \quad \text{(C.54)} \\ &= \eta(T_L/T_e) + (3/2)ln(m_c) - ln(N_c)T_L/T_e \quad \text{(C.55)} \end{aligned}$$

Using the unitless W:

$$\begin{aligned} \eta^h &= \eta 1.5(kT_L/1.5kT_e) \\ &\quad +(3/2)ln(m_c) - 1.5ln(N_c)(kT_L/1.5kT_e) \quad \text{(C.56)} \\ &= \eta 1.5/W + (3/2)ln(m_c) - 1.5ln(N_c)/W \quad \text{(C.57)} \end{aligned}$$

Similarly

$$V_{te} = kT_e/q \quad \text{(C.58)}$$

$$V_{te} = 1.5kT_e kT_L/(1.5kT_L q) \quad \text{(C.59)}$$

$$V_{te} = (1.5kT_e/kT_L)/1.5(kT_L/q) \quad \text{(C.60)}$$

$$V_{te} = W/1.5V_t \quad \text{(C.61)}$$

Appendix D

GREEN'S FUNCTION AND SPONTANEOUS EMISSION

This chapter starts from the Maxwell equation to establish some notations. Then we recall some basic theories of partial differential equations. Finally, we derive the theories for spontaneous recombination noise power which is critical for our understanding of linewidth and mode spectrum.

D.1 Maxwell Equations

The four fundamental equations of Maxwell are as follows:

$$\nabla \cdot \mathbf{D} = \rho, \tag{D.1}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{D.2}$$

$$\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} = 0, \tag{D.3}$$

$$\nabla \times \mathbf{H} - \frac{\partial \mathbf{D}}{\partial t} = \mathbf{J}. \tag{D.4}$$

The notations are defined as follows,

- **D** is the electrical displacement vector in $coulombs/meter^2$.
- **B** is the magnetic induction in $webers/meter^2$.
- **E** is the electric field intensity in $volts/meter$.
- **H** is the magnetic field in $ampere/meter$.

- ρ is the free charge density in $coulombs/meter^3$.
- $\mathbf{J}$ is the current density in $coulombs/meter^2$.

With the introduction of the polarization vector $\mathbf{P}$ and the magnetization vector $\mathbf{M}$, it is possible to eliminate the vectors $\mathbf{D}$ and $\mathbf{H}$ from the fundamental equations by substituting:

$$\mathbf{D} = \epsilon_0\mathbf{E} + \mathbf{P}, \tag{D.5}$$

$$\mathbf{H} = \frac{\mathbf{B}}{\mu_0} - \mathbf{M}. \tag{D.6}$$

Then we re-write the Maxwell equations in the following form:

$$\nabla \cdot \mathbf{E} = \frac{1}{\epsilon}(\rho - \nabla \cdot \mathbf{P}) \tag{D.7}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{D.8}$$

$$\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} = 0, \tag{D.9}$$

$$\nabla \times \mathbf{B} - \epsilon_0\mu_0\frac{\partial \mathbf{E}}{\partial t} = \mu_0\left(\mathbf{J} + \frac{\partial \mathbf{P}}{\partial t} + \nabla \times \mathbf{M}\right). \tag{D.10}$$

In our applications, no magnetization is involved. This allows us to set the $\mathbf{M}$ to be zero in the above equations. The Maxwell equations can be simplified further if we concentrate on the Fourier component at the optical frequency, also referred to as $\mathbf{E}_\omega$ elsewhere in the book. Using a time dependent factor of $exp(-j\omega t)$ for all the variables, we obtained the following expressions for the last two equations of the Maxwell equation set:

$$\nabla \times \mathbf{E} + j\omega\mathbf{B} = 0, \tag{D.11}$$

$$\nabla \times \mathbf{B} - j\omega\epsilon_0\mu_0\mathbf{E} = \mu_0\left(\mathbf{J} + j\omega\mathbf{P}\right). \tag{D.12}$$

In a waveguiding structure, we can consider the free current $\mathbf{J}$ to be zero at the optical frequencies. The polarization vector can be decomposed into two components as follows:

$$\mathbf{P} = \epsilon_0(\epsilon_r - 1)\mathbf{E} + P_{sp}. \tag{D.13}$$

where the first term is the polarization induced by the electric field which includes the usual effects from the dielectric constants and also effects like loss and gain. The second term is polarization induced by spontaneous emission of photons associated with radiative recombination of electron-hole pairs. Such a spontaneous emission process is random and not directly related to the external electric field intensity. As a results, we expect the second term to have a zero mean in both the spatial and time domain.

Elimination of the $\mathbf{B}$ vector from the Maxwell equations results in the following

$$-\nabla \times \nabla \times \mathbf{E} + \omega^2 \epsilon_0 \epsilon_r \mu_0 \mathbf{E} = -\mu_0 \omega^2 \mathbf{P_{sp}}. \tag{D.14}$$

With the assumption $\nabla \cdot \mathbf{E} = 0$, we finally arrived at the vector wave equation we use most often in our applications,

$$\nabla^2 \mathbf{E} + \omega^2 \epsilon_0 \epsilon_r \mu_0 \mathbf{E} = -\mu_0 \omega^2 \mathbf{P_{sp}}. \tag{D.15}$$

Using the speed of light $c_0 = 1/\sqrt{\epsilon_0 \mu_0}$ the wave equation can be written in a more familiar form:

$$\nabla^2 \mathbf{E} + \frac{\omega^2}{c_0^2} \epsilon_r \mathbf{E} = \mathbf{F}_\omega. \tag{D.16}$$

The right hand side term is also called the Langevin force F_ω:

$$F_\omega = -\mu_0 \omega^2 \mathbf{P_{sp}}. \tag{D.17}$$

D.2 Conventions

In this work we shall assume the oscillating field takes the form $exp(-j\omega t)$. As a result, the forward (or right going) wave propagates with a factor $exp(+jkz)$ and backward (or left going) wave with factor $exp(-jkz)$. Under such a convention, a loss material has positive imaginary wave number k'' where $k = k' + jk''$. Any time dependence quantities can be expanded in Fourier component with $exp(-j\omega t)$ and we assume that the real field can be expressed as

$$E(t) = \int_0^\infty E_\omega exp(-j\omega t) d\omega + c.c. \tag{D.18}$$

where c.c. is used to denote the complex conjugate. Using this notation, the field with single frequency can be written as

$$E(t) = E_\omega exp(-j\omega t) + E_\omega^* exp(j\omega t) \tag{D.19}$$

In some cases, we adopt the bracket notation from quantum mechanics such that

$$\phi(\mathbf{r}) = |\phi> = |n> \tag{D.20}$$

$$\phi^*(\mathbf{r}) = <\phi| = |n> \tag{D.21}$$

$$<\psi|\phi> = <n|n> = \int \psi^* \phi d\mathbf{r} \tag{D.22}$$

D.3 Basic Wave Equation

For simplicity we are only concerned with the scalar field in the optical frequency domain ω. The scalar wave equation is written as:

$$\nabla^2 E_\omega + \frac{\omega^2}{c_0^2}\epsilon_r E_\omega = F_\omega. \tag{D.23}$$

In a wave guide environment, it common to solve the above wave equation by separating variables in the field (we drop the subscript ω and it is understood that we deal with single frequency):

$$E(x, y, z) = Z(z)\phi(x, y) \tag{D.24}$$

or simply

$$E(x, y, z) = E(z)\phi(x, y) \tag{D.25}$$

We assume that there is a orthogonal and complete set of transverse modes $\phi_n(x, y)$ satisfying

$$\left[\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\omega^2}{c_0^2}\epsilon_r(\mathbf{r})\right]\phi_n(x, y) = \frac{\omega^2}{c_0^2} n_{eff}^2(z)\phi_n(x, y). \tag{D.26}$$

where the z-dependent effective index $n_{eff}(z)^2$ is also used to define the z-dependent wave number (or propagation constant):

$$k(z) = \frac{\omega}{c_0} n_{eff}(z). \tag{D.27}$$

We notice that the lateral solution $|\phi_n >$ may still be z-dependent. However, if the z-dependence is weak compared with the x-y dependence, we can treat it as a parameter and evade the differential operator $\frac{\partial}{\partial z}$. For this reason we temporarily hide its z-dependence here.

It is well known that solution to Eq. (D.26) has nice properties such as orthogonality and completeness. The orthogonality is

$$\int \phi_n \phi_m =< \phi_n \phi_m >= \delta_{nm} < \phi_n \phi_m > . \tag{D.28}$$

And the completeness is

$$\sum_n \frac{\phi_n(x, y)\phi_n(x_s, y_s)}{< \phi_n \phi_n >} = \delta(x - x_s)\delta(y - y_s). \tag{D.29}$$

For an arbitrary solution for $\phi(x, y)$, we expand it in terms of the complete set of ϕ_n

$$|\phi >= \sum_n A_n |\phi_n > \tag{D.30}$$

where

$$A_n = \frac{< \phi_n | \phi >}{< \phi_n | \phi_n >} \tag{D.31}$$

Using this expansion, we are able to rewrite the wave equation as follows:

$$E(z)\left(\frac{\partial^2}{x^2} + \frac{\partial^2}{y^2}\right)|\phi > + |\phi > \frac{\partial^2}{z^2}E(z) + \frac{\omega^2}{c_0^2}\epsilon_r E(z)|\phi > \;=\; F \tag{D.32}$$

$$E(z)\left(\frac{\partial^2}{x^2} + \frac{\partial^2}{y^2}\right)(\sum_n A_n|\phi_n >) + |\phi > \frac{\partial^2}{z^2}E(z) + \frac{\omega^2}{c_0^2}\epsilon_r E(z)(\sum_n A_n|\phi_n >) \;=\; F \tag{D.33}$$

$$|\phi > \frac{\partial^2}{z^2}E(z) + E_z\left[\left(\frac{\partial^2}{x^2} + \frac{\partial^2}{y^2}\right)(\sum_n A_n|\phi_n >) + \frac{\omega^2}{c_0^2}\epsilon_r(\sum_n A_n|\phi_n >)\right] \;=\; F \tag{D.34}$$

$$|\phi > \frac{\partial^2}{z^2}E(z) + E(z)\left[\left(\frac{\partial^2}{x^2} + \frac{\partial^2}{y^2}\right)(\sum_n A_n|\phi_n >) + \frac{\omega^2}{c_0^2}\epsilon_r(\sum_n A_n|\phi_n >)\right] \;=\; F \tag{D.35}$$

$$|\phi > \frac{\partial^2}{z^2}E(z) + (\sum_n A_n k^2 \phi_n)E(z) \;=\; F \tag{D.36}$$

Let us multiply the above equation by $< \phi_n|$ and obtain

$$< \phi_n|\phi > \frac{\partial^2}{z^2}E_z + A_n k^2 < \phi_n|\phi_n > E_z \;=\; < \phi_n|F > . \tag{D.37}$$

$$< \phi_n|\phi > \frac{\partial^2}{z^2}E_z + A_n k^2 < \phi_n|\phi_n > E_z \;=\; < \phi_n|F > . \tag{D.38}$$

$$< \phi_n|\phi > \frac{\partial^2}{z^2}E_z + k^2 < \phi_n|\phi > E_z \;=\; < \phi_n|F > . \tag{D.39}$$

$$\frac{\partial^2}{z^2}E_z + k^2 E_z \;=\; \frac{< \phi_n|F(x,y,z) >}{< \phi_n|\phi >} \tag{D.40}$$

$$\frac{\partial^2}{z^2}E_z + k^2 E_z \;=\; f(z) \tag{D.41}$$

where

$$f(z) = \frac{< \phi_n|F(x,y,z) >}{< \phi_n|\phi >} \tag{D.42}$$

We have finally reduced the 3-variable equation to a single variable differential equation. If we are lucky enough to find the solution for E_z, our final solution for the

three dimensional equation can be written as

$$E_z(z)[\sum_n A_n(z)\phi_n(x,y;z)] \tag{D.43}$$

Here we have explicitly show the dependence of the lateral solution to z. One good example of z-dependent solution is beam propagation in a tapered wave guiding structure.

D.4 The Green's Function

A well know technique for solving partial differential equation is the Green's function method. The basic idea is that for a differential equation of the form

$$\mathcal{L}(Z) = f(z) \tag{D.44}$$

where $\mathcal{L}$ is a linear differential operator. The solution can be written as

$$Z(z) = \int f(z_s)g(z,z_s)dz_s \tag{D.45}$$

where $g(z,z_s)$ is the Green's function satisfying

$$\mathcal{L}(Z) = \delta(z-z_s) \tag{D.46}$$

The interpretation of the Green function technique is simple enough: If we can find a solution due to a single point solution located at z_s, the solution due to a continuous distribution of sources $f(z)$ is just the superposition of all the solutions from the points sources contained in $f(z)$.

The Green's function technique is ideal for the analysis of our waveguiding system because the the spontaneous noise can be considered as a random ensemble of point sources. The Green's function allows us to give a compact formal solution in terms of the noise. Note that the formal solution is all we need here because it is impossible to write down an analytical expression for random noise.

D.5 Longitudinal Green's Function

Our final goal is to find the solution to the Green's function in Eq. (D.41). Several derivation steps should be taken before we arrive at the solution of the Green's function. We consider the following ordinary differential equation:

$$\mathcal{L}(Z) = \left(\frac{d^2}{dz^2} + k^2(z)\right) Z(z) = f(z). \tag{D.47}$$

The longitudinal Green's function is just a special case of the above equation with $f(z) = \delta(z-z_s)$.

D.5.1 The Wronskian

It is convenient to introduce the Wronskian from functions $Z_1(z)$ and $Z_2(z)$:

$$W = \begin{vmatrix} Z_1(z) & Z_1'(z) \\ Z_2(z) & Z_2'(z) \end{vmatrix} \tag{D.48}$$

$$= Z_1(z)Z_2'(z) - Z_2(z)Z_1'(z) \tag{D.49}$$

where we have used $'$ to denote the derivative with respect to z.

We notice a couple of interesting properties of the Wronskian. First the Wronskian can be used to test the dependency of two functions. If the two functions Z_1 and Z_2 are dependent on each other, i.e., if they are proportional to each other, then the Wronskian is zero everywhere. If the Wronskian differs from zero for any range of of values of z, then Z_1 is a completely different function from Z_2 and the two functions are said to be independent.

The second properties of the Wronskian is that, if Z_1 and Z_2 are two independent solutions of the homogeneous equation $\mathcal{L}(\mathcal{Z}) = \prime$, then the Wronskian should be independent of z, or in another word,

$$\frac{dW}{dz} = Z_1 Z_2'' - Z_2 Z_1'', \tag{D.50}$$

$$= Z_1[-k^2(z)Z_2] - Z_2[-k^2(z)Z_1], \tag{D.51}$$

$$= 0. \tag{D.52}$$

If W is not zero but a constant independent of z, and if we already find one solution Z_1 to the homogeneous equation $\mathcal{L}(Z) = 0$, then we can use the Wronskian to find another independent solution to the homogeneous equation as follows,

$$W = Z_1(z)Z_2'(z) - Z_2(z)Z_1'(z), \tag{D.53}$$

$$= Z_1^2 \frac{d}{dz}\left(\frac{Z_2}{Z_1}\right). \tag{D.54}$$

$$Z_2(z) = W Z_1(z) \int_{z_0}^{z} \frac{du}{Z_1^2(u)}. \tag{D.55}$$

D.5.2 Solution of Inhomogeneous Equation

Let us try to find a solution to the inhomogeneous equation Eq. (D.47) by setting $Z(z) = u(z)v(z)$ in $\mathcal{L}(Z) = f(z)$, obtaining

$$v\mathcal{L}(u) + uv'' + 2u'v' = f(z). \tag{D.56}$$

If now we set u equal to one of the solutions Z_1 of the homogeneous equation, $\mathcal{L}(Z) = 0$, we obtain

$$v'' + 2(Z_1'/Z_1)v' = f(z)/Z_1 \tag{D.57}$$

On the other hand, we already know that the second solution from is related to first by the Wronskian by the following [see Eq. (D.55)]

$$\frac{d}{dz}\left(\frac{Z_2(z)}{Z_1(z)}\right) = \frac{W}{Z_1^2(z)}. \tag{D.58}$$

Taking another derivative, we have

$$\begin{aligned}
\frac{d}{dz^2}\left(\frac{Z_2(z)}{Z_1(z)}\right) &= \frac{W}{Z_1^2(z)} && \text{(D.59)}\\
&= \left(\frac{W}{Z_z^2}\right)' && \text{(D.60)}\\
&= -2\frac{WZ_1'}{Z_1^3} && \text{(D.61)}
\end{aligned}$$

Using Eq. (D.58) so that

$$(Z_2/Z_1)'' + 2(Z_1/Z_1)(Z_2/Z_1)' = 0 \tag{D.62}$$

Multiplying this equation by v' and Eq. (D.57) by $(Z_2/Z_1)'$ and subtracting so that the second term on both equations cancel out, we obtain

$$\begin{aligned}
\left(\frac{Z_2}{Z_1}\right)' v'' - v'\left(\frac{Z_2}{Z_1}\right)'' &= \left[\left(\frac{Z_2}{Z_1}\right)'\right]^2 \frac{d}{dz}\left[\frac{v'}{(Z_2/Z1)'}\right] && \text{(D.63)}\\
&= \left(\frac{f(z)}{Z_1}\right)\left(\frac{Z_2}{Z_1}\right)' && \text{(D.64)}\\
&= \left(\frac{f(z)}{Z_1}\right)\left(\frac{W}{Z_1^2}\right) && \text{(D.65)}
\end{aligned}$$

By now, we have reduced our original second order inhomogeneous equation $\mathcal{L}(Z) = f(z)$ to a simple first order inhomogeneous equation

$$\frac{d}{dz}\left[\frac{v'}{(Z_2/Z_1)'}\right] = \frac{fZ_1}{W} \tag{D.66}$$

Integrating this first order equation, we obtain

$$v' = \left[\frac{d}{dz}\left(\frac{Z_2}{Z_1}\right)\right]\int \frac{fZ_1}{W}dz \tag{D.67}$$

or by adding another term on both side of the above equation

$$v' + \frac{fZ_2}{W} = \frac{d}{dz}\left[\left(\frac{Z_2}{Z_1}\right)\int \frac{fZ_1}{W}dz\right] \tag{D.68}$$

Integration of the above equation gives our final solution:

$$Z = vZ_1 \tag{D.69}$$
$$= Z_1 \int_z^{z_2} \frac{fZ_2 dz}{W} + Z_2 \int_{z_1}^{z} \frac{fZ_1 dz}{W} \tag{D.70}$$

where z_1 and z_2 are constants used to fit boundary conditions. It is interesting to note that z_1 and z_2 can be regarded as the starting and ending points along the longitudinal direction and Z_1 and Z_2 may be regarded as solutions to the homogeneous equation satisfying the boundary condition at z_1 and z_2. To be more specific, we examine Eq. (D.70) in more details. If we take the limit of $z- > z_1$, the second term goes to zero and the first one is proportional to Z_1. Similarly if $z- > z_2$, then the first term is zero and the second term is proportional to Z_2. Therefore, if Z_1 and Z_2 satisfy the boundary conditions at $z = z_1$ and $z = z_2$, respectively, then we have found a solution to the inhomogeneous equation which satisfies all boundary conditions.

In the case of Green's function, we set $f(z) = \delta(z - z_s)$ and obtain:

$$g(z, z_s) = Z_1 \int_z^{z_2} \frac{\delta(z - z_s)Z_2 dz}{W} + Z_2 \int_{z_1}^{z} \frac{\delta(z - z_s)Z_1 dz}{W} \tag{D.71}$$
$$g(z, z_s) = [Z_1(z)Z_2(z_s)\theta(z_s - z) + Z_2(z)Z_1(z_s)\theta(z - z_s)]/W \tag{D.72}$$

D.6 Green's Function for an AR Coated Waveguide

We consider a simple example of an unending waveguide (or a traveling wave amplifier with perfect anti-reflection AR coatings). This example has special significance because we need to use the result to derive a more general relation between spontaneous emission power and the diffusion coefficient.

We need to choose Z_1 and Z_2 to construct the Green's function and the Wronskian. For such a waveguide with perfect AR coating, the boundary condition at the two facets are waves traveling out of the device.

$$Z_1 = exp(-jkz) \tag{D.73}$$
$$Z_2 = exp(+jkz) \tag{D.74}$$

The Wronskian is

$$W_n = 2jk \tag{D.75}$$

and the Green's function is

$$
\begin{aligned}
g(z,z_s)W_n &= Z_1(z)Z_2(z_s)\theta(z_s-z)+Z_2(z)Z_1(z_s)\theta(z-z_s) && \text{(D.76)}\\
&= exp[-jk(z-z_s)]\theta(z_s-z)+exp[-jk(z-z_s)]\theta(z-z_s) && \text{(D.77)}\\
&= exp[-jk(z-z_s)]\theta(z_s-z)+exp[jk(z-z_s)]\theta(z-z_s) && \text{(D.78)}\\
&= exp(jk|z-z_s|) && \text{(D.79)}
\end{aligned}
$$

Let us try to find the solution of the wave equation in this waveguide if we assume the driving noise term is uncorrelated and has a δ function like correlation:

$$< f(z)f^*(z') >= f_0\delta(z-z') \tag{D.80}$$

where <> is used denote ensemble average and

$$f_0(z) = \left|\frac{< \phi_n|F(x,y,z) >}{< \phi_n|\phi >}\right|^2 \tag{D.81}$$

The field E_z can be expressed as a function of the Green's function:

$$
\begin{aligned}
E(z) &= \int f(z_s)g(z,z_s)dz_s && \text{(D.82)}\\
&= \int \frac{f(z_s)exp(jk|z-z_s|)}{W_n}dz_s && \text{(D.83)}\\
&= \int \frac{f(z_s)exp(jk|z-z_s|)}{2jk}dz_s && \text{(D.84)}
\end{aligned}
$$

Since $f(z)$ is used to represent noise, it is not possible to express E_z analytically. However, we can consider the average field amplitude due to the noise:

$$
\begin{aligned}
& < E_z(z)E_z^*(z) > \\
= & \int_0^L \int_0^L < f(z_s)f^*(z_s')g(z,z_s)g^*(z,z_s')dz_s dz_s' && \text{(D.85)} \\
= & \int_0^L \int_0^L \frac{< f(z_s)f^*(z_s') > exp(jk|z-z_s| - jk^*|z-z_s'|)}{4|k|^2} dz_s dz_s' \\
& && \text{(D.86)} \\
= & \int_0^L \frac{f_0\delta(z_s-z_s')exp(jk|z-z_s| - jk^*|z-z_s'|)}{4|k|^2} dz_s dz_s' && \text{(D.87)} \\
= & f_0 \int_0^L \frac{exp(-2k''|z-z_s|)}{4|k|} dz_s && \text{(D.88)} \\
= & f_0 \int_0^z \frac{exp[-2k''(z-z_s)]}{4|k|} dz_s + f_0 \int_z^L \frac{exp[+2k''(z-z_s)]}{4|k|} dz_s && \text{(D.89)} \\
= & \frac{f_0}{4|k|^2} \left\{ \int_0^z exp[-2k''(z-z_s)]dz_s + \int_z^L exp[+2k''(z-z_s)]dz_s \right\} \\
& && \text{(D.90)} \\
= & \frac{f_0}{4|k|^2} \left\{ \frac{1}{2k''} \{1 - exp[-2k''z]\} + \frac{1}{2k''} \{1 - exp[-2k''(L-z)]\} \right\} \\
& && \text{(D.91)} \\
= & \frac{f_0}{8|k|^2k''} \{\{1 - exp[-2k''z]\} + \{1 - exp[-2k''(L-z)]\}\} && \text{(D.92)}
\end{aligned}
$$

If we plot the above quantity as a function of the relative distance $z_r = z/L$, we will have the following interesting observations. In a gain medium, we define $g_r = -2k''L$ and

$$< E_z(z)E_z^*(z) > \frac{4|k|^2}{f_0} = \frac{1}{g_r} \{exp[g_r z_r] - 1\} + \frac{1}{g_r} \{exp[g_r(1-z_r)] - 1\} \quad \text{(D.93)}$$

Similarly in the case of a loss material, we set $\alpha_r = \alpha L = 2kL$ and obtain

$$< E_z(z)E_z^*(z) > \frac{4|k|^2}{f_0} = \frac{1}{\alpha_r} \{1 - exp[-\alpha_r z_r]\} + \frac{1}{\alpha_r} \{1 - exp[-\alpha_r(1-z_r)]\} \quad \text{(D.94)}$$

We plot the amplitude of the field due to the noise in Fig D.1. It shows that for a gain medium, the noise field amplitude (or the noise power) gets amplified and shows a maximum value at both side of the device, while a lossy material attenuates the noise field and has minimum amplitudes at both ends.

In the following sections, we need to estimate the pure effect of spontaneous emission. This requires the elimination of boundary condition and any amplification effect. A

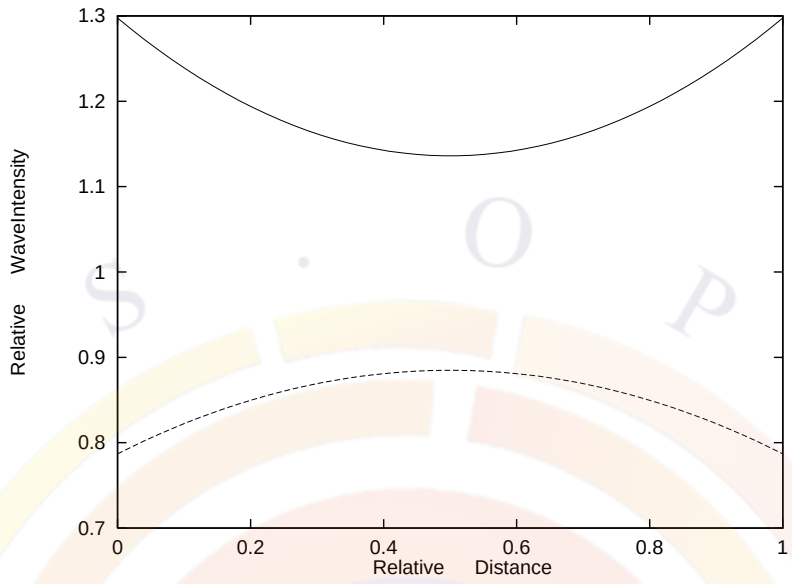


Figure D.1: The relative wave intensity of wave in a AR coated waveguide with gain (solid) and loss (dash). The relative gain (g_r)and loss (α_r) are set at 0.5 each.

long and lossy waveguide is regarded to have this pure effect at one end. Consider z=L and L being large, we have

$$< E_z(z)E_z^*(z) >= \frac{f_0}{8|k|^2k''} \tag{D.95}$$

or

$$< E_z(z)E_z^*(z) >= \frac{2D_{FF}(z)}{8|k|^2k''} \tag{D.96}$$

D.7 Diffusion Coefficient of the Langevin Forces

The driving noise term F appearing on the right hand side of wave equation is random in nature and uncorrelated over time or distance. Our starting point is that the ensemble average

$$< F_\omega(\mathbf{r})F_{\omega'}(\mathbf{r}') > = 0 \tag{D.97}$$
$$< F_\omega^*(\mathbf{r})F_{\omega'}^*(\mathbf{r}') > = 0 \tag{D.98}$$

and the correlation function is of δ function like:

$$< F_\omega(\mathbf{r})F_{\omega'}^*(\mathbf{r}') >= 2D_{FF^*}(\mathbf{r})\delta(\mathbf{r}-\mathbf{r}')\delta(\omega-\omega') \tag{D.99}$$

where the coefficient $2D_{FF^*}(\mathbf{r})$ was introduced by C. H. Henry [80] and was called "diffusion coefficient". For convenience we drop the delta function in ω and always use the same frequency for all the field quantities. The driving noise term can be expressed as

$$f(z) = \frac{<\phi_n|F(x,y,z)>}{<\phi_n|\phi>} \tag{D.100}$$

$$f(z')^* = \frac{<F(x',y',z')|\phi_n>,}{<\phi|\phi_n>} \tag{D.101}$$

$$<f(z)f(z')^*> = \left\langle \frac{<\phi_n(\mathbf{r}_{xy})|F(\mathbf{r})>}{<\phi_n|\phi>} \frac{<F(\mathbf{r}')|\phi_n(\mathbf{r}'_{xy})>}{<\phi|\phi_n>} \right\rangle \tag{D.102}$$

$$= \left\langle \frac{<\phi_n(\mathbf{r}_{xy})|<F(\mathbf{r}')|F(\mathbf{r})>|\phi_n(\mathbf{r}'_{xy})>}{|<\phi_n|\phi>|^2} \right\rangle \tag{D.103}$$

$$= \frac{<\phi_n(\mathbf{r}_{xy})|2D_{FF^*}(\mathbf{r})\delta(\mathbf{r}'-\mathbf{r})|\phi_n(\mathbf{r}'_{xy})>}{|<\phi_n|\phi>|^2} \tag{D.104}$$

$$= \frac{<\phi_n(\mathbf{r}_{xy})|2D_{FF^*}|\phi_n(\mathbf{r}_{xy})>}{|<\phi_n|\phi>|^2}\delta(z-z') \tag{D.105}$$

$$= f_0(z)\delta(z-z') \tag{D.106}$$

In order to determine the diffusion coefficient, we wish to evaluate the emitted optical power within $\Delta\omega$ from the bulk photon energy density as follows:

$$S_\omega \hbar\omega = 2\varepsilon_0 n n_g < |E(z)|^2 > |\phi(x,y)|^2 \tag{D.107}$$

The linear energy density is as follows

$$s_\omega \hbar\omega = 2\varepsilon_0 n n_g < |E(z)|^2 > \int dxdy|\phi(x,y)|^2 \tag{D.108}$$

or

$$s_\omega \hbar\omega = 2\varepsilon_0 n n_g < |E(z)|^2 > \tag{D.109}$$

From the result of the previous section

$$s_\omega \hbar\omega = 2\varepsilon_0 n n_g \frac{2D_{FF}(z)}{8|k|^2 k''} \tag{D.110}$$

The power emitted is just the above times v_g:

$$P_n = v_g 2\varepsilon_0 n n_g \frac{2D_{FF}(z)}{8|k|^2 k''} \tag{D.111}$$

$$P_n = -v_g 2\varepsilon_0 n n_g \frac{2D_{FF}(z)}{4|k|^2 g} \tag{D.112}$$

$$P_n = -c_0 \varepsilon_0 n \frac{2D_{FF}(z)}{2|k|^2 g} \tag{D.113}$$

Please note the minus sign comes in because $-g = 2k''$.

Now let us derive the optical power from a completely different view point. When the photons are in equilibrium with the environment, the average noise power within $\Delta\omega$ can be expressed in terms of the mode occupation number $< n_\omega >$:

$$P_n \Delta\omega = \sum_n v_{gn} \hbar\omega < n_\omega > \Delta\omega D_1(E) \tag{D.114}$$

where $D_1(E)$ is the 1D mode density (modes per unit energy per unit length) which is derived as follows.

The mode spacing is

$$\Delta k_z = \frac{2\pi}{L_z} \tag{D.115}$$

Therefore the mode number per unit length per unit ω is $D_1(E) = \frac{1}{2\pi}\frac{dk}{d\omega}$. To a good approximation, the group velocity may be used:

$$\frac{dk}{d\omega} = 1/v_g \tag{D.116}$$

This gives a power

$$P_n = \hbar\omega < n_\omega > /(2\pi) \tag{D.117}$$

Comparison of the two different expressions of optical power, we finally obtain the following relation for the diffusion coefficient:

$$2D_{FF}(z) = -|k|^2 \hbar\omega g < n_\omega > /(\pi c_0 \varepsilon_0 n) \tag{D.118}$$

or, if we neglect the imaginary part of the refractive index in $|k|$,

$$2D_{FF}(z) = -\frac{\hbar\omega^3 n'(z) g(z) < n_\omega >}{\pi c_0^3 \varepsilon_0} \tag{D.119}$$

Please note that a difference of factor $(2\pi)^2$ than that from Ref. [80] is due to the use of different unit system.

According to Henry [80], the mode occupation number is given by

$$< n_\omega >= -n_{sp} = \frac{1}{exp\left(\frac{\hbar\omega - eV_F}{kT}\right) - 1} \tag{D.120}$$

Using the well-known relation between optical gain and spontaneous emission, we can also use an alternative (or more convenient) form for the diffusion coefficient in terms of bulk spontaneous recombination rate.

$$g = \frac{r_{sp}(E)}{v_g D(E)} \left\{1 - exp[(\hbar\omega - eV_F)/kT]\right\}. \tag{D.121}$$

where

$$D(E) = \frac{\pi k^2 \Delta k}{\pi^3 \Delta E}. \tag{D.122}$$

$$= \left(\frac{n}{\hbar c}\right)^3 \frac{E^2}{\pi^2} \tag{D.123}$$

is the photon mode density. Thus, the product gn_{sp} be re-written as

$$gn_{sp} = \frac{r_{sp}(E)}{v_g D(E)} \tag{D.124}$$

Using the above for the n-th lateral mode, we have

$$2D_{FF}(z) = |k|^2 E < \phi_n | r_{sp} | \phi_n > /(D(E) v_g \pi c_0 \varepsilon_0 n) \tag{D.125}$$

$$2D_{FF}(z) = n^2 \omega^2 E < n | r_{sp} | n > \pi^2 \hbar^3 c_0^3 / (n^3 E^2 v_g \pi c_0^3 \varepsilon_0 n) \tag{D.126}$$

$$2D_{FF}(z) = (\hbar\omega) < n | r_{sp}(E) | n > \pi\hbar / (n^2 v_g \varepsilon_0) \tag{D.127}$$

Please note that $r_{sp}(E)$ must be in units of rate per volume per energy interval.

PARS · OPTICS · GROUP

Appendix E

THEORY OF POWER SPECTRUM

It has been shown in Ref. [106] that near lasing the Wronskian is independent of the position z and is only a function of frequency. Thus we can expand it over a complex omega plane as follows:

$$\frac{1}{W} \approx \sum_i \frac{1}{\frac{dW(\omega_i)}{d\omega}(\omega - \omega_i)} \tag{E.1}$$

Let us derive the photon density in terms of the electric field. According to Landau and Lifschitz [155], the energy density, U, of a dispersive medium can be written as

$$U = \varepsilon_0 \frac{\partial}{\partial \omega}(\varepsilon' \omega) E^2 + \varepsilon_0(\varepsilon') E^2 \tag{E.2}$$

where ε' is the real part of the dielectric constant at the optical frequencies. We go on and derive the following:

$$U = 2\varepsilon_0 n' \omega \frac{\partial n'}{\partial \omega} E^2 + \varepsilon_0 n'^2 E^2 + \varepsilon_0 \varepsilon' E^2 \tag{E.3}$$

which can be expressed in terms of the group index as follows:

$$U = 2\varepsilon_0 \omega \frac{\partial n'}{\partial \omega} E^2 + 2\varepsilon_0 n' E^2 \tag{E.4}$$

$$= 2\varepsilon_0 n' n_g E^2 \tag{E.5}$$

where

$$n_g = n' + \omega \frac{\partial n'}{\partial \omega} \tag{E.6}$$

Using the Fourier transform of the electric field, we write down our expression of photon density as

$$S(\omega) = \frac{U}{\hbar\omega} = \frac{2\varepsilon_0 n' n_g}{\hbar\omega} \left| E_\omega(z)\phi(x,y) \right|^2 \tag{E.7}$$

Integration over x-y leads to the linear photon density:

$$s_l(\omega, z) = \frac{2\varepsilon_0 n' n_g}{\hbar\omega} |E_\omega(z)|^2 \tag{E.8}$$

We recall that the spectrum of the electric field can be repressed in terms of the Green's function as follows:

$$E_\omega(z) = \frac{\int_0^l g(z, z') f_\omega(z') dz'}{W(\omega)} \tag{E.9}$$

We perform the spatial integral

$$\begin{aligned} s_l(\omega, z) &= \frac{2\varepsilon_0 n'(z) n_g}{\hbar\omega} |E_\omega(z)|^2 \\ &= \frac{2\varepsilon_0 n'(z) n_g}{\hbar\omega |W(\omega)|^2} \int_0^l g(z, z')^* f_\omega^*(z') dz' \int_0^l g(z, z'') f_\omega(z'') dz'' \\ &= \frac{2\varepsilon_0 n'(z) n_g}{\hbar\omega |W(\omega)|^2} \int_0^l \int_0^l g(z, z')^* g(z, z'') f_\omega^*(z') f_\omega(z'') dz' dz'' \qquad \text{(E.10)} \\ & \qquad \text{(E.11)} \end{aligned}$$

We take the ensemble average of the above to obtain:

$$\begin{aligned} s_l(\omega, z) &= \frac{2\varepsilon_0 n'(z) n_g}{\hbar\omega |W(\omega)|^2} \int_0^l \int_0^l g(z, z')^* g(z, z'') < f_\omega^*(z') f_\omega(z'') > dz' dz'' \\ &= \frac{2\varepsilon_0 n'(z) n_g}{\hbar\omega |W(\omega)|^2} \int_0^l \int_0^l g(z, z')^* g(z, z'') 2D_\omega(z') \delta(z' - z'') dz' dz'' \\ &= \frac{2\varepsilon_0 n'(z) n_g}{\hbar\omega |W(\omega)|^2} \int_0^l |g(z, z')|^2 2D_\omega(z') dz' \qquad \text{(E.12)} \end{aligned}$$

Let us transform the Wronskin as follows:

$$\begin{aligned} \frac{1}{|W(\omega)|^2} &= \sum_i \sum_j \frac{1}{\frac{dW^*(\omega_i)}{d\omega}(\omega - \omega_i^*)} \frac{1}{\frac{dW(\omega_j)}{d\omega}(\omega - \omega_j)} \\ &= \sum_i \frac{1}{|\frac{dW(\omega_i)}{d\omega}|^2 (\omega - \omega_i^*)(\omega - \omega_i)} \\ &= \sum_i \frac{1}{|\frac{dW(\omega_i)}{d\omega}|^2 [(\omega - \omega_{re,i})^2 + \omega_{im,i}^2]} \qquad \text{(E.13)} \end{aligned}$$

Please note that we have dropped the cross terms of $\frac{1}{(\omega-\omega_i)(\omega-\omega_j)^*}$ because these are normally much smaller than the sqared terms.

Using and n' and n interchangeably and combining Eq. (E.12) and Eq. (E.13), we obtain

$$s_l(\omega, z) = \frac{2\epsilon_0 n(z) n_g}{\hbar\omega} \left[\sum_i \frac{1}{|\frac{dW(\omega_i)}{d\omega}|^2 [(\omega - \omega_{re,i})^2 + \omega_{im,i}^2]} \right] \int_0^l |g(z, z')|^2 2D_\omega(z') dz' \tag{E.14}$$

where $\omega_{re,i}$ and $\omega_{im,i}$ are the real and imaginary parts of a pole respectively. ϵ_0, n and n_g are the vacuum permitivity, refractive index and group index, respectively. $D_\omega(z)$ is the diffusion coefficient of the Langevin force $f_\omega(z)$ [106]. The above expression shows that the spectrum of the photon density is a sum of Lorentzian lines centered at $\omega_{re,i}$, with peak intensity proportional to $1/\omega_{im,i}^2$.

We integrate over z and obtain the modal spectrum:

$$\begin{aligned} I_\omega &= \int_0^l s_l(\omega, z)dz \\ &= \frac{4\pi\epsilon_0}{\hbar\omega}\sum_i \left[\left|\frac{dW(\omega_i)}{d\omega}\right|^{-2} \int_0^l\int_0^l n(z)n_g|g(z,z')|^2 2D_\omega(z')dz'dz\right] \\ &\times \frac{(2\pi)^{-1}}{[(\omega-\omega_{re,i})^2+\omega_{im,i}^2]} \end{aligned} \tag{E.15}$$

The term related to the Wronskian has been worked out in Ref. [106] as

$$\frac{dW(\omega_i)}{d\omega} = 2\int_0^l k(z)Z_L(z)Z_R(z)\frac{\partial k(\omega_i,z)}{\partial\omega}dz \tag{E.16}$$

Near the material gains peak, to a good approximation we obtain

$$\frac{\partial k}{\partial\omega} = \frac{1}{v_g} \tag{E.17}$$

Assuming the system is near threshold, both $Z_L(z)$ and $Z_R(z)$ reduce to a common solution $Z_{0,i}(z)$ where the label i indicates wave function at frequency ω_i:

$$\frac{dW(\omega_i)}{d\omega} = \frac{2}{v_g}\int_0^l k(z)Z_{0,i}^2(z)dz \tag{E.18}$$

The Green's function is reduced to a simpler form:

$$g(z,z') = Z_0(z)Z_0(z') \tag{E.19}$$

where $Z_0(z)$ is understood as $Z_{0,i}$.

We have arrived at the following spectrum

$$\begin{aligned} I_\omega &= \frac{4\pi\epsilon_0}{\hbar\omega}\sum_i \frac{\int_0^l\int_0^l n(z)n_g|Z_0(z)|^2|Z_0(z')|^2 2D_\omega(z')dz'dz}{\left|\frac{2\omega}{c^2}\int_0^l n(z)n_g Z_0^2(z)dz\right|^2} \\ &\times \frac{(2\pi)^{-1}}{[(\omega-\omega_{re,i})^2+\omega_{im,i}^2]} \end{aligned} \tag{E.20}$$

From Appendix D, we recall that the diffusion constant is related to the optical gain by

$$2D_{FF}(z) = \frac{\hbar\omega^3 n(z) g(z) n_{sp}}{\pi c^3 \varepsilon_0} \tag{E.21}$$

We thus obtain:

$$\begin{aligned} I_\omega &= \frac{4\pi\epsilon_0}{\hbar\omega}\frac{\hbar\omega^3}{\pi c^3\varepsilon_0}\sum_i \frac{\int_0^l\int_0^l n(z)n_g|Z_0(z)|^2|Z_0(z')|^2 n(z')g(z')n_{sp}dz'dz}{\left|\frac{2\omega}{c^2}\int_0^l n(z)n_g Z_0^2(z)dz\right|^2} \\ &\times \frac{(2\pi)^{-1}}{[(\omega-\omega_{re,i})^2+\omega_{im,i}^2]} \end{aligned} \tag{E.22}$$

which may be written in our final form of the spectrum:

$$\begin{aligned} I_\omega &= c\sum_i \frac{\left[\int_0^l n(z)n_g|Z_0(z)|^2dz\right]\left[\int_0^l |Z_0(z')|^2 n(z')g(z')n_{sp}dz'\right]}{\left|\int_0^l n(z)n_g Z_0^2(z)dz\right|^2} \\ &\times \frac{(2\pi)^{-1}}{[(\omega-\omega_{re,i})^2+\omega_{im,i}^2]} \end{aligned} \tag{E.23}$$

Introducing the parameter $R_{sp,i}$:

$$R_{sp,i} = c\frac{\left[\int_0^l n(z)n_g|Z_0(z)|^2dz\right]\left[\int_0^l |Z_0(z')|^2 n(z')g(z')n_{sp}dz'\right]}{\left|\int_0^l n(z)n_g Z_0^2(z)dz\right|^2} \tag{E.24}$$

which is also referred to as the rate of spontaneous emission into the i^{th} mode. The spectrum may be written as

$$I_\omega = \frac{(2\pi)^{-1}R_{sp,i}}{[(\omega-\omega_{re,i})^2+\omega_{im,i}^2]} \tag{E.25}$$

Integrating over all frequencies, we get the total number of photons in the cavity $I_{p,i}$ in the i^{th} mode:

$$I_{p,i} = \frac{R_{sp,i}}{2\omega_{im,i}} \tag{E.26}$$

which agrees with results from other works in the literature (see, e.g., Ref. [106]).

In the above formulas, $Z_0(z) = Z_{0,i}(z)$ is the solution to the homogeneous scalar wave equation at $\omega = \omega_i$, v_g the group velocity, $g(z)$ the modal gain, and $n_{sp}(z)$ the inversion parameter. The expression (E.24) agrees completely with results in Ref. [106].

Application notes on SMSR

The side-mode suppression ratio (SMSR) is often used to determine whether or not a laser is single-mode. Based on the above derivations, two definitions of SMSR may be used.

Based on the photon number (integrated over all frequencies):

$$SMSR_{ij} = \frac{I_{p,i}}{I_{p,j}} = \frac{R_{sp,i}}{R_{sp,j}} \frac{\omega_{im,j}}{\omega_{im,i}} \tag{E.27}$$

Based on the peaks of the power spectrum (at $\omega = \omega_{re,i}$ for each mode):

$$SMSR_{ij} = \frac{I_{\omega_i}}{I_{\omega_j}} = \frac{R_{sp,i}}{R_{sp,j}} \left(\frac{\omega_{im,j}}{\omega_{im,i}} \right)^2 \tag{E.28}$$

On a log scale, the two definitions therefore differ by a factor of 2, all other values being equal. The first definition is used in the *rtg_smsr* variable of **plot_scan**; the RTG power spectrum of **gain_spectrum** may be used to measure the second.

We also note that the side mode suppression ratio may be strongly affected by how the population inversion factor is defined inside the Green's function model; depending on the choice of **intern_spon_par** in **material_3d**, the spontaneous emission in each mode may be replaced by $R_{sp,av} = \sum_i^N R_{sp,i}/N$.

PARS OPTICS GROUP

Appendix F

NOMENCLATURE

A_x	Coefficient for band-gap narrowing effects.
a_0, a	Lattice constants of GaAs (a_0) and InGaAs (a).
B	Radiative recombination coefficient.
B_1,B_2,B_z	Complex wave amplitudes of the optical wave propagating from right to left in a laser.
B_k	Constant used in mathematical fittings.
b	Shear deformation potential.
C_n, C_p	Auger recombination coefficients, for electrons (n) and holes (p).
c_{nj}, c_{pj}	Electron (n), hole (p) capture coefficients of the j th deep trap.
c	Velocity of light in vacuum.
c_{11}, c_{12}	Elastic constants.
D_c, D_v	Density of states of the conduction and valence band.
E_D, E_A	Shallow donor (D) and acceptor (A) levels.
E_i^0, E_j^0, E_{ij}^0	Energy minimums of the i th and the j th levels, and the difference between them, . The i refers to the valence band sub-band levels and j refers to the conduction band sub-band levels.
E, E_{ij}	Photon energy and energy difference between the i th and j th levels.
E_{fn}, E_{fp}	Quasi-Fermi energies of electrons (n) and holes (p).
E_g	Bandgap.
E_{g0}	Unstrained bandgap.
ΔE_{PF}	Shift in ionization energy of a dopant due to Poole-Frenkel effect.
δE_{hy}	Hydrostatic strain energy.
δE_{sh}	Energy associated with shear strain.
F	Electric field intensity.
F_l	Lorentzian shape function.
F_{0n},F_{0p}	Threshold electric field used in the electron (n) and hole (p) mobility models.
$F_{1/2}$	Fermi integral of order one-half.

f_D, f_A	Occupancy of the donor (D) and acceptor (A) levels.
f_{tj}	Occupancy of the j th deep trap level.
f_i, f_j	Fermi functions for the i th and the j th level.
f_i', f_j'	Integrated Fermi functions for the i th and the j th levels.
g	Interband local gain.
g_{ij}, g	Local gain due to transition between the i th and the j th levels, and the local gain of the material.
g_d, g_a	Degeneracies of the shallow donor (d) and acceptor (a).
H_{ij}	Hamiltonian matrix element between the i th and the j th states.
$h(x)$	step function of variable x.
h,$\hbar$	Plank's constants.
I_1, I_2	The integrals needed to evaluate the quantum well gain and spontaneous emission rate.
J_n, J_p	Electron (n) and hole (p) current flux densities.
J_{sn}, J_{sp}	Electron (n) and hole (p) current flux densities on the surface.
J_{hn}, J_{hp}	Electron (n) and hole (p) current flux densities across the heterojunction.
J_{source}	Current flux source.
k	Boltzmann constant.
L	Laser cavity length.
$L()$	Lines shape function for gain broadening.
m_0	Electron mass
m_i, m_j, m_{ij}	Relative effective masses of the i th and the j th level and the reduced effective mass between the i th and the j th level. Their relation is defined by $1/m_{ij} = 1/m_i + 1/m_j$. Indices i and j refer here to quantum well sub-bands.
m_n,m_p	Bulk effective masses for electrons (n) and holes (p).
m_e,m_h	The same as m_n and m_p, respectively.
m_{bn},m_{bp}	Bulk effective masses for electrons (n) and holes (p) on the barrier side of the heterojunction.
m_{zl}	Relative effective mass of the L-band used in the quantum level calculation.
m_{vx}, m_{vy}, m_{vz}	Effective hole masses in the x, y and z direction.
M_0, M_{ij}, M_{lh}, M_{hh}	Momentum matrix elements for bulk material, between the i th and the j th states, involving light- and heavy-hole transitions, respectively.
N	Total number of grid points in the simulation space. Also used for the total number of longitudinal layers.
N_b	Number of grid points associated with a boundary of interest.
N_D, N_A	Doping density of shallow donors (D) and shallow acceptors (A).
N_{tj}	Density of the j th deep trap.
n	Electron concentration or density.

n_b	Electron concentration or density on the barrier side of the hetero-junction.
n_{b0}	Electron concentration or density on the barrier side of the hetero-junction when its quasi-Fermi level coincides with that on the other side.
n_s	Electron concentration or density on the surface.
$\overline{n}$	Real part of refractive index.
n_i	Intrinsic carrier density.
n_{2D}	The surface concentration of a quantum well, equal to the bulk density times the layer thickness.
N_{rn}, N_{rp}	Reference doping density in electron (n) and hole (p) mobility equations.
P_{ij}	Probability of a transition from the i th to the j th level.
p	Hole concentration or density.
p_b	Hole concentration or density on the barrier side of the hetero-junction.
p_{b0}	Hole concentration or density on the barrier side of the hetero-junction when its quasi-Fermi level coincides with that on the other side.
p_s	Hole concentration or density on the surface.
Q^j_{em}	Energy density of the jth emitted longitudinal mode.
Q^j_{cav}	Energy density of the jth longitudinal mode within the laser cavity.
q	Electronic charge.
R^{tj}_n, R^{tj}_p	Electron (n) and hole (p) recombination rates per unit volume through the j th deep trap.
R_{sp}, R^b_{sp}, R^{qw}_{sp}	Spontaneous recombination rate per unit volume, also called the radiative recombination rate, and the same quantity in bulk (b) and in the quantum well (qw).
r^{qw}_{sp}	Frequency dependent spontaneous recombination rate for the quantum well.
R_s	Resistance associated with a boundary condition.
R_{st}	Stimulated recombination rate per unit volume.
R_{au}	Auger recombination rate per unit volume.
r_m,r^{eff}_m	Single facet reflectivity and effective reflectivity of a laser.
T	Absolute temperature.
T_{scat}	The scattering kernel for carrier-carrier scattering.
t	Thickness of the quantum well.
V	Electrical potential.
V_s	Electrical potential on the surface.
$V_{applied}$	Applied electrical potential.
v	Volume.

$\overline{v}_n$, $\overline{v}_p$	Average thermal velocity of electrons (n) and holes (p). The average is taken over all three dimensions.
$\overline{v}_n^{therm}$, $\overline{v}_p^{therm}$	Average thermal velocity of electrons (n) and holes (p). The average is taken only over the dimensional of interest.
$\overline{v}_{bn}^{therm}$, $\overline{v}_{bp}^{therm}$	Average thermal velocity of electrons (n) and holes (p) located on the barrier side of the heterojunction. The average is taken only over the dimensional of interest.
v_{sn},v_{sp}	Saturation velocity for electrons (n) and holes (p).
W	Carrier energy.
W_n	Electron energy.
W_n	Hole energy.
W_j^e, W_i^h	Effective width of the wave functions for electron and hole in the jth and ith subbands, respectively.
x, y	Spatial coordinates used in the model. The x-axis is parallel to the quantum well. x is also used elsewhere as the Al composition in AlGaAs or the In composition in InGaAs.
α α_{qw}	Local loss coefficient due to loss other than interband recombination, and local loss in the quantum well.
α_{int}	Internal loss defined as the weighted average of the local loss α.
α_n, α_p	Constants used in doping dependent mobility functions for electrons (n) and holes (p).
α_{fn}, α_{fp}	Coefficients of the free carrier absorption for electrons (n) and holes (p) in the quantum well.
α_0	Absorption coefficient for regions outside the quantum well.
β, β_1, β_2	Complex eigenvalues of the wave equation and its real (1) and imaginary (2) parts. They are also considered effective indices.
α_{em}	Optical loss coefficient due to emission of light in the lasing mode.
α_{emj}	Optical loss coefficient due to emission of light in the jth longitudinal mode.
β_p	Constant used in the hole mobility equation.
Δ	Spin-orbit splitting of the valence band energy.
Δ_{ij}^a	Transition energy shift term in the Asada broadening model.
δ	Constant. Value is 1 for deep donors and 0 for deep acceptors when used to represent electrical charge. Also used as a δ-function.
ϵ_0	Dielectric constant of vacuum.
ϵ_{dc}	Relative DC or low frequency dielectric constant.
ϵ, ϵ_1, ϵ_2	Complex optical dielectric constant and its real (1) and imaginary (2) parts. ϵ is also used, under a completely context, for non-linear gain coefficient.

ϵ_2^g, ϵ_2^α	Imaginary parts of the dielectric constant corresponding to gain due to interband transition and loss due to other mechanisms, respectively.
ε	Strain.
Γ	Energy level broadening parameter, or the half width of the Lorentzian broadening function.
Γ_0	The maximum energy level broadening parameter, or the half width of the Lorentzian broadening function.
Γ_a	The energy level broadening parameter, or the half width of the Lorentzian broadening function, as used in the Asada's broadening model.
$\gamma_1, \gamma_2, \gamma_3$	Luttinger numbers.
γ	Constant relating the optical wave amplitude to electric field.
γ_n,γ_p	Adjustable constants for the electron (n) and hole (p) thermionic emission model, used to account for tunneling and other effects.
γ_{hn},γ_{hp}	Adjustable constants for the electron (n) and hole (p) thermionic emission model for heterojunctions, used to account for tunneling and other effects.
χ	Affinity.
χ	Affinity of a reference material.
λ	Optical wavelength.
λ_j	Optical wavelength of the jth longitudinal mode.
μ_n, μ_p	Mobility of electrons (n) and holes (p).
μ_{0n}, μ_{0p}	Low field electron (n) and hole (p) mobilities.
μ_{1n}, μ_{1p}	Minimum electron (n) and hole (p) mobilities.
μ_{2n}, μ_{2p}	Maximum electron (n) and hole (p) mobilities.
ν	Function used in the Fermi-Direct statistics.
ρ_i, ρ_j, ρ_{ij}	Density of states for the i th and the j th levels, and the reduced density of states for transition between the j th and the i th levels.
ϕ_n,ϕ_p	Quasi-Fermi potential for electrons (n) and holes (p).
ϕ_b	Schottky barrier height.
ρ_i^0, ρ_j^0, ρ_{ij}^0	Constants for the two-dimensional density of states for the i th and the j th levels, and reduced density of states for transition between the i th and the j th levels. For example $\rho_i^0 = m_i/\pi\hbar^2 t$.
σ_{nj}, σ_{pj}	Electron (n) and hole (p) capture cross sections of the j th deep trap.
τ	Intra-band scattering lifetime.
τ_{nj},τ_{pj}	Electron (n) and hole (p) life time due to the jth trap.
τ_h, τ_v	Hole-hole scattering life time for the Asada gain broadening model. $\tau_h = 2\tau_v$.
ω	Angular frequency of the optical wave.

ω_a,ω_β	SOR parameters used for the wave equation.
ξ	Function used in the Fermi-Dirac statistics.
ζ, ζ_n, ζ_p	Variable related to the effective masses. ζ is also used for half the shear strain energy for a different context.
$< \ >$	Average with $\mid W \mid^2$ as weight.

Appendix G

POST-PROCESSING & PLOTTING VARIABLES

G.1 Bias-dependent Variables

Bias-dependent variables refer to quantities characterizing the whole device rather than a local value at a particular mesh point. They can be plotted using **plot_scan** or analyzed using commands such as **fourier_power**.

Table G.1: Bias-dependent variables available for plotting

Variable	Definition
all_mode_power	Emitted laser power for all lateral modes in units of mW (LASTIP) [1] .
all_mode_micav_power	Emitted microcavity laser power for all lateral modes (PICS3D). Units in W/m for 2D and W for 3D.
bottom_led_power	LED power emitted from the bottom of the device.
bottom_led_power_te	LED power emitted from the bottom of the device (TE mode).
bottom_led_power_tm	LED power emitted from the bottom of the device (TM mode).
Continued on next page	

[1] Starting with the 2012 version, this value is no longer scaled by a factor of two to account for implied symmetry

Table G.1 – continued from previous page

Variable	Definition
current_i	Total current (sum of electron, hole and displacement current) at electrode i.
disp_curr_i	Displacement current (time derivative of the displacement vector) at electrode i.
effective_index	Real effective refractive index for a waveguide device.
efficiency	Laser emission power efficiency (or slope efficiency) (LASTIP).
elec_curr_i	Electron current at electrode i
hole_curr_i	Hole current at electrode i.
laser_current_i	Total injection current into a laser diode; this value is multiplied by the device length to convert the current into mA (LASTIP)[1].
laser_power	Emitted laser power for a single mode in units of mW (LASTIP)[1].
led_curr_effi	LED luminosity/current efficiency (cd/A) (APSYS).
led_effi	LED internal quantum efficiency (APSYS).
led_effx1	Similar to led_effz1.
led_effx2	Similar to led_effz1.
led_effy1	Similar to led_effz1.
led_effy2	Similar to led_effz1.
led_effz1	LED power extraction efficiency from facet 1 (i.e., left or bottom facet) facing the z-direction based on the (cubic) broad-area LED model. This result may be different from that calculated from the ray tracing model.
led_effz2	Similar to led_effz1 above.
led_power	Total amount of emitted optical power from all facets of an LED (APSYS) in the (cubic) broad-area LED model. This may be different from the optical power calculated from the ray tracing model.
led_power_te	Total amount of emitted optical power (TE mode) from all facets of an LED (APSYS) in the (cubic) broad-area LED model. This may be different from the optical power calculated from the ray tracing model.

Continued on next page

Table G.1 – continued from previous page

Variable	Definition
led_power_tm	Total amount of emitted optical power (TM mode) from all facets of an LED (APSYS) in the (cubic) broad-area LED model. This may be different from the optical power calculated from the ray tracing model.
led_spon_power	Total power generated from spontaneous emission events inside a LED (APSYS).
led_spon_power_te	Total power (TE mode) generated from spontaneous emission events inside a LED (APSYS).
led_spon_power_tm	Total power (TM mode) generated from spontaneous emission events inside a LED (APSYS).
light	Incident light power to a photodetector, modulator, SOA or any other light-sensitive device.
micav_laser_power	Emitted microcavity laser power for specified mode (PICS3D). Units in W/m for 2D and W for 3D.
more_bias_vari(i=1..9)	Special variable created by using various statements such as **bias_output_near_point**.
pd_efficiency	External quantum efficiency of a photodetector [2].
pd_responsivity	Responsivity of a photodetector[2].
peak_gain	Peak model gain in a waveguide device with lateral mode and active layer(s). It is the modal gain at peak wavelength.
rtg_left_power	Left power emission in RTG model (PICS3D).
rtg_left_power_allmode	All-mode left power emission in RTG model (PICS3D).
rtg_photon_num	Photon number in RTG model (PICS3D).
rtg_photon_num_allmode	All-mode photon number in RTG model (PICS3D).
rtg_right_power	Left facet power emission in RTG model (PICS3D).
rtg_right_power_allmode	Right facet power emission in RTG model (PICS3D).
rtg_surf_power	Surface emitting power (for 2nd order grating) in RTG model (PICS3D).
rtg_surf_power_allmode	All-mode surface emitting power (for 2nd order grating) in RTG model (PICS3D).

Continued on next page

[2]This term does not include dark current effects

Table G.1 – continued from previous page

Variable	Definition
rtg_wavelength	Wavelength in RTG model (PICS3D).
rtg_wave_phase	Phase shift of input wave in RTG model (PICS3D).
rtg_2facet_power	2-facet power for a particular mode (PICS3D).
rtg_2facet_power_allmode	All-mode power for a particular mode (PICS3D).
rtg_efficiency	External quantum efficiency for laser device (PICS3D).
rtg_ase_left	Amplified spontaneous emission from the left facet for a particular longitudinal mode (PICS3D).
rtg_ase_left_allmode	Amplified spontaneous emission from the left facet for all modes (PICS3D).
rtg_ase_right	Amplified spontaneous emission from the right facet for a particular longitudinal mode (PICS3D).
rtg_ase_right_allmode	Amplified spontaneous emission from the right facet for all modes (PICS3D).
rtg_ase_2facet_allmode	Amplified spontaneous emission from both facets for all modes (PICS3D).
rtg_ase_efficiency	External quantum efficiency for amplified spontaneous emission source (PICS3D).
rtg_smsr	Side mode suppression ratio (PICS3D). This is based on the photon number of each mode, as defined in Appendix E.
rtg_wave_phase	Phase change of input wave at exit for a modulator/SOA (PICS3D).
sum_space_charge	Sum of space charge over a region.
temp_max	Maximum temperature in device.
time	Time in picoseconds used in a transient simulation.
top_led_power	LED power emitted from the top of the device .
top_led_power_te	LED power (TE mode) emitted from the top of the device .
top_led_power_tm	LED power (TM mode) emitted from the top of the device .
total_joule_heat	Integrated total Joule heat.
total_optic_heat	Total heat due to optical absorption.
total_peltier_heat	Integrated total Peltier heat.
total_recomb_heat	Integrated total recombination heat.
total_thomson_heat	Integrated total Thomson heat.

Continued on next page

Table G.1 – continued from previous page

Variable	Definition
tunneling_enhancement	Maximum tunneling enhancement/increase to the classical drift-diffusion current.
volt_res_i	The voltage at boundary i in a mixed-mode simulation, including the external resistor.[3]
voltage_i	Voltage at electrode i.
wavelength	Laser emission wavelength at peak modal gain (LASTIP).
wavelength_scan	Wavelength of the optical pump.

G.2 Scalar Variables

Scalar variables are structural quantities defined for every mesh point of the device. They depend on also depend on bias but are only printed at certain data sets, as defined in the **scan** statement. These quantities may be plotted using statements such as **plot_1d** and **plot_2d**.

Table G.2: Scalar variables available for plotting

Variable	Definition
absorption	background optical absorption not including gain/loss from active layers.
acceptor_conc	acceptor concentration.
acceptor_conc2	acceptor concentration (level2).
all_conc	both electron and hole.
all_heat	all heat sources.
band	energy band. (for plot_1d and lplot_xy)
bulk_bandgap	bandgap of bulk material; in a QW region, the heavy hole bandgap is used even if the light hole bandgap is smaller.
bulk_elec_conc	bulk elec. concentration in QW system.
bulk_hole_conc	bulk hole concentration in QW system.
central_area	local surface area around mesh point.

Continued on next page

[3] The definitions voltage_i and volt_res_i switched in v.2015 due to various improvements in the mixed-mode model.

Table G.2 – continued from previous page

Variable	Definition
ch_val_band	crystal field hole valence band edge for wurtzite structures.
compact_iqe	localized photon conversion efficiency: $\frac{R_sp(x,y,z)}{J(x,y,z)}$.
cond_band	conduction band.
disp_curr_mag	magnitude of displacement current.
disp_curr_x	displacement current x-component.
disp_curr_y	displacement current y-component.
disp_curr_z	displacement current z-component.
donor_conc	donor concentration.
donor_conc2	donor concentration (level2).
efb_cond_band	conduction band in effective QW miniband model.
efb_val_band	valence band in effective QW miniband model.
efm_cond_band	conduction band in effective bulk medium model.
efm_val_band	valence band in effective bulk medium model.
elec_conc	electron concentration.
elec_curr_mag	magnitude of electron current.
elec_curr_x	electron current x-component.
elec_curr_y	electron current y-component.
elec_curr_z	electron current z-component.
elec_diff	magnitude of the electron concentration gradient $\|\nabla n\|$.
elec_diff_gain	electron differential gain (cm^2).
elec_imref	electron IMREF.
elec_mobility	average nodal electron mobility.
exciton_dipole_source	exciton dipole source density.
exciton_singlet	singlet exciton density.
exciton_triplet	triplet exciton density.
fdfd_efield_r	optical field component E_r from microcavity vectorial mode solver.
fdfd_efield_z	optical field component E_z from microcavity vectorial mode solver.
fdfd_hfield_phi	optical field component H_ϕ from microcavity vectorial mode solver.
field_mag	electrical field magnitude.
field_x	electric field x-component.
field_y	electric field y-component.
heat_flux_density_mag	heat flux magnitude.
heat_flux_density_x	heat flux x-component.

Continued on next page

Table G.2 – continued from previous page

Variable	Definition
heat_flux_density_y	heat flux y-component.
heat_flux_density_z	heat flux z-component.
hole_conc	hole concentration.
hole_curr_mag	magnitude of hole current.
hole_curr_x	hole current x-component.
hole_curr_y	hole current y-component.
hole_curr_z	hole current z-component.
hole_diff_gain	hole differential gain (cm^2).
hole_imref	hole IMREF.
hole_mobility	average nodal hole mobility.
impact_alpha_n	electron impact ionization coefficient.
impact_alpha_p	hole impact ionization coefficient.
impact_ionization	total impact ionization rate.
incident_power	same as optical_energy.
index_change	refractive index change from equilibrium value.
interband_pumping	optically pumped interband absorption.
interface_length	equivalent interface length around mesh point; this value only considers ALL mesh triangle edges touching something other than the material number of local the mesh point.
joule_heat	Joule heat source.
kpref_val_band	reference valence band edge for wurtzite structures.
lattice_temp	lattice temperature.
lh_val_band	light hole valence band edge.
linear_acceptor_conc	acceptor concentration in linear scale.
linear_bulk_elec	bulk elec. concentration in QW system (linear scale).
linear_bulk_hole	bulk hole concentration in QW system (linear scale).
linear_donor_conc	donor concentration in linear scale.
linear_elec_conc	electron concentration in linear scale.
linear_heat	Heat source linear in current.
linear_hole_conc	hole concentration in linear scale.
linear_rec_elec	same as rec_elec_conc but in linear scale.
linear_rec_hole	same as rec_hole_conc but in linear scale.
linear_subb_elec	QW subband elec. concentration (linear scale).
linear_subb_hole	QW subband hole concentration (linear scale).
local_gain	local optical gain.

Continued on next page

Table G.2 – continued from previous page

Variable	Definition
lower_elec_imref	IMREF of the lower electron state in the split-state model
lower_hole_imref	IMREF of the lower hole state in the split-state model
lower_elec	electron concentration in the lower state in the split-state model
lower_hole	hole concentration in the lower state in the split-state model
magnetic_field_mag	magnitude of flux density (B) pseudo-vector.
magnetic_field_x	x-component of magnetic flux density (B) pseudo-vector.
magnetic_field_y	y-component of magnetic flux density (B) pseudo-vector.
magnetic_field_z	z-component of magnetic flux density (B) pseudo-vector.
magnetic_potential_mag	magnitude of magnetic vector potential (A).
magnetic_potential_x	x-component of magnetic vector potential (A).
magnetic_potential_y	y-component of magnetic vector potential (A).
magnetic_potential_z	z-component of magnetic vector potential (A).
material_num	internal material number.
negative_fix_charge	negative fixed space charge.
negf_elec_diff	same as elec_diff but for NEGF transport.
occup_trap	occupancy of trap.
optic_heat	optical heat source.
optical_energy	relative energy distribution scaled to incident light energy density.
optical_gen	optical generation rate induced by incident light.
optical_gen_led	optical generation rate in resonant-cavity model, induced by trapped spontaneous emission.
outer_interface_length	equivalent interface length around mesh point; this value only considers only the mesh triangle edges touching the outer edges of the simulation domain.
peltier_heat	Peltier heat source.
polar_vector	Total polarization vector; sum of spontaneous and strain-induced polarization.
piezo_stress_xx	Piezoelectric stress component in SAWAVE simulation.

Continued on next page

Table G.2 – continued from previous page

Variable	Definition
piezo_stress_xy	Piezoelectric stress component in SAWAVE simulation.
piezo_stress_xz	Piezoelectric stress component in SAWAVE simulation.
piezo_stress_yy	Piezoelectric stress component in SAWAVE simulation.
piezo_stress_yz	Piezoelectric stress component in SAWAVE simulation.
piezo_stress_zz	Piezoelectric stress component in SAWAVE simulation.
piezo_vector	Strain-induced polarization vector.
piezo_vector_external	Polarization vector due to external stress.
positive_fix_charge	positive fixed space charge.
potential	potential.
qdot_electron	electron concentration in the embedded quantum dot
qdot_hole	hole concentration in the embedded quantum dot
radiation_heat	radiation heat source term.
real_index	real index.
rec_elec_conc	electron concentration participated in recombination only as opposed to that in transport.
rec_hole_conc	hole concentration participated in recombination only as opposed to that in transport.
recomb_aug	Auger recombination rate.
recomb_heat	recombination heat source.
recomb_rad	radiative recombination rate (TE+TM).
recomb_rad_te	radiative recombination rate (TE).
recomb_rad_tm	radiative recombination rate (TM).
recomb_srh	SRH recombination rate.
recomb_st	stimulated recombination rate.
space_charge	net space charge.
spatial_grad_thermal_power_n	magnitude of spatial gradient of electron thermoelectric power ∇P_n; this terms controls the combined Peltier and Thomson heat.
spatial_grad_thermal_power_p	magnitude of spatial gradient of hole thermoelectric power ∇P_p this terms controls the combined Peltier and Thomson heat.
stress_xx	normal stress in x direction.
stress_xy	shear stress (xy).

Continued on next page

Table G.2 – continued from previous page

Variable	Definition
stress_xz	shear stress (xz).
stress_yy	normal stress in y direction.
stress_yz	shear stress (yz).
stress_zz	normal stress in z direction.
subb_elec_conc	QW subband elec. concentration.
subb_hole_conc	QW subband hole concentration.
temp_grad_thermal_power_n	magnitude of temperature-driven gradient for electron thermoelectric power ($\frac{\partial P_n}{\partial T}\nabla T$).
temp_grad_thermal_power_p	magnitude of temperature-driven gradient for hole thermoelectric power ($\frac{\partial P_p}{\partial T}\nabla T$).
thermal_power_n	electron thermoelectric power (P_n)
thermal_power_p	hole thermoelectric power (P_p)
thomson_heat	Thomson heat source.
total_curr_mag	magnitude of total current.
total_curr_x	total current x-component.
total_curr_y	total current y-component.
total_curr_z	total current z-component.
trap_conc	trap concentration.
trapped_elec_conc	trapped electron concentration
trap_level	trap level.
tunnel_enhancement	enhancement factor applied to drift-diffusion current in the tunneling and NEGF models.
val_band	valence (also heavy hole) band edge.
wave_2ndwavel	SOA Traveling Wave (2nd wavelength).
wave_real_part	Real part of the optical mode. Note that the optical wave is normalized to within a complex constant; only the spatial difference is significant.
wave_imag_part	Imaginary part of the optical mode. See above comment for normalization.
wave_intensity	relative wave intensity.
wave_intensity_all_modes	wave intensity of all modes.

G.3 Vectorial Variables

Vectorial variables, like scalar variables, are structural quantities defined at every mesh point of the device. The corresponding vector field may be plotted using

statements such as **plot_2d** and **vplot_xy**:

Table G.3: Vectorial variables available for plotting

Variable	Definition
elec_curr	electron current
hole_curr	hole current
disp_curr	displacement current
elec_field	electric field (static)
total_curr	total current
flowline_curr	flowline plot of total current

PARS OPTICS GROUP

Appendix H

SUPPORTED EXAMPLES

H.1 APSYS Examples

```
------------------------
\apsys_examples\A_tutorial
[title] Simulation of a camel diode: A tutorial.
[structure] Silicon camel diode.
[comment] Basic simulation procedures using APSYS.
------------------------
\apsys_examples\basic_silicon\bjt_3d
[title] Full 3D simulation of BJT.
[structure] 3D structure of BJT with edge effect.
[comment] The edge effect makes significant difference.
------------------------
\apsys_examples\basic_silicon\diode
[title] Turn-on characteristics of pn junction diode.
[structure] Simple p-n junction diode.
[comment] Set up of transient simulation.
------------------------
\apsys_examples\basic_silicon\hot_carrier
[title] Hot electron simulation.
[structure] N-type MOS transistor.
[comment] Different I-V is obtained if energy dep. mobility is used.
------------------------
\apsys_examples\basic_silicon\impact_mos
[title] Impact ionization effect in MOS.
[structure] N-type MOS transistor.
[comment] Break down effect due to impact ionization.
------------------------
```

```
\apsys_examples\basic_silicon\jfet_temperature
[title] JFET simulation.
[structure] Silicon JFET.
[comment] Set up of simulation at different temperatures.
------------------------
\apsys_examples\basic_silicon\jfet_trap
[title] Deep level traps in JFET.
[structure] Silicon JFET with deep traps.
[comment] Deep trap model at low temperature.
------------------------
\apsys_examples\basic_silicon\mini_bjt
[title] Small size BJT.
[structure] Silicon bipolar junction transistor.
[comment] Show how to generate a family of I-V curves.
------------------------
\apsys_examples\basic_silicon\npn_indium
[title] NPN silicon BJT with indium doped base
[structure] N-P-N with boron and indum doped base.
[comment] Show how to use two p dopings with different ionization energies.
------------------------
\apsys_examples\basic_silicon\SOI_impact
[title] Impact ionization in SOI.
[structure] SOI with gate length of 1.6 um and poly gate.
[comment] Switch to current control before snap back in I-V curve.
------------------------
\apsys_examples\basic_silicon\TFT_3D
[title] 3D Simulation of a-SI TFT device.
[structure] Strip cell with mirror symmetry in x-direction and z-direction.
[comment] Show how to simulate 3D a-Si TFT by Csuprem and apsys.
------------------------
\apsys_examples\basic_silicon\thermal_npn
[title] Thermal analysis of a BJT.
[structure] Silicon NPN BJT
[comment] Substantial difference when self-heating effect is included.
------------------------
\apsys_examples\basic_silicon\thyristor
[title] Simulation of forward I-V characteristics of a thyristor
[structure] silicon p-n-p-n
[comment] Use of special convergence techniques to get the S-shape I-V curve
------------------------
\apsys_examples\CCD_CIS\CCD
[title] Simplified CCD structure
[structure] Silicon CCD detector
```

```
[comment] Basic simulation procedures using APSYS.
------------------------
\apsys_examples\CCD_CIS\PD_CIS
[title] 2D CMOS image sensor with photodiode
[structure] 0.2 um CMOS with .layer format.
[comment] Demonstration of how clock cycle is controlled in CCD/CIS
------------------------
\apsys_examples\CCD_CIS\PG_CIS
[title] CIS with a photogate
[structure] PG+TX-MOS+Reset-MOS
[comment] Classic design of CIS based on photogate as light integrator
------------------------
\apsys_examples\EAM\CQW_modulator
[title] Simulation of a vertical coupled quantum well modulator.
[structure] InGaAs/InAlAs CQW modulator.
[comment] Detailed analysis and comparison with experiments
------------------------
\apsys_examples\EAM\franz_keldysh
[title] Franz-Keldysh in Bulk DH Device.
[material] Bulk InGaAsP at 1.3 micron bandgap.
[structure] 1D p-n junction EAM
------------------------
\apsys_examples\EAM\Mach-Zehnder-2D
[title] 2D analysis of InP based MQW Mach-Zehnder modulator
[structure] MQW ridge waveguide
[keywords] Quantum confined Stark effect (QCSE), Pockels' effect, index change
------------------------
\apsys_examples\EAM\QCSE
[title] Electro-Absorption modulator simulation.
[structure] SQW EAM of AlGaAs.
[comment]  Use of "self_consistent" needed to update potential profile.
------------------------
\apsys_examples\EAM\QW_exciton
[title] Exciton model for Electro-Absorption Modulator spectrum
[structure] SQW EAM of AlGaAs.
[comment]  Use of "self_consistent" and "exciton".
------------------------
\apsys_examples\ESD\impact
[title] Electrostatic discharge (ESD) simulation
[structure] A simple silicon diode
[comment] Basic ESD simulation procedures using APSYS.
------------------------
\apsys_examples\ESD\self_heating
```

```
[title] Electrostatic discharge (ESD) thermal simulation
[structure] A simple silicon diode
[comment] Basic ESD simulation procedures using self-heating.
------------------------
\apsys_examples\FDTD\2d_lense
[title] FDTD simulation of 2D lense structure
[device] 2D lense shape with light incidence from top
[comment] Illustrates focusing effect
------------------------
\apsys_examples\FDTD\2d_slab
[title] FDTD material absorption in a simple 2D slab
[device] Simple silicon slab with light incidence from top
[comment] Illustrates basic parameter setting for FDTD modeling
------------------------
\apsys_examples\FDTD\3D_lense_PD3x3
[title] FDTD material absorption in a simple 2D slab
[device] Simple silicon slab with light incidence from top
[comment] Illustrates basic parameter setting for FDTD modeling
------------------------
\apsys_examples\FDTD\3D_lense_PD3x3
[title] 3D micro lense simulation by FDTD
[structure] 3D lense constructure by multiple segments
[comment] use of MEEP package
------------------------
\apsys_examples\FDTD\3D_pyramid_cell
[title] 3D textured thinfilm cell simulation using FDTD
[structure] muc-Si cell with pyramid texture
[comment] Use of mesh plane bending to obtain  3D texture
------------------------
\apsys_examples\FDTD\box3d_FDTD
[title] FDTD interface to APSYS
[structure] Simple 3D box of silicon with an Al contact pad
[comment] Illustrates use of FDTD interface through a simple 3D box
------------------------
\apsys_examples\FDTD\inside_src
[title] FDTD with inside light source
[structure] silicon device
[remark] place light source inside of device
------------------------
\apsys_examples\FDTD\Lense_PD_FDTD
[title] Lensed photodetector simulation using FDTD
[structure] Silicon PD with lense with SiO2 AR coating
[comment] Start from .layer and use GeoEditor to construct lense
```

```
------------------------
\apsys_examples\FDTD\thick_glass
[title] FDTD simulation with light source through thick glass
[structure] p-i-n silicon solar cell with 3mm glass layer.
[comment] Demo of FDTD simulation of solar cell with thick glass cover.
------------------------
\apsys_examples\HBT\hbt_ingaas
[title] Simulations of compound semiconductor HBT.
[structure] InGaAs/InP.
[comment] Basic procedures of HBT simulation.
------------------------
\apsys_examples\HBT\hbt0_tunnel
[title] InGaAs/InP HBT models.
[structure] HBT based in InGaAs/InP material.
[comment] Effect of tunneling in HBT simulation.
------------------------
\apsys_examples\HBT\sige_hbt
[title] SiGe HBT simulation.
[structure] SiGe HBT.
[comment] Set up of DC and AC simulation for SiGe HBT.
------------------------
\apsys_examples\HBT\sige_hbt2
[title] Analysis of SiGe HBT Performance
[structure] Selective-epitaxial SiGe
[comment] Study of bandgap narrowing effects in SiGe HBT.
------------------------
\apsys_examples\HEMT_and_FET\GaAs_HEMT\hemt3d
[title] GaN/AlGaN HEMT 3D structure with tapered barrier layer
[structure] GaN/AlGaN HEMT with deep level traps.
[comment] Use of y_taper_line to set up taper in 3D HEMT application
------------------------
\apsys_examples\HEMT_and_FET\GaAs_HEMT\impact_ionization
[title] AlGaAs HEMT simulation with selfconsistent method.
[structure] GaAs/AlGaAs HEMT with 2D gas system.
[comment] Use of "complex MQW" to define the 2D gas.
------------------------
\apsys_examples\HEMT_and_FET\GaAs_HEMT\MM_HEMT
[title] Metamorphic InAlAs/InGaAs/GaAs HEMT.
[structure] Metamorphic InAlAs/InGaAs/GaAs HEMT with delta doping.
[comment] Heavily doped layer as the so-called delta-doping layer.
------------------------
\apsys_examples\HEMT_and_FET\GaN_HEMT\AlGaNGaN_HEMT_Brkdwn
[title] High breakdown voltage AlGaN/GaN HEMT.
```

```
[structure] AlGaN/GaN HEMT with magnesium doping layer under 2-DEG channel.
[comment] Including the following features;
------------------------
\apsys_examples\HEMT_and_FET\GaN_HEMT\AlGaN-hemt
[title] AlGaN HEMT simulation using piezo surface charge and substrate trap.
[structure] GaN/AlGaN HEMT with 2D gas system.
[comment] Piezo surface charge and substrate trap are varied to fit experiment.
------------------------
\apsys_examples\HEMT_and_FET\GaN_HEMT\HEMT_gate_leakage
[title] Simulation of gate leakage in GaN HEMT
[structure] GaN/AlGaN/GaN/Schottky
[comment] To study condition for gate leakage
------------------------
\apsys_examples\HEMT_and_FET\GaN_HEMT\HEMT_nano
[title] Nano scale AlGaN HEMT simulation
[structure] GaN/AlGaN QW HEMT
[comment] Demonstrate the set up of a D-mode AlGaN HEMT
------------------------
\apsys_examples\HEMT_and_FET\GaN_HEMT\HEMT_nano_emode
[title] Enhancement mode nano scale AlGaN HEMT simulation
[structure] GaN/AlGaN QW HEMT
[comment] Demonstrate the use of local adjustment of polarization charges
------------------------
\apsys_examples\HEMT_and_FET\GaN_HEMT\HEMT_selfcs
[title] AlGaAs HEMT simulation with selfconsistent method.
[structure] GaAs/AlGaAs HEMT with 2D gas system.
[comment] Use of "complex MQW" to define the 2D gas.
------------------------
\apsys_examples\HEMT_and_FET\GaN_HEMT\hemt_suprem_quantum
[title] Process and device simulation of GaN-HEMT with quantum effects
[structure] GaN/AlGaN HEMT with CSuprem simulation
[purpose] To provide a template for CSuprem and APSYS simulation of HEMT
------------------------
\apsys_examples\HEMT_and_FET\GaN_HEMT\hot_traps\no_traps
[title] GaN/AlGaN HEMT hot-carrier effects
[structure] GaN/AlGaN HEMT with p(-) substate
[comment] Purpose is to compare with similar cases with deep level traps.
------------------------
\apsys_examples\HEMT_and_FET\GaN_HEMT\hot_traps\traps
[title] GaN/AlGaN HEMT hot-carrier trapping effects
[structure] GaN/AlGaN HEMT with deep level traps.
[comment] Deep level traps in substrate and SiN passivated surface
------------------------
```

```
\apsys_examples\HEMT_and_FET\other_FET\broken_gap_FET
[title] GaSb/InGaAs broken bandgap FET
[structure]  GaSb/InGaN TJ at source
[comment] Use of equivalent mobility to model broken gap TJ
------------------------
\apsys_examples\HEMT_and_FET\other_FET\GTFET
[title] Simulation of a Green Tunnel Field Effect
Transistor (GTFET) with large turn-of swing
[model] Tunnel junction with PAT
------------------------
\apsys_examples\HEMT_and_FET\other_FET\PAT_in_TFET
[title] Tunnelling Junction FET simulation
[structure] vertical silicon FET
[remark] Demonstration of phonon-assisted and trap-assisted tunnelling
------------------------
\apsys_examples\HEMT_and_FET\other_FET\Sb_FET
[title] Sb based HEMT simulation
[structure] AlSb/InAs/AlSb/AlGaSb layers
[comment] demonstration of setting up a Sb based HEMT
------------------------
\apsys_examples\LED_GaN_MQW\2d\AlGaN_block
[title] Compare AlGaN blocking layer design with and without supperlattice
[structure] GaN based MQW system with AlGaN SL blocking layer
[comment] Use of Piezo charge and quantum tunneling in blocking layer
------------------------
\apsys_examples\LED_GaN_MQW\2d\AlGaN_block\no_sl
[title] AlGaN blocking layer design (I): no superlattice
[structure] GaN based MQW system with AlGaN blocking layer
[comment] Use of Piezo charge and quantum tunneling in blocking layer
------------------------
\apsys_examples\LED_GaN_MQW\2d\AlGaN_block\with_sl
[title] AlGaN blocking layer design (II): with superlattice
[structure] GaN based MQW system with AlGaN SL blocking layer
[comment] Use of Piezo charge and quantum tunneling in blocking layer
------------------------
\apsys_examples\LED_GaN_MQW\2d\complex_well
[title] Use of complex macros to engineer QW of LED
[structure] GaN/AlGaN based MQW system.
[comment] We compare different ways to define a non-symmetric QW
------------------------
\apsys_examples\LED_GaN_MQW\2d\deep_wells\non_seq_trap
[title] InGaN MQW LED with deep wells.
[structure] GaN based MQW system with AlGaN blocking layer
```

```
[comment] IQE roll-off even for deep wells and high EBL
------------------------
\apsys_examples\LED_GaN_MQW\2d\deep_wells\non_sequential
[title] InGaN MQW LED with deep wells.
[structure] GaN based MQW system with AlGaN blocking layer
[comment] IQE roll-off even for deep wells and high EBL
------------------------
\apsys_examples\LED_GaN_MQW\2d\deep_wells\q_trap
[title] InGaN MQW LED with deep wells.
[structure] GaN based MQW system with AlGaN blocking layer
[comment] IQE roll-off even for deep wells and high EBL
------------------------
\apsys_examples\LED_GaN_MQW\2d\dome
[title] Epoxy domed LED simulation
[structure] GaN LED with epoxy dome.
[comment]  Demonstration of setting up ray tracing with dome-packaging.
------------------------
\apsys_examples\LED_GaN_MQW\2d\InGaN
[title] Piezo and thermal effects in LED.
[structure] GaN based MQW system.
[comment] L-I curves would role off due to thermal effect
------------------------
\apsys_examples\LED_GaN_MQW\2d\triangle_well
[title] Triangle quantum well in LED modeled as graded complex layers.
[material] InGaN/GaN with graded InGaN
[comment] To show how to construct a triangle well using cx-InGaN
------------------------
\apsys_examples\LED_GaN_MQW\2d\tunnel_junction
[title] use tunnel junction as p-contact in LED.
[structure] GaN based MQW system.
[comment] demonstrate how to use TJ in LED
------------------------
\apsys_examples\LED_GaN_MQW\2d\tunnel_junction\with_TJ
[title] use tunnel junction as p-contact in LED.
[structure] GaN based MQW system.
[comment] demonstrate how to use TJ in LED
------------------------
\apsys_examples\LED_GaN_MQW\2d\tunnel_junction\without_TJ
[title] use tunnel junction as p-contact in LED.
[structure] GaN based MQW system.
[comment] This project has turned off TJ model to use as a reference
------------------------
\apsys_examples\LED_GaN_MQW\3d_fancy\circle
```

```
[title] 3D simulation of LED with circular contacts
[structure] geometry defined by circulr masks
[purpose] templates for simulating devices with complicated geometry
------------------------
\apsys_examples\LED_GaN_MQW\3d_fancy\interdigi
[title] Simulation of an LED with interdigital electrodes
[structure]  LED with interdigital electrodes
[comment] Demonstrate how to set up complicated eletrodes using .layer
------------------------
\apsys_examples\LED_GaN_MQW\3d_fancy\spiral
[title] 3D simulation of LED with spiral-shaped contacts
[structure] geometry defined by mask data
[purpose] templates for simulating devices with complicated geometry
------------------------
\apsys_examples\LED_GaN_MQW\3d_fancy\star
[title] 3D simulation of LED with star shaped contacts
[structure] geometry defined by mask data
[purpose] templates for simulating devices with complicated geometry
------------------------
\apsys_examples\LED_GaN_MQW\3d_wellxz\LED_maskeditor
[title] LED IQE simulation 2010
[stucture] MQW InGaN/GaN LED with EBL
[remark] IQE sensitive to band offset
------------------------
\apsys_examples\LED_GaN_MQW\3d_wellxz\LED_maskeditor
[title] LED simulation set up using MaskEditor/CSuprem
[stucture] MQW InGaN/GaN LED with EBL
[remark] Use of Csuprem / MaskEditor for mesh
------------------------
\apsys_examples\LED_GaN_MQW\3d_wellxz\side_contact3d
[title] 3D AlGaN/AlGaN MQW LED simulation with ray-tracing
[structure] AlGaN/AlGaN MQW emitting at 0.34 micron.
[comment] Use of self-consistency model in multiple segment 3D structure
------------------------
\apsys_examples\LED_GaN_MQW\barrier_emit
[title] LED barrier emission
[structure]  InGaN/GaN MQW.
[comment]  Study of barrier emission spectrum
------------------------
\apsys_examples\LED_GaN_MQW\IQE_droop
[title] LED IQE simulation 2010
[stucture] MQW InGaN/GaN LED with EBL
[remark] IQE sensitive to band offset
```

```
------------------------
\apsys_examples\LED_GaN_MQW\LED_external_stress\reference
[title] LED IQE simulation 2010
[stucture] MQW InGaN/GaN LED with EBL
[remark] IQE sensitive to band offset
------------------------
\apsys_examples\LED_GaN_MQW\LED_external_stress\with_stress
[title] LED IQE simulation 2010
[stucture] MQW InGaN/GaN LED with EBL
[remark] IQE sensitive to band offset
------------------------
\apsys_examples\LED_GaN_MQW\mplane_LED
[title] InGaN/GaN QW mplane LED simulation using full k.p theory
[material]  InGaN/GaN
[device]  InGaN/GaN LD
------------------------
\apsys_examples\LED_GaN_MQW\selfconsistent_onetime
[title] New method of selfconsistent simulation
[stucture] MQW InGaN/GaN LED with EBL
[remark] Three stages of simulation to achieve selfconsistency
------------------------
\apsys_examples\LED_GaN_MQW\split_local_trapping
[title]   Introducing split QW state local carrier trapping model

This directory contains examples to demo. the new split-QW state local
------------------------
\apsys_examples\LED_GaN_MQW\split_local_trapping\AlGaN_local
[title] Split state model in LED with AlGaN blocking layer
[structure] GaN based MQW system with AlGaN blocking layer
[comment] Use of split state model
------------------------
\apsys_examples\LED_GaN_MQW\split_local_trapping\AlGaN_nonlocal
[title] Split state model in LED with AlGaN blocking layer
[structure] GaN based MQW system with AlGaN blocking layer
[comment] Use of split state model
------------------------
\apsys_examples\LED_GaN_MQW\split_local_trapping\GaAs_local
[title] Demonstrate of local split state trapping model.
[structure] Simple MQW GaAs/AlGaAs LED
[comment] Use of large tau causes IQE droop, even in GaAs LED.
------------------------
\apsys_examples\LED_GaN_QDOT\QD_4sizes
[title] InGaN based LED with quantum dots.
```

```
[structure] MQW-QDOT InGaN/AlGaN LED.
[comment] Procedures of simulation
------------------------
\apsys_examples\LED_GaN_QDOT\QD_life
[title] Non-radiative lifetime effect in InGaN LED with quantum dots.
[structure] MQW-QD InGaN/AlGaN LED.
[comment] Study of carrier lifetime within QD effect.
------------------------
\apsys_examples\LED_general\BCLED\OneStage
[title] OneStage bipolar cascade LED LED.
[material] InGaAs/GaAs MQW
[structure] tunnel junction is used.
------------------------
\apsys_examples\LED_general\BCLED\PIN
[title] PIN LED.
[material] InGaAs/GaAs MQW
[structure] Simple PIN structure.
------------------------
\apsys_examples\LED_general\dome2d_rt3d
[title] 3D Ray Tracing structure w/dome
[structure] Simple InGaAlP/GaAs LED with external
dome structure to enhance extraction efficiency
------------------------
\apsys_examples\LED_general\led_absorption_each_well
[title] Extraction of absorption spectrum for each well in MQW LED
[stucture] MQW InGaN/GaN LED with EBL
[remark] Use of output_more_spectrum and plot_more_spectrum
------------------------
\apsys_examples\LED_general\pumped_led
[title] GaN based LED with optical pumping.
[structure] MQW GaN LED.
[comment] Demo of MQW LED simulation with incident light.
------------------------
\apsys_examples\LED_general\RT_TE_TM
[title] Simulation TE/TM raytracing in a LED
[structure] InGaN LED
[comment]  shows how to plot both TE and TM from LED
------------------------
\apsys_examples\LED_general\Textured_LED
[title] Simulation of a LED with textured surface.
[structure] MQW LED with star-shaped contact.
[comment] A simulation by cooperation of Csuprem,Apsys,FDTD and RT3D.
------------------------
```

```
\apsys_examples\LED_general\tunnel_spectrum\no_sl
[title] AlGaN blocking layer design (I): no superlattice
[structure] GaN based MQW system with AlGaN blocking layer
[comment] Use of Piezo charge and quantum tunneling in blocking layer
------------------------
\apsys_examples\LED_general\tunnel_spectrum\with_sl
[title] AlGaN blocking layer design (II): with superlattice
[structure] GaN based MQW system with AlGaN SL blocking layer
[comment] Use of Piezo charge and quantum tunneling in blocking layer
------------------------
\apsys_examples\mixed_mode\MOSFET
[title] Mixmode simulation of external circuit including a MOSFET
[structure] Simple meshed silicon resistor connected to a test.cir file.
[comment] CuteSpice command drives the circuit simulator capatible with SPICE
------------------------
\apsys_examples\NAND_flash_memory
[title] 2D simulation of multi-cell NAND flash memory
[structure] 50nmx50nm multi-cell flash memory line
[remark] real device simplified to two select gates and 4 pass gates
------------------------
\apsys_examples\nanowire\nanowire_bulk
[title] A simple cylindrical bulk nanowire of InGaN/GaN
[material] wurtzite bulk active layer of InGaN used
[comment] For more complicated 3D nanowire structure SemiCrafter is recommended
------------------------
\apsys_examples\nanowire\nanowire_mqw
[title] Modeling nanowire with multiple crystal orientation.
[structure] InGaN/GaN/ZnO with ZnO as growth template
[remark]  Use of new type of 3D taper: symmetric polygon
------------------------
\apsys_examples\nanowire\nanowire_mqw_ito
[title] Modeling nanowire with multiple crystal orientation.
[structure] InGaN/GaN/ZnO with ZnO as growth template
[remark]  Use of new type of 3D taper: symmetric polygon
------------------------
\apsys_examples\nanowire\nanowire_multiplane
[title] Modeling nanowire with multiple crystal orientation.
[structure] InGaN/GaN/ZnO with ZnO as growth template
[remark]  Use of new type of 3D taper: symmetric polygon
------------------------
\apsys_examples\NEGF\FINFET_3D_SUPREM_NEGF
[title] 3D process and NEGF device simulation of FINFET
[device] FINFET
```

```
[structure] Si on SiO2, with nitride spacer
------------------------
\apsys_examples\NEGF\NEGF_nmos
[title] NEGF quantum ballistic transport model of NMOS
[structure] Nano scale MOSFET
[purpose]  To demonstrate the set up of NEGF method with drift-diffusion solver
------------------------
\apsys_examples\nonplanar_layer
[title] Setting up non-planar layer structure in simulation
[structure] GaAs/AlGaAs quantum well device
[comment] Use of modify_layer_height in .layer demonstrated
------------------------
\apsys_examples\OLED\AMOLED_yang
[title] OLED in AMOLED application
[structure] m-MTDATA/alpha-NPD/Alq3/LiF/Al/Ag
[comment] Comparison with published experimental data
------------------------
\apsys_examples\OLED\bilayer
[title] Bilayer OLED simulation
[material] NPB/Alq3 OLED
[demo] Use of metal on OLED to form microcavity; multiple active layers
------------------------
\apsys_examples\OLED\DCM_micro
[title] Microcavity analysis of an OLED with exciton diffusion model
[structure] Alq3/DCM:Alq3/TPD/ITO
[comment] To perform both electrical and optical analysis of an OLED stack
------------------------
\apsys_examples\OLED\EL_abs_spectra
[title] EL/Absorption Spectrum Model for Organic Light-Emitting Diode
[structure] Single active layers of Alq3 and Alq3:DCM
[comments] Use of Frenkel exciton model to generate EL/absorption spectrum
------------------------
\apsys_examples\OLED\EL_fit
[title] Fitting EL Spectrum of Organic Materials
[structure] Active layers of organic semiconductors
[comment] Explain the procedures involved in fitting to experimental EL data
------------------------
\apsys_examples\OLED\PIN_OLED
[title] Low-voltage electrolumiscent device
[structure] PIN Organic LED w/Alq3 active region
[comment] Reasonable agreement with experimental I-V and L-V curves
------------------------
\apsys_examples\OLED\pumped_oled
```

```
[title] Opticalled pumped OLED
[structure] TPD/CBP/F8BT/SiO2
[comment] Optical pumping to generate exciton only, not free carriers.
------------------------
\apsys_examples\OLED\RC-OLED
[title] OLED with resonant cavity effect
[structure] CuPc/TPD/Alq3/LiF
[comment] For the study of micro cavity effect in OLED
------------------------
\apsys_examples\OLED\tandem
[title] Tandem OLED with resonant cavity effect
[structure] Stacks of NPB, Alq3, DCJTB:Alq3, n-doped Alq3 and p-doped NPB
[comment] Use of tunnel junction to connect series of OLED
------------------------
\apsys_examples\OLED\tandem2
[title] Tandem OLED with multple active layers.
[structure] Stacks of NPB, Alq3, C545T:Alq3, n-doped Alq3 and p-doped NPB
[comment] Use of tunnel junction to connect series of OLED
------------------------
\apsys_examples\OLED\trilayer
[title] Trilayer OLED simulation
[material] NPB/Alq3/Alq3:DCM
[demo] Multiple active layers in OLED
------------------------
\apsys_examples\OLED\triplet_diff
[title] Simulation of a WOLED with exciton diffusion model
[structure] ITO/NPD/BCzVBi:CBP/CBP/PQIr:CBP/Ir(ppy)3:CBP/CBP/BCzVbi:CBP/BPhen
[comment] To demonstrate white OLED simulation with triplet diffusion
------------------------
\apsys_examples\OLED\triplet_diff\no_biex
[title] Simulation of a WOLED with exciton diffusion model
[structure] ITO/NPD/BCzVBi:CBP/CBP/PQIr:CBP/Ir(ppy)3:CBP/CBP/BCzVbi:CBP/BPhen
[comment] This example turns off biexciton quenching.
------------------------
\apsys_examples\OLED\triplet_diff\no_el_transfer
[title] Simulation of a WOLED with exciton diffusion model
[structure] ITO/NPD/BCzVBi:CBP/CBP/PQIr:CBP/Ir(ppy)3:CBP/CBP/BCzVbi:CBP/BPhen
[comment] This example turns off dopant-dopant energy transfer.
------------------------
\apsys_examples\OLED\triplet_diff\ref
[title] Simulation of a WOLED with exciton diffusion model
[structure] ITO/NPD/BCzVBi:CBP/CBP/PQIr:CBP/Ir(ppy)3:CBP/CBP/BCzVbi:CBP/BPhen
[comment] To demonstrate white OLED simulation with triplet diffusion
```

```
------------------------
\apsys_examples\other\3D_stacking\xyplane_attachment
[title] Attachment of multiple 3D object in mesh generation
[structure] Multiple 3D domain (M3DD)
[comment] Use 3d_attachment command to construct complicated 3D mesh
------------------------
\apsys_examples\other\amplifier
[title] Guided wave amplifier on y-z plane.
[structure] MQW InGaAsP/InP structure at 1.55 um.
[comment] Transient simulation of traveling wave amplifier.
------------------------
\apsys_examples\other\angled_interface
[title] Modeling angled interface
[device] InGaN based MQW
[comment] Angled cosine factor may be used to modify fixed charge.
------------------------
\apsys_examples\other\bulk_exciton
[title] Bulk GaAs absorption spectrum simulation using exciton model
[material] bulk GaAs QW at 0.83 um.
[comment] Manybody/exciton model as compared with experiment
------------------------
\apsys_examples\other\Esaki_junction
[title] Esaki tunnel junction
[structure] Ge tunnel junction
[comment] Based on 1957 paper in PhysRev.
------------------------
\apsys_examples\other\HB_Varactor
[title] Simulation of a  heterostructure barrier varactor.
[structure] AlGaAs/GaAs symmetrical heterostructure barrier varactor
[comment] Use of self-consistent, tunneling, complex-MQW models.
------------------------
\apsys_examples\other\import_gen_rate
[title] Simulation of Photodetector with Imported Optical Gen. Rate
[structure] Ridge waveguide SQW PD.
[comment] Direct use of carrier generation rate from 3rd party software
------------------------
\apsys_examples\other\InN_non_parabolic_surface
[title] Simulation of a InN solar cell
[structure] Mg-doped InN layer with an electrolyte layer
[comment] Space-charge output
------------------------
\apsys_examples\other\jdos_tail
[title] Adding tail states in joint density of states
```

```
[structure] 1D quantum well and bulk active layers.
[comment] Study of tail states, JDOS and shape of gain/PL spectrum.
------------------------
\apsys_examples\other\laser_heat
[title] Thermal transient simulation of laser lift-off process
[structure] Copper, GaN, amorphase GaN and sapphire
[comment] High power laser pulse is used to heat up GaN/sapphire interface
------------------------
\apsys_examples\other\lead_salt\bulk_leadsalt
[title] Bulk lead-salt active layer with non-parabolic band model
[structure] lead-salt PbSrSe
[comment] Use of generic-bulk macro for lead-salt
------------------------
\apsys_examples\other\lead_salt\lead_salt_cx_mqw
[title] Lead-salt MQW quantum well gain/spontaneous emisson model
[structure] complex MQW lead-salt emitting at 5 um.
[comment] Use of generic-complex-mqw macro for lead-salt
------------------------
\apsys_examples\other\lead_salt\lead_salt111
[title] Lead-salt MQW quantum well grown in 111 model
[structure] complex MQW lead-salt PbSrSe
[comment] Use of generic-complex-mqw macro for lead-salt
------------------------
\apsys_examples\other\mesfet
[title] Simulation of GaAs MESFET.
[structure] GaAs MESFET.
[comment] Set up of a basic MESFET simulation.
------------------------
\apsys_examples\other\miniband\basic_nin
[title] Use of mini-band transport model for superlattice in GaN-based lasers
[material] InGaN/AlGaN
[structure] 2D LD model
------------------------
\apsys_examples\other\QWIP\QWIP_ydir
[title] Simulation of GaAs/AlGaAs QWIP
[structure]  GaAs/AlGaAs QWIP with 40 wells.
[comment] Basic simulation procedures using APSYS.
------------------------
\apsys_examples\other\QWIP\QWIP_zdir_no_fdtd
[title] 3D Simulation of GaAs/AlGaAs QWIP with well on xy-plane
[structure]  GaAs/AlGaAs QWIP with 40 wells with well on xy-plane.
[comment] Basic simulation procedures using APSYS.
------------------------
```

```
\apsys_examples\other\radiation_heavy_ion
[title] Radiation model for heavy ion strikes
[structure] 1-um nMOSFET
[comment] command radiation_heavy_ion demonstrated
------------------------
\apsys_examples\other\RTD\RTD_selfcons
[title] RTD simulation using tunneling model with self-consistent treatment.
[structure] 1D RTD with two barriers.
[comment] Simultanous use of self-consistent MQW and tunneling model.
------------------------
\apsys_examples\other\schottky_tunnel
[title] Study of tunneling effects in a Schottky diode.
[structure] 1D Schottky diode.
[comments] Demo of tunneling effect in Schottky contact.
------------------------
\apsys_examples\other\STL_import
[title] 3D mesh import using the STL format
[structure] p-i-n silicon solar cell.
[comment] A simple example of STL file import procedure
------------------------
\apsys_examples\other\STL_import\STL_organizer_sample
[Title] How to use the STL_organizer program
[content] Program files and documents
[comment] A character user interface program
------------------------
\apsys_examples\other\trap_emission
[title] Trap emission model
[structure] GaN and AlGaN bulk HEMT-like structure
[purpose] To demonstrate trap emission via optical pumping
------------------------
\apsys_examples\other\ZnO
[title] ZnO/MgZnO  material model
[structure] ZnO/MgZnO QW
[comment] Setting up wurtzite macro for ZnO/MgZnO MQW
------------------------
\apsys_examples\PhCLED\InGaN
[title] PhCLED of InGaN/GaN
[structure] GaN based MQW system.
[comment] Air holes are used on top to enhanced extraction.
------------------------
\apsys_examples\PhCLED\with_DBR
[title] Modeling InGaAs/AlGaAs PhCLED/RCLED
[structure] Cylindrical RCLED with DBR and air hole photonic crystal
```

```
[comment] PhC. air holes are used to enhance light extraction
------------------------
\apsys_examples\photo_sensitive\APD_GB
[title] 2D simulation of InGaAsP/InP APD.
[structure] InGaAsP/InP APD.
[comment] Gain-bandwidth calculation using impulse transient simulation.
------------------------
\apsys_examples\photo_sensitive\APD_GB\apd
[title] 2D simulation of InGaAsP/InP APD.
[structure] InGaAsP/InP APD.
[comment] gain-bandwidth study
------------------------
\apsys_examples\photo_sensitive\APD_GB\apd2
[title] 2D simulation of InGaAsP/InP APD.
[structure] InGaAsP/InP APD.
[comment] gain-bandwidth study
------------------------
\apsys_examples\photo_sensitive\APD_GB\apd3
[title] 2D simulation of InGaAsP/InP APD.
[structure] InGaAsP/InP APD.
[comment] gain-bandwidth study
------------------------
\apsys_examples\photo_sensitive\CIGS
[title] CIGS detector simulation
[structure] Cu[In(1-x)Ga(x)]Se2 based detector.
[comment] Dark current simulation
------------------------
\apsys_examples\photo_sensitive\diffraction
[title] Single Slit Driffraction in Photo-sensitive Devices
[structure] MSM with 0.8 um inlet slit for light power.
[comment] Shows how to set up a single slit diffraction model.
------------------------
\apsys_examples\photo_sensitive\extern_optics
[title] 2D Simulation of Photodetector with External Light Profile
[structure] Ridge waveguide SQW PD.
[comment] Use of different forms of external optical field profile
------------------------
\apsys_examples\photo_sensitive\gaas_apd
[title] A simple 1D example of GaAs/AlGaAs APD
[structure] GaAs/AlGaAs.
[comment] Use of bandgap_reduction technique.
------------------------
\apsys_examples\photo_sensitive\GaN_PD\GaN_APD
```

```
[title] 1D simulation of separated-absorption-multipliation(SAM) GaN APD.
[structure] p-GaN/i-GaN(multiplication layer)/n-GaN/i-GaN(absorption layer)/n-GaN <--
------------------------
\apsys_examples\photo_sensitive\GaN_PD\GaN_PIN
[title] 1D simulation of p-i-n GaN APD.
[structure] p-GaN/i-GaN/n-GaN.
[comment] BV and transient simulation for GaN APD is demonstrated.
------------------------
\apsys_examples\photo_sensitive\inp_apd
[title] 2D simulation of InGaAsP/InP APD.
[structure] InGaAsP/InP APD.
[comment] DC and AC simulation with bandgap_reduction technique.
------------------------
\apsys_examples\photo_sensitive\kwon_gaas_apd
[title] A 1D example of GaAs/AlGaAs/GaAs APD for GBP study
[structure] GaAs/AlGaAs/GaAs.
[comment] Use of bandgap_reduction technique.
------------------------
\apsys_examples\photo_sensitive\msm_ac
[title] MSM PD simulation - AC photo-response analysis.
[structure] MSM with 0.5 um epi GaAs/AlGaAs/SI-GaAs.
[comment] AC response analysis for external light input.
------------------------
\apsys_examples\photo_sensitive\msm155
[title] Modeling optical absorption in active regions of PD.
[structure] InGaAs/InAlAs/InP.
[comment] Declaration of active region to allow interband transition model.
------------------------
\apsys_examples\photo_sensitive\PD_raytrace
[title] Photodetector Ray Tracing simulation of InGaAsP/InP APD.
[structure] InGaAsP/InP APD.
[comment] Use of ray tracing technique to generate photo-carriers.
------------------------
\apsys_examples\photo_sensitive\PD3D_external
[title] 3D Simulation of Photodetector with External Light Profile
[structure] Tapered ridge waveguide SQW PD.
[comment] Use of different forms of external optical field profile
------------------------
\apsys_examples\photo_sensitive\PIN_cyl
[title] Cylindrical PIN PD simulation.
[structure] InGaAs/InP PD
[comment] Demo of use of cylindrical coordinate system.
```

```
------------------------
\apsys_examples\photo_sensitive\SAGCM_APD

[title] 1D simulation of InP/InGaAs SAGCM APD.
[structure] InP/InGaAs SAGCM APD APD.
------------------------
\apsys_examples\photo_sensitive\silicon_apd\npip
[title] Basic impact ionization model for silicon.
[structure] Simple silicon pn-junctons.
[comment] Illustrate basic techniques of modeling break-down effects.
------------------------
\apsys_examples\photo_sensitive\silicon_apd\simple
[title] Basic impact ionization model for silicon.
[structure] Simple silicon pn-junctons.
[comment] Illustrate basic techniques of modeling break-down effects.
------------------------
\apsys_examples\photo_sensitive\surface
[title] Fermi level pinning effect due to surface states.
[structure] MSM PD.
[comment] Surface effects reduces dark current of PD.
------------------------
\apsys_examples\photo_sensitive\UV_PD
[title] GaN Ultra-violet photo-detector using
[structure] Bulk InGaN/AlGaN for 0.28 micron meter wavelength.
[comment] Demo of short wavelength photo-detector.
------------------------
\apsys_examples\photo_sensitive\waveguide_pd
[title] Waveguide photodetector simulation.
[structure] GRINCH-SQW-SCH waveguide PD.
[comment] Optical modes must be solved to compute carrier generation.
------------------------
\apsys_examples\photo_sensitive\zener_impact
[title] Junction breakdown due to Zener effect and impact ionization.
[structure] Silicon pn-junction.
[comment] Comparison of Zener and impact ionization effects.
------------------------
\apsys_examples\quantum_MOS\FINFET_qw3d
[title] Quantum effects in 3D-FINFET
[structure] FINFET with SiO2 as confining barriers.
[comment] Current injection in FINFET is 3D in nature.
------------------------
\apsys_examples\quantum_MOS\FINFETs\FINFET_bulk
[title] FINFET simulation using bulk model
```

```
[structure] FINFET with SiO2 as confining barriers.
[comment] For the purpose of comparison with the quantum finfet model.
------------------------
\apsys_examples\quantum_MOS\FINFETs\FINFET_qw
[title] Quantum effects in FINFET
[structure] FINFET with SiO2 as confining barriers.
[comment] Quantum effects in FINFET are significant.
------------------------
\apsys_examples\quantum_MOS\FINFETs\simple
[title] Quantum effects in a simple double gate MOSFET
[structure] FINFET with SiO2 as confining barriers.
[comment] Quantum effects in single segment of FINFET is considered.
------------------------
\apsys_examples\quantum_MOS\mobility_transverse
[title] Comparison of transverse field depedent mobility models.
[structure] Silicon N-MOS (80 nm)
[comment] Comparison of models named Intel1, Intel2 and Lombardi
------------------------
\apsys_examples\quantum_MOS\mobility_transverse\intel1
[title] Comparison of transverse field depedent mobility models.
[structure] Silicon N-MOS (80 nm)
[comment] Comparison of models named Intel1, Intel2 and Lombardi
------------------------
\apsys_examples\quantum_MOS\mobility_transverse\intel2
[title] Comparison of transverse field depedent mobility models.
[structure] Silicon N-MOS (80 nm)
[comment] Comparison of models named Intel1, Intel2 and Lombardi
------------------------
\apsys_examples\quantum_MOS\mobility_transverse\lombardi
[title] Comparison of transverse field depedent mobility models.
[structure] Silicon N-MOS (80 nm)
[comment] Comparison of models named Intel1, Intel2 and Lombardi
------------------------
\apsys_examples\quantum_MOS\mobility_transverse\reference
[title] Comparison of transverse field depedent mobility models.
[structure] Silicon N-MOS (80 nm)
[comment] Comparison of models named Intel1, Intel2 and Lombardi
------------------------
\apsys_examples\quantum_MOS\nmos_highk
[title] Simulation of a high-k MOSFET
[structure] Silicon MOSFET
[comment] Shows how to define new materials using custom
------------------------
```

```
\apsys_examples\quantum_MOS\npoly_experiment
[title] Quantum MOS gate current study
[structure] Silicon N-MOS with n+ polysilicon gate.
[comment] Use of direct tunneling model with quantum confinement in channel.
------------------------
\apsys_examples\quantum_MOS\poly
[title] Quantum effects in MOS with thin gate oxide and poly-gate
[structure] Silicon N-MOS with polysilicon gates
[comment] Quantum effects in MOS are significant.
------------------------
\apsys_examples\quantum_MOS\qw_mos
[title] Quantum effects in MOS with thin gate oxide
[structure] Silicon N-MOS and P-MOS
[comment] Quantum effects in MOS are significant.
------------------------
\apsys_examples\quantum_MOS\SGT
[title] Simulation of a surrounding gate transistor.
[structure] cylndrical MOSFET.
[comment] Quantum effects are included using APSYS.
------------------------
\apsys_examples\quantum_MOS\SGT3d
[title] Full 3D Simulation of a surrounding gate transistor.
[structure] cylndrical MOSFET without rotation symmetry.
[comment] Rotation symmetry broken by gate oxide being non-uniform.
------------------------
\apsys_examples\quantum_MOS\SONOS\300
[title] Quantum mechanical simulation for a SONOS-EEPROM
[structure] Semiconductor-oxide-nitride-oxide-semiconductor MOSFET
[comment] trapping/detrapping, quantum confinement/tunneling.
------------------------
\apsys_examples\quantum_MOS\SONOS\450
[title] Quantum mechanical simulation for a SONOS-EEPROM
[structure] Semiconductor-oxide-nitride-oxide-semiconductor MOSFET
[comment] trapping/detrapping, quantum confinement/tunneling.
------------------------
\apsys_examples\quantum_MOS\uniaxial_SSi
[title] Simulation of uniaxially stress MOSFET
[structure] n-MOSFET and p-MOSFET on 100 wafer with 100 or 110 channel
[comment] Both quantum well property and full MOSFET simulation are studied.
------------------------
\apsys_examples\quantum_MOS\uniaxial_SSi\n_qwell_100
[title] Study of mass induced mobility enhancement under uniaxial strain.
[structure] Strained Si with uniaxial tensile strain.
```

```
[comment] A series of preview files are used to conc. average valley mobility
------------------------
\apsys_examples\quantum_MOS\uniaxial_SSi\n_qwell_110
[title] Study of mass induced mobility enhancement under uniaxial strain.
[structure] Strained Si with uniaxial tensile strain [110].
[comment] A series of preview files are used to conc. average valley mobility
------------------------
\apsys_examples\quantum_MOS\uniaxial_SSi\nmos_nostrain
[title] Unstrained n-MOSFET as referece for strained silicon modeling
[structure] Silicon N-MOS
[comment] This structure has zero stress and exports data
------------------------
\apsys_examples\quantum_MOS\uniaxial_SSi\nmos_strain100
[title] Simulation of strained n-MOSFET [100]
[structure] Silicon N-MOS with poly gate
[comment] This structure has 1 GPa tensile stress
------------------------
\apsys_examples\RCLED\70well_noDBR
[title] Thermal simulation 70 pair MQW RCLED
[structure]  RCLED of InGaAlP emitting at 600 nm
[comment]  Use of current/density dependent lifetime
------------------------
\apsys_examples\RCLED\AlGaAs
[title] Model of resonant cavity LED (RCLED) with multilayer optic module
[structure] Cylindrical RCLED with implanted current confinement regions.
[comment] Note the input of DBR stacks in .sol file
------------------------
\apsys_examples\RCLED\DBR_detuned
[title] Effect of DBR detuning in RCLED
[structure] Cylindrical RCLED with implanted current confinement regions.
[comment] To show that if DBR is detuned, resonance will ocurr at an angle
------------------------
\apsys_examples\RCLED\InGaAs
[title] Modeling InGaAs/AlGaAs RCLED
[structure] Cylindrical RCLED with DBR and Ag HR coating
[comment] Optical phase of Ag mirror affects resonant wavelength
------------------------
\apsys_examples\RCLED\long_cavity
[title] InGaN based RCLED at 0.52 um.
[structure] RCLED with long cavity.
[comment] We find many resonant peaks in both wavelenghths and angles
------------------------
\apsys_examples\RCLED\thermal_70well
```

```
[title] Thermal simulation 70 pair MQW RCLED
[structure]  RCLED of InGaAlP emitting at 600 nm
[comment]  Use of current/density dependent lifetime
------------------------
\apsys_examples\RCLED\thermal_rcled
[title] Thermal simulation of an RCLED
[structure] RCLED with multiple quantum wells.
[comment] Demonstration of rolling off RCLED power at higher temperature
------------------------
\apsys_examples\sige_proc_device
[title] Process and device simulation of SiGe MOSFET
[strucutre] MOSFET with SiGe pocket
[purpose] To provide a template for strain treatment of SiGe MOSFET
------------------------
\apsys_examples\solar_cell\compound\3D_beam
[title] 3D models of triple junction solar cell powered by focused beam
[structure] 3D triple junction solar cell with tunnel junction.
[comment] Use of loop structure to set up 3D planes.
------------------------
\apsys_examples\solar_cell\compound\3D_beam_thermal
[title] 3D models of triple junction solar cell powered by focused beam
[structure] 3D triple junction solar cell with tunnel junction.
[comment] Use of loop structure to set up 3D planes.
------------------------
\apsys_examples\solar_cell\compound\3D_beam_zdir
[title] 3D focused beam model with beam from z-direction
[structure] 3D triple junction solar cell with tunnel junction.
[comment] Set up cell with junction parallel to x-y plane.
------------------------
\apsys_examples\solar_cell\compound\CdSe_CZT_CIGS_cell
[title] CsSe-CIGS cell simulation
[material] CdSe - CuInGaSe layers
[remark] provide serveral templates for CIGS based cells.
------------------------
\apsys_examples\solar_cell\compound\diffraction
[title] triple junction solar cell simulation with diffraction model.
[structure] 2D triple junction solar cell using diffraction model.
[comment] Diffraction model takes into account different wavelengths.
------------------------
\apsys_examples\solar_cell\compound\InGaN_cell
[title] InGaN solar cell
[material] InGaN
[purpose] To provide a working template of InGaN cell
```

```
------------------------
\apsys_examples\solar_cell\compound\InGaN_Si
[title] InGaN/Si solar cell
[structure] InGaN/Si heterojunction solar cell
[comment] Use of tunneling junction in a special structure
------------------------
\apsys_examples\solar_cell\compound\inverted
[title] GaInP/GaAs/GaInAs solar cell simulation.
[structure] 2D triple junction solar cell with tunnel junction.
[comment] High efficiency (about 34%) solar cell
------------------------
\apsys_examples\solar_cell\compound\invtTJSCEQE
[title] GaInP/GaAs/GaInAs solar cell simulation for EQE computation.
[structure] 2D triple junction solar cell with tunnel junction.
[comment] Using series Project to compute EQE
------------------------
\apsys_examples\solar_cell\compound\multi-light-source
[title] triple junction solar cell simulation, multiple light source.
[structure] 2D triple junction solar cell with tunnel junction.
[comment] Demo of multiple light source
------------------------
\apsys_examples\solar_cell\compound\tj_thermal
[title] triple junction solar cell thermal simulation.
[structure] 2D triple junction solar cell with tunnel junction.
[comment] Setting up thermal boundary conditions.
------------------------
\apsys_examples\solar_cell\compound\triple_junc
[title] triple junction solar cell simulation.
[structure] 2D triple junction solar cell with tunnel junction.
[comment] Use of equivalent mobility for tunnel junction.
------------------------
\apsys_examples\solar_cell\Si_simple
[title] Silicon solar cell simulation.
[structure] 1D silicon p-n junction.
[comment] Use of AM15 solar spectrum to achieve accuracy.
------------------------
\apsys_examples\solar_cell\silicon\EQE_wavelength
[title] Si solar cell for wavelength dependent simulation
        using internal wavelength scan.

------------------------
\apsys_examples\solar_cell\silicon\import_doping
[title] Silicon solar cell simulation with imported doping profile
```

```
[structure] 1D silicon p-n junction.
[comment] Use of AM15 solar spectrum to achieve accuracy.
------------------------
\apsys_examples\solar_cell\silicon\IQE_wavelength
[title] Computing IQE versus wavelength for solar cell
[structure] silicon P(+)P(-)N structure
[comment] Using series to scan wavelength
------------------------
\apsys_examples\solar_cell\silicon\LFC_RCC_solar_cell
[title] Modeling of RCC solar cell with laser fired contact
[structure] Structure by CSuprem, & combine CSuprem & APSYS
[comment] RUN_all.bat will run all the 4 steps.
------------------------
\apsys_examples\solar_cell\silicon\LFC_RCC_solar_cell\step01
[title] Modeling of RCC solar cell with laser fired contact
[structure] Structure by CSuprem, & combine CSuprem & APSYS
[comment] RUN_all.bat will run all the 4 steps.
------------------------
\apsys_examples\solar_cell\silicon\LFC_RCC_solar_cell\step02
[title] Modeling of RCC solar cell with laser fired contact
[structure] Structure by CSuprem, & combine CSuprem & APSYS
[comment] RUN_all.bat will run all the 4 steps.
------------------------
\apsys_examples\solar_cell\silicon\LFC_RCC_solar_cell\step03
[title] Modeling of RCC solar cell with laser fired contact
[structure] Structure by CSuprem, & combine CSuprem & APSYS
[comment] RUN_all.bat will run all the 4 steps.
------------------------
\apsys_examples\solar_cell\silicon\LFC_RCC_solar_cell\step04
[title] Modeling of RCC solar cell with laser fired contact
[structure] Structure by CSuprem, & combine CSuprem & APSYS
[comment] RUN_all.bat will run all the 4 steps.
------------------------
\apsys_examples\solar_cell\silicon\metal_tunnel
[title] Silicon solar cell simulation with Schottky & tunneling.
[structure] 1D silicon p-n junction.
[comment] Use of tunneling for Schotky effect.
------------------------
\apsys_examples\solar_cell\silicon\PERT_2coatings
[title] Silicon PERT Cell.
[structure] Passivated emitter, rear totally diffused (PERT).
[comment] PERT Cell with contacts on both sides.
------------------------
```

```
\apsys_examples\solar_cell\silicon\photocurrent_life\surf_vel1
[title] Silicon wafer photo-current lifetime calibration
[structure] 2D Si wafer, lightly n-doped
[comment] May be used to calibrate surface recombination velocity
------------------------
\apsys_examples\solar_cell\silicon\photocurrent_life\surf_vel2
[title] Silicon wafer photo-current lifetime calibration
[structure] 2D Si wafer, lightly n-doped
[comment] May be used to calibrate surface recombination velocity
------------------------
\apsys_examples\solar_cell\silicon\PIN
[title] Silicon solar cell simulation.
[structure] 1D silicon p-n junction.
[comment] Use of AM15 solar spectrum to achieve accuracy.
------------------------
\apsys_examples\solar_cell\silicon\poly
[title] Simple PIN-like solar cell based on Si or poly Si.
[structure] 2D silicon PIN strcuture.
[comment] Use of AM15 solar spectrum.
------------------------
\apsys_examples\solar_cell\silicon\RCC_2coatings
[title] Silicon Rear Contacted cell (RCC) simulation.
[structure] 2D Si RCC structure with double coating layers.
[comment] RCC with double coating layers.
------------------------
\apsys_examples\solar_cell\silicon\RCC_2D_coating
[title] Silicon Rear Contacted cell (RCC) simulation.
[structure] 2D silicon RCC structure.
[comment] RCC.
------------------------
\apsys_examples\solar_cell\silicon\RCC_flat2d_rt3d
[title] Ray-tracing (RT) technique for comparison.
[structure] RCC structure (x-direction size reduced).
[comment] For comparison with triangle textured case.
------------------------
\apsys_examples\solar_cell\silicon\RCC_triangle2d_rt3d
[title] Ray-tracing (RT) technique for comparison.
[structure] RCC structure with triangle interface.
[comment] For comparison with flat case. This example uses
------------------------
\apsys_examples\solar_cell\silicon\RCC_triangle3d_rt3d
[title] Ray-tracing (RT) technique for comparison.
[structure] RCC structure with triangle interface.
```

```
[comment] For comparison with flat case. This example uses
------------------------
\apsys_examples\solar_cell\silicon\temperature_dependent
[title] Si solar cell for temp dependent simulation with series
        Command.

------------------------
\apsys_examples\solar_cell\silicon\wavelength_dependent
[title] Si solar cell for wavelength dependent simulation
        with series Command.

------------------------
\apsys_examples\solar_cell\solar_qd_qw\QD
[title] Simulation of quantum dot solar cell
[structure] InAs/GaAs dot in i region of pin structure
[purpose] To demo how effective miniband model is used in QD device
------------------------
\apsys_examples\solar_cell\thinfilm\3d_tandem_texture
[title] 3D textured thinfilm tandem cell simulation
[structure] a-Si/muc-Si tandem cell with pyramid texture
[comment] Use of mesh plane bending to obtain  3D texture
------------------------
\apsys_examples\solar_cell\thinfilm\alpha
[title] Simple PIN-like solar cell based on alpha-Si, or amorphous Si.
[structure] 2D silicon PIN strcuture.
[comment] Use of AM15 solar spectrum.
------------------------
\apsys_examples\solar_cell\thinfilm\Si_Tandem_cell
[title] Amorphous/micro-crystal silicon tandem cells
[structure]  Ag/ZnO/muC-Si-PIN/a-Si/ITO
[comment] Use of tunneling junction to connect 2 cells back to back
------------------------
\apsys_examples\solar_cell\thinfilm\SiC_TJ_tandem
[title] Amorphous SiC and SiGe triple junction tandem cells
[structure]  muC-Si/a-SiGe/a-SiC
[comment] Use of tunneling junction to connect 3 cells back to back
------------------------
\apsys_examples\solar_cell\thinfilm\SiGe_TJ_tandem
[title] Amorphous SiGe triple junction tandem cells
[structure]  a-SiGe/a-SiGe/a-Si
[comment] Use of tunneling junction to connect 3 cells back to back
------------------------
\apsys_examples\solar_cell\thinfilm\STL_import_solar
```

```
[title] STL import example
[structure] p-i-n silicon solar cell.
[comment] A simple example of STL file import procedure
------------------------
\apsys_examples\solar_cell\thinfilm\tandem_ito_semicon
[title] Amorphous/micro-crystal silicon tandem cells - ITO as semiconductor
[structure]  Ag/ZnO/muC-Si-PIN/a-Si/ITO
[comment] Use of tunneling junction to connect 2 cells back to back
------------------------
\apsys_examples\tunnel_junc_model\physical_tunnel_junc
[title] Tunneling junction based on non-local physical model
[structure] GaAs tunneling junction
[comment] TJ model in forward bias agrees with experiment
------------------------
\apsys_examples\tunnel_junc_model\trap_ass_tunnel
[title] Trap assisted tunnelling model
[structure] GaAs p-n junction
[remark] Demo of TAT in TJ simulation
------------------------
\apsys_examples\tunnel_junc_model\tunnel_junc_IV
[title] Tunneling junction biased on non-local physical model
[structure] GaAs/AlGaAs tunneling junction
[comment] Interband tunneling models are used in both forward and reverse.
------------------------
\apsys_examples\tunnel_junc_model\tunnel_junc_local
[title] Tunneling junction with heterojunction and MQW
[structure] Fused junction InGaAlAs/InP/AlGaAs, Also GaN junction
[comment] Demo. the use of tunneling junction to achieve injection.
------------------------
\apsys_examples\tunnel_junc_model\zener_impact_local
[title] Simulation of a tunneling junction biased in both directions
[structure] GaAs/AlGaAs tunneling junction
[comment] Both impact ionzation and Zener interband tunneling models are used
------------------------
\apsys_examples\typeII_PD
[title] Simulation of type II MQW photo detector
[device] Type II MQW of GaAsSB/InGaAsP grown on InP
[remark] Use of type II QW transport model
------------------------
```

H.2 LASTIP Examples

```
\lastip_examples\active_grade\complex
[title] A layered laser diode simulation with graded complex layers.
[material] GaAs/AlGaAs at 0.82 microns.
[structure] 1D laser, GRIN-SCH-SQW structure.
[purpose] To demo how to grade a complex coupled MQW structure.
------------------------
\lastip_examples\active_grade\grade_function
[title] Arbitrary composition grading in graded complex layers.
[material] GaAs/AlGaAs at 0.82 microns.
[structure] 1D laser, GRIN-SCH-SQW structure.
[purpose] To demo how to using a composition grading function in .mater
------------------------
\lastip_examples\active_grade\single_well
[title] A layered laser diode simulation with graded active layer.
[material] GaAs/AlGaAs at 0.82 microns.
[structure] 1D laser, GRIN-SCH-SQW structure, with graded active layer.
[purpose] To provide an example on how to grade the active/barrier layers.
------------------------
\lastip_examples\broad_areas\asymmetric
[title] Asymmetric broad area laser.
[material] InGaAs/GaAs at 0.98 micron.
[structure] An asymmetric broad area laser.
[comment] How to deal with multiple modes in broad area lasers.
------------------------
\lastip_examples\broad_areas\broad_ingaalp
[title] Simulation of broad area InGaAlP laser.
[material] Strained MQW GaInP/AlGaInP (lambda=0.65-0.71 um)
[structure] Broad area laser.
[purpose] Set up of multiple mode in broad area laser model.
------------------------
\lastip_examples\bulk_InGaAsP
[title] 1D/2D bulk DH laser.
[material] Bulk InGaAsP at 1.3 micron.
[structure] 1D/2D bulk channel-substrate waveguide laser.
[comment] For demo of conventional bulk DH laser.
------------------------
\lastip_examples\complex_MQW\combo
[title] Combination of complex and simple quantum wells.
[material] InGaAsP MQW emisison at 1.5 um.
[structure] MQW with mixed type of barriers. Strain compensated.
[comment] To illustrate the use of complex MQW with simple QWs.
```

```
------------------------
\lastip_examples\complex_MQW\coupled
[title] Coupled wells.
[material] InGaAsP MQW at 1.55 um emission.
[structure] Complex cx-InGaAsP coupled wells.
[comment] Study of interaction between wells.
------------------------
\lastip_examples\complex_MQW\step_qw
[title] Optical gain model for step quantum well
[material] InGaAsP QW on GaAs substrate
[structure] A step is formed near the barrier to make a 5 layer QW structure
[comment] Use of inner_bar_gain to calculate optical gain at inner barrier
------------------------
\lastip_examples\complex_MQW\uneven_bar
[title] Uneven barrier
[material] AlGaAs/AlGaAs QW at 0.83 um
[structure] Complex cx-AlGaAs single well with uneven barrier
[comment] Demo of uneven barrier using complex-MQW feature
------------------------
\lastip_examples\current_blocking\buryhet
[title] Buried het. laser with irregular shape.
[material] Bulk AlGaAs at 0.8 micron.
[structure] 2D bury het. laser.
[comment] Use of "new_doping" method to treat difficult structure.
------------------------
\lastip_examples\current_blocking\leaky_mqw
[title] Leakage structure.
[material]QW  InGaAsP/InP at 1.6 micron.
[structure] 2D MQW laser.
[comment] Demo of new_doping method in 3-column structure.
------------------------
\lastip_examples\current_blocking\new_doping
[title] New_doping technique.
[material] Bulk AlGaAs at 0.8 micron.
[structure] 2D bury het. laser.
[comment] To show "new_doping" method for current blocking structure.
------------------------
\lastip_examples\dual_polar
[title] Dual polarization lasing
[material] InGaAsP wells grown on InP substrate.
[structure] 1D laser, MQW strained well, unstrained barrier.
[purpose] To demo set up of dual polarization simulation.
------------------------
```

```
\lastip_examples\gainguide_3e
[title] Three electrode laser.
[material] GaAs/AlGaAs at 0.82 micron.
[structure]3-electrode stripe geometry, gain guided laser.
[comment] Demo of how gain guiding can control the optical beam and modes.
------------------------
\lastip_examples\GaN_growth_plane\miller_index\mplane_wgpara
[title] InGaN/GaN QW mplane laser diode simulation using full k.p theory
[material]  InGaN/GaN
[device]  InGaN/GaN LD
[comment]  Use of k.p gain data import to speed up 2/3 D simulation.
------------------------
\lastip_examples\GaN_growth_plane\miller_index\mplane_wgperp
[title] InGaN/GaN QW mplane laser diode simulation using full k.p theory
[material]  InGaN/GaN
[device]  InGaN/GaN LD
[comment]  Use of k.p gain data import to speed up 2/3 D simulation.
------------------------
\lastip_examples\GaN_growth_plane\miller_index\semipolar
[title] InGaN/GaN QW mplane laser diode simulation using full k.p theory
[material]  InGaN/GaN
[device]  InGaN/GaN LD
[comment]  Use of k.p gain data import to speed up 2/3 D simulation.
------------------------
\lastip_examples\GaN_growth_plane\plane_angle\bulk_effective_mass
[title] m-plane InGaN/GaN MQW LD simulation
[material] InGaN/GaN at 0.41 um
[structure] Simplified 1D MQW structure
[comment] To illustrate effect of wurtzite crystal orientation
------------------------
\lastip_examples\GaN_growth_plane\plane_angle\bulk_effective_mass\phi0
[title] m-plane InGaN/GaN MQW LD simulation
[material] InGaN/GaN at 0.41 um
[structure] Simplified 1D MQW structure
[comment] To illustrate effect of wurtzite crystal orientation
------------------------
\lastip_examples\GaN_growth_plane\plane_angle\bulk_effective_mass\phi90
[title] m-plane InGaN/GaN MQW LD simulation
[material] InGaN/GaN at 0.41 um
[structure] Simplified 1D MQW structure
[comment] To illustrate effect of wurtzite crystal orientation
------------------------
\lastip_examples\GaN_growth_plane\plane_angle\bulk_effective_mass\ref
```

```
[title] m-plane InGaN/GaN MQW LD simulation
[material] InGaN/GaN at 0.41 um
[structure] Simplified 1D MQW structure
[comment] To illustrate effect of wurtzite crystal orientation
------------------------
\lastip_examples\GaN_growth_plane\plane_angle\mplane_LD
[title] InGaN/GaN QW mplane laser diode simulation using full k.p theory
[material]  InGaN/GaN
[device]  InGaN/GaN LD
[comment]  Use of k.p gain data import to speed up 2/3 D simulation.
------------------------
\lastip_examples\GaN_growth_plane\plane_angle\mplane_park
[title] InGaN/GaN QW mplane optical gain model
[structure]  InGaN/GaN MQW.
[comment]  Study of optical gain of arbitrary crystal orientation.

------------------------
\lastip_examples\manybody\GaAs
[title] Manybody gain enhancement effect.
[material] AlGaAs QW at 0.83 um.
[structure] SQW.
[comment] Manybody effect is done under the Pade approximation.
------------------------
\lastip_examples\manybody\InGaN
[title] InGaN/GaN MQW laser simulation with manybody effect
[material] Simplified InGaN/GaN/AlGaN MQW laser.
[comment] Comparison of manybody gain/PL/index effect on GaN based lasers.

------------------------
\lastip_examples\miniband\basic_nin
[title] Use of mini-band transport model for superlattice in GaN-based lasers
[material] InGaN/AlGaN
[structure] 2D LD model
[purpose] To demonstrate setting up mini-band transport model
------------------------
\lastip_examples\miniband\LD
[title] Use of mini-band transport model for superlattice in GaN-based lasers
[material] InGaN/AlGaN
[structure] 2D LD model
[purpose] To demonstrate setting up mini-band transport model
------------------------
\lastip_examples\modulation
[title]Small signal frequency modulation
```

```
[material] InGaAs/AlGaAs at 0.95 microns.
[structure] 2D laser, GRIN-SCH structure.
[purpose] To compare the LI and modulation of SQW and MQW.
------------------------
\lastip_examples\mqw_ridge\GaAs
[title] A classic GRINCH-SCH-SQW laser.
[material] GaAs/AlGaAs at 0.82 microns.
[structure] 1D/2D ridge waveguide laser, GRIN-SCH-SQW structure.
[purpose] To demo a simple yet representative structure.
------------------------
\lastip_examples\mqw_ridge\ingaalas_inp
[title] InGaAlAl laser.
[material] QW InGaAlAs/InP at 1.3 micron.
[structure] 1D GRIN-SCH-SQW laser.
[comment] An alternative to InGaAsP/InP system.
------------------------
\lastip_examples\mqw_ridge\InGaAs
[title] InGaAs/GaAs: An early strained QW laser.
[material] InGaAs/AlGaAs at 0.98 microns.
[structure] 1D/2D ridge waveguide laser, GRIN-SCH-SQW structure.
[comment] To show one of the early strained QW laser.
------------------------
\lastip_examples\mqw_ridge\InGaAsN
[title] InGaAsN/GaAs laser.
[material] InGaAsN/GaAs at 1.3 um.
[structure] Ridge waveguide.
[purpose] To provide a convient structure for a new material system.
------------------------
\lastip_examples\mqw_ridge\qw_InGaAsP
[title] InGaAsP laser for 1.55 application.
[structure] 1D/2D MQW ridge waveguide laser.
[material] MQW strained InGaAsP/InP.
[comment] To provide a good structure of MQW laser at 1.55 emission.
------------------------
\lastip_examples\mqw_ridge\setup_MQW
[title] Use of SetupLayer for an InGaAsP MQW laser
[material] InGaAsP/InGaAsP/InP at 1.55 microns.
[structure] 1D MQW laser, may be converted to 2D ridge waveguide laser.
[purpose] To provide a tutorial example on how to use SetupLayer program.
------------------------
\lastip_examples\multi_active
[title] Super lattice with variable QW with.
[material] GaAs/AlGaAs.
```

```
[structure] 1D super lattice with different well widths.
[purpose] To show different type of active region in one simulation.
------------------------
\lastip_examples\multicavity\bipolar_cascade
[title] Two color bipolar cascade laser.
[material] InGaAs/GaAs MQW
[structure] Two cavities connected by tunnel junction.
[purpose] To demonstrate the simulation of multi-cavity laser with tunnel junctions.
------------------------
\lastip_examples\multicavity\side_by_side
[title] Simulation of a multiple cavity laser structure
[material] GaAs/AlGaAs at 0.81 and 0.84 microns.
[structure] 3-column, 2 SQW structures.
[purpose] To show how to set up a 2-cavity laser.
------------------------
\lastip_examples\multimode\bulk2d
[title] Multimode bulk 2D laser simulation.
[material] bulk InGaAsP/InP at 1.3 micron.
[structure] 2D bulk channel-substrate waveguide laser.
[purpose] Demo of multimode solver based on examples/bulk_InGaAsP
------------------------
\lastip_examples\multimode\ingaasp_gaas
[title] Multimode QW InGaAsP/GaAs laser.
[material] QW InGaAsP/GaAs system at 980nm.
[structure] 2D ridge waveguide LD.
[purpose] To provide a convenient example of multimode laser 980nm application.
------------------------
\lastip_examples\multimode\kink_effect
[title] Lateral model dependent mirror reflectivity and kink effect
[material] AlGaAs QW at 0.83 um.
[structure] Ridge waveguide.
[purpose] To set variable mirror reflectivity and study kink effect
------------------------
\lastip_examples\multimode\msas
[title] MSAS laser with multimode.
[material] AlGaAs at 0.78 um.
[structure] MSAS structure.
[purpose] To demo the power of Arnoldi solver in selecting lateral modes.
------------------------
\lastip_examples\other\chuang_gain
[title] Gain and subbands of InGaN/AlGaN QW.
[material] InGaN/AlGaN
[structure] InGaN/AlGaN QW, 1D.
```

```
[comment] To compare with Chuang's published data.
------------------------
\lastip_examples\other\ex_import_gain
[title] Export and import of gain data to and from ASCII file
[material] GaAs/AlGaAs at 0.82 microns.
[structure] 1D laser, GRIN-SCH-SQW structure.
[purpose] To provide an example on how to export/import gain/index/PL data
------------------------
\lastip_examples\other\fancy_geo
[title] Flexible device geometry
[material] bulk AlGaAs at 0.8 micron.
[structure] a fancy geometry ridge waveguide laser.
[purpose] demo of fancy geometry and treatment of curved boundary.
------------------------
\lastip_examples\other\farfield
[title] Far field computation.
[structure] non-planar ridge waveguide laser.
[material] bulk AlGaAs at 0.8 micron.
[purpose] Demo of far field computation.
------------------------
\lastip_examples\other\gain_index\index_change
[title] Gain and index change calculations.
[material] GaAs/AlGaAs, bulk AlGaAs.
[structure] Bulk and quantum well laser.  1D.
[comments] study of alpha factor and index change.
------------------------
\lastip_examples\other\gain_index\landsberg
[title] Gain broadening of Landsberg type.
[structure] Bulk and QW laser.
[material] Bulk and QW  InGaAsP/InP.
[purpose] Demo of Landsberg gain broadening.
------------------------
\lastip_examples\other\mmb_gain
[title] Use of microscopic many-body gain table
[material] InGaAs/AlGaAs at 1. micron.
[structure] 1D laser, GRIN-SCH-SQW structure.
[purpose] To provide a tutorial example on how to import the mmb gain table.
------------------------
\lastip_examples\other\mmb_gain_export
[title] Export of microscopic many-body gain table (Koch gain table)
[material] InGaAs/AlGaAs at 1. micron.
[structure] 1D laser, GRIN-SCH-SQW structure.
[purpose] To convert into general purpose gain table good for thermal model.
```

```
------------------------
\lastip_examples\other\parameterize
[title] Parameterization of quantum well optical models
[material] GaAs/AlGaAs at 0.82 microns.
[structure] 1D laser, GRIN-SCH-SQW structure.
[purpose] Speed up of simulation using parameterized optical gain/spon/index.
------------------------
\lastip_examples\other\pump_spectrum
[title] Opitcally pumped laser using continuous spectrum.
[material] GaAs/AlGaAs emitting at 0.82 microns.
[structure] 1D laser, GRIN-SCH-SQW structure.
[comment] To demonstrate optical pumping using a spectrum.
------------------------
\lastip_examples\other\pumped_laser
[title] Opitcal pumping.
[material] GaAs/AlGaAs at 0.82 microns.
[structure] 1D laser, GRIN-SCH-SQW structure.
[comment] To demonstrate optical pumping in simulation.
------------------------
\lastip_examples\other\quick_LD
[title] A quick laser diode simulation.
[material] GaAs/AlGaAs at 0.82 microns.
[structure] 1D laser, GRIN-SCH-SQW structure.
[purpose] To demonstrate how to use the gain preview for a quick laser model
------------------------
\lastip_examples\other\trap
[title] Trap models for semi-insulating layers.
[material] QW  InGaAsP/InP at 1.55 micron.
[structure] 2D MQW bury het. laser with semi-insulating blocking layer.
[comment] Deep level trap model with new_doping technique.
------------------------
\lastip_examples\other\vectorial
[title] Vectorial wave model.
[structure] Bulk buryhet laser.
[material] Bulk AlGaAs at 0.78 um.
[comments] Difference between scalar and vectorial wave model.
------------------------
\lastip_examples\pml_eeim\eeim_basic
[title] EEIM lateral modes.
[material] GaAs/AlGaAs QW.
[structure] Ridge waveguide.
[comment] To show the basic set up of EEIM simulation.
------------------------
```

```
\lastip_examples\pml_eeim\msas_pml
[title] Use of PML in MSAS Structure
[material] Bulk DH structure with buried GaAs loss layer.
[structure] MSAS structure
[comment] To show the use of PML boundary on MSAS.
------------------------
\lastip_examples\pml_eeim\scowl\PML
[title] Slab coupled optical waveguide laser (SCOWL).
[material] InGaAs/AlGaAs at 0.95 microns.
[structure] 2D laser, GRIN-SCH-MQW structure.
[comment] To demonstrate the use of PML in SCOWL
------------------------
\lastip_examples\pml_eeim\scowl\without_PML
[title] Slab coupled optical waveguide laser (SCOWL).
[material] InGaAs/AlGaAs at 0.95 microns.
[structure] 2D laser, GRIN-SCH-MQW structure.
[comment] To be used as a reference case without PML
------------------------
\lastip_examples\pml_eeim\substrate_loss\gaas26a
[title] Modeling substrate loss using PML.
[material] AlGaAs QW at 0.83 um.
[structure] Ridge waveguide laser with lossy substrate closs to active region.
[comment] To illustrate the use of PML to study substrate radiative loss.
------------------------
\lastip_examples\pml_eeim\substrate_loss\gaas26b
[title] Modeling substrate loss using PML.
[material] AlGaAs QW at 0.83 um.
[structure] Ridge waveguide laser with lossy substrate closs to active region.
[comment] To be compared with case of gaas26a
------------------------
\lastip_examples\quantum_cascade\main_project\qcl_1d
[title] 1D macroscopic model of a quantum cascade laser
[material] InGaAlAs/InP
[structure] MQW QCL
[comment] To demonstrate the set up of macroscopic MQW QCL simulation
------------------------
\lastip_examples\quantum_cascade\main_project\qcl_2d
[title] 2D macroscopic model of a quantum cascade laser
[material] InGaAlAs/InP
[structure] MQW QCL
[comment] To demonstrate the set up of macroscopic MQW QCL simulation
------------------------
\lastip_examples\quantum_cascade\micro_gain\4wells
```

```
[title] Subband structure design for  a four wells QCL
[material] InGaAs/InAlAs strain balanced MQW.
[structure] 1D structure with periodic MQW
[purpose] To provide a tutorial example for setting up basic QC Laser
------------------------
\lastip_examples\quantum_cascade\micro_gain\GaAs
[title] Subband structure design for GaAs  QCL
[material] GaAs/AlGaAs MQW QCL at 9um.
[structure] 1D structure with periodic MQW
[purpose] To provide a tutorial example for setting up basic QC Laser
------------------------
\lastip_examples\quantum_cascade\micro_gain\InGaAs
[title] Subband structure design for quantum cascade laser
[material] InGaAs/InAlAs MQW.
[structure] 1D structure with periodic MQW
[purpose] To provide a tutorial example for setting up basic QC Laser
------------------------
\lastip_examples\quantum_dots\box_half
[title] Quantum states of half a quantum dot
[material] InGaAs/GaAs
[structure] Half of InGaAs/GaAs quantum dot of box shape
[purpose] To provide a setup example for a half structure of a quantum dot
------------------------
\lastip_examples\quantum_dots\box_single
[title] Quantum states of a simple quantum dot
[material] InGaAs/GaAs
[structure] InGaAs/GaAs quantum dot of box shape
[purpose] To provide a setup of a 3D quantum dot structure
------------------------
\lastip_examples\quantum_dots\columnar
[title] Simulation of a self-assembled columnar structure QDOT laser
[material] InGaAs/GaAs/AlGaAs
[structure] Stacked interacting columnar quantum dots
[purpose] To setup example for coupled structure in cylindrical system
------------------------
\lastip_examples\quantum_dots\columnar\cone2
[title] Quantum states of stacked columnar quantum dots
[material] InGaAs/GaAs
[structure] Stacked interacting columnar quantum dots
[purpose] To setup example for coupled structure in cylindrical system
------------------------
\lastip_examples\quantum_dots\columnar\cone400
[title] Quantum states of stacked columnar quantum dots
```

```
[material] InGaAs/GaAs
[structure] Stacked interacting columnar quantum dots
[purpose] To setup example for coupled structure in cylindrical system
------------------------
\lastip_examples\self-consistent\AlGaAs
[title] A Simple Self-consistent MQW model
[material] GaAs/AlGaAs
[structure] SQW 1D laser structure.
[purpose] Setting a simple self-consistent 1D model.
------------------------
\lastip_examples\self-consistent\InGaAlAs
[title] Self-consistent MQW model
[material] InGaAlAs/InP.
[structure] MQW 1D laser structure.
[purpose] Explain the basic steps of turning on self-consistent model
------------------------
\lastip_examples\self-consistent\InGaN
[title] Piezo-electric effect and self-consistent model.
[material] GaN-base wurtzite emitting at blue
[structure] MQW 1D laser structure.
[purpose] Demo the use of "interface" to define the piezo surface charge.
------------------------
\lastip_examples\self-consistent\nakamura
[title] Room Temperature-CW Operated InGaN MQW Laser Diode
[material] InGaN/GaN/AlGaN.
[comment] Reproduced experiment results of Shuji Nakamura (1997).

------------------------
\lastip_examples\self-consistent\UV_algan
[title] Piezo-electric effects in a UV-InGaN/AlGaN laser
[material] GaN-based wurtzite
[structure] MQW 1D laser structure.
[purpose] Illustrate the use of "interface" to define the piezo surface charge.
------------------------
\lastip_examples\short_wavelength\GaN\shuji
[title] A Simplified MQW GaN blue LED/LD
[material] InGaN/InGaN MQW.
[structure] N-contact placed on the side.
[comment] Basic set up with no piezo effects included.
------------------------
\lastip_examples\short_wavelength\ZnSe
[title] Short wavelength ZnSe laser.
[material] QW ZnSe-based laser emitting at blue.
```

```
[structure] 1D and 2D QW laser.
[comments] Demo of multiple tranverse mode model.
------------------------
\lastip_examples\tau_model
[title] Intraband relaxation time.
[material] AlGaAs QW.
[structure] 1D QW laser.
[purpose] Illustrate the model of scattering tau parameter.
------------------------
\lastip_examples\temperature\GaAs_temperature
[titel] Temperature performance of AlGaAs QW laser.
[material] GaAs/AlGaAs.
[structure] 1D ridge waveguide laser, GRIN-SCH-SQW structure.
[purpose] To demo temperature capability.
------------------------
\lastip_examples\temperature\InP_T0
[title] T0 of InGaAsP long wavelength laser.
[material] MQW InGaAsP at 1.5 micron.
[structure] 1D MQW structure.
[purpose] Show how to calculate T0.
------------------------
\lastip_examples\thermal\bulk1d
[title]Thermal effects in bulk InGaAsP laser.
[material]Bulk InGaAsP at 1.3 um.
[structure]Bulk 1D DH InGaAsP laser.
[comment] Demo of simple simulation of self-heating effect.
------------------------
\lastip_examples\thermal\external_cir
[title] External thermal circuit.
[material] AlGaAs QW at 0.83 um.
[structure] 1D AlGaAs QW at 0.83 um with external thermal ciruit.
[comment] Use of external thermal circuit as special boundary.
------------------------
\lastip_examples\thermal\hot_algaas
[title] Thermal effects in AlGaAs QW laser.
[material] AlGaAs QW at 0.83 um.
[structure] 1D and 2D ridge waveguide AlGaAs QW at 0.83 um.
[comment] Demo of basic set up of a thermal simulation.
------------------------
\lastip_examples\thermal\import_gain
[title] Use of imported optical gain for thermal simulation
[material] GaAs/AlGaAs laser at 0.83 micron.
[structure] 1D laser, GRIN-SCH-SQW structure.
```

```
[purpose] To demo imported gain data for thermal simulation.
------------------------
\lastip_examples\thermal\iso_vs_self
[title] Isothermal versus self-heating simulation.
[material] MQW InGaAsP at 1.5 micron.
[structure] 1D MQW structure.
[purpose] Demo uniform self-heating is equivalent to isothermal simulation.
------------------------
\lastip_examples\thermal\thermal_bound
[title] Thermal boundaries.
[material] AlGaAs QW.
[structure] Ridge waveguide laser with SiO2 cover cooled to 77K.
[purpose] To demo the set up of pure thermal boundary.
------------------------
\lastip_examples\thermal\thermal_trans
[title]Thermal transient simulation for bulk InGaAsP laser.
[material]Bulk InGaAsP at 1.3 um.
[structure]Bulk 1D DH InGaAsP laser.
[comment] Demo of thermal transient simulation.
------------------------
\lastip_examples\transient\eye_diagram
[title] Calculation of eye diagram
[material] SQW GaAs/AlGaAs
[comments] Generation of eye diagram from transient simulation data.

------------------------
\lastip_examples\transient\fourier
[title] Modulation response using Fourier transform technique
[material] SQW GaAs/AlGaAs
[structure] 1D GRIN-SCH-SQW laser.

------------------------
\lastip_examples\transient\trans_bulk
[title] Transient simulation bulk 1D DH laser.
[material] Bulk InGaAsP/InP.
[structure] 1D bulk het. laser.
[comment] Demo of large signal modulation simulation.
------------------------
\lastip_examples\transient\trans_GaAs
[title] Transient and power spectrum RWD laser.
[material] SQW GaAs/AlGaAs
[comments] Large and small signal analysis of GRIN-SCH-SQW laser modulation.
```

```
------------------------
\lastip_examples\tunneling\ingaalp_650nm
[titel] Short wavelength InGaAlP laser
[material] MQW InGaAlP/GaAs emitting at 650 nm.
[structure] 2D non-planar ridge structure.
[comments] Use of tunneling and grading junction to reduce resistance.
------------------------
\lastip_examples\tunneling\schottky_diode
[title] Study of tunneling effects.
[structure] 1D Schottky diode.
[material] Bulk GaAs.
[comments] Demo of tunneling effect in Schottky contact.
------------------------
\lastip_examples\type2_MQW\GaAsSb
[title] A simple type-II electrically pumped laser of Sb based material
[material] cx-GaAsSb to form type-II QW complex
[structure] 1D laser
[purpose] To provide a simple demonstration of setting up a type-II QW laser.
------------------------
\lastip_examples\type2_MQW\type2_setup
[title] A simple type-II electrically pumped laser
[material] cx-AlGaAs with artificial band offset to form type-II QW complex
[structure] 1D laser
[purpose] To provide a simple demonstration of setting up a type-II QW laser.
------------------------
\lastip_examples\valence_mixing\8x8chuang
[title] Comparison of different k.p theory for zincblende quantum well
[material] InGaAsP/InGaAsP at 1.55 microns.
[structure] MQW structure laser.
[purpose] Comparing results and speed of 4x4, 6x6 and 8x8 k.p theory
------------------------
\lastip_examples\valence_mixing\8x8compare
[title] Comparison of different k.p theory for zincblende quantum well
[material] GaAs/AlGaAs at 0.82 microns.
[structure] 1D laser, GRIN-SCH-SQW structure.
[purpose] Comparing results and speed of 4x4, 6x6 and 8x8 k.p theory
------------------------
\lastip_examples\valence_mixing\compare_chuang
[title] Comparing k.p model with Chuang's data.
[material] Quantum well of InGaAsP grown on InP.
[structure] Single quantum well.
[comment] Study of k.p model accuracy as compared with published data
------------------------
```

```
\lastip_examples\valence_mixing\gaas_mixing
[title] k.p model in AlGaAs devices.
[material] Quantum well of AlGaAs/AlGaAs at 0.83 um.
[structure] Single quantum well.
[comment] Study of valence mixing effect in AlGaAs system.
------------------------
\lastip_examples\valence_mixing\inp15_mixing
[title] Valence mixing effect in strained InGaAsP.
[material] Strained quantum well of InGaAsP/InP at 1.55 um.
[structure] Strained single quantum well.
[comment] Study of valence mixing effect in InGaAsP/InP system.
------------------------
```

H.3 PICS3D Examples

```
pics3d_examples\A_tutorial
[title] A tutorial example of PICS3D.
[structure] Bulk DH laser, InGaAsP, DFB grating.
[purpose] To illustrate the set up of a PICS3D simulation.
------------------------
pics3d_examples\AC_analysis\edge
[title] AC analysis for edge laser for PICS3D.
[structure] Bulk DH laser, InGaAsP, DFB grating.
[purpose] To illustrate the set up of a an AC analysis.
------------------------
pics3d_examples\AC_analysis\vcsel
[title] AC analysis for a simple VCSEL with finely tuned MQW location.
[structure] Simple VCSEL with MQW.
[comment] AC analysis method is similar to LASTIP
------------------------
pics3d_examples\amplifier_3d\amp2mode
[title] Semiconductor Optical Amplifier (SOA) with multi-lateral mode
[structure]  DH InGaAsP/InP 1.3 micron, ridge waveguide
[comment] 2 lateral modes are included in simulation
------------------------
pics3d_examples\amplifier_3d\BPM_multimode
[title] BPM-SOA 3D model with multimode initialization
[structure]  Tapered ridge waveguide quantum well SOA
[comment] Use of multiple lateral mode initial BPM solution
------------------------
pics3d_examples\amplifier_3d\inp13amp
[title] 3D model of semiconductor optical amplifier (SOA).
[structure]  DH InGaAsP/InP 1.3 micron.
[comment] Same as the tutorial exmple except DFB gratings removed.
------------------------
pics3d_examples\amplifier_3d\transient
[title] 3D traveling wave optical amplifier (SOA).
[structure]  DH InGaAsP/InP 1.3 micron.
[comment] Same as the tutorial exmple except DFB gratings removed.
------------------------
pics3d_examples\blue_LD
[title] A Simplified MQW GaN blue LED/LD
[material] InGaN/InGaN MQW.
[structure] N-contact placed on the side.
------------------------
pics3d_examples\BPM\mask_taper
```

```
[title] Set up of 3D simulation of tapered structure laser using mask.
[structure] AlGaAs SQW laser with ridge width tapered.
[comment] Use of gemerate_mask program to set up structure
------------------------
pics3d_examples\BPM\symmetric_taper
[title] 3D simulation of tapered structure laser using BPM.
[structure] AlGaAs SQW laser with ridge width tapered.
[comment] Use of BPM in two segments with different mesh structures.
------------------------
pics3d_examples\BPM\taper
[title] 3D simulation of tapered structure laser using BPM.
[structure] AlGaAs SQW laser with ridge width tapered.
[comment] Use of BPM in two segments with different mesh structures.
------------------------
pics3d_examples\broad_area
[title] Broad area high power laser simulation by PICS3D.
[structure] FP laser with 0.5um AlGaAs bulk active layer.
[purpose] To simulate broad area laser PICS3D simulation.
------------------------
pics3d_examples\csuprem_wavegd_stack
[title] BPM simulation of stacked taper waveguide
[structure] silicon waveguides with tapers
[comment] Use of taper and taper_range for complex taper structures
------------------------
pics3d_examples\DBR\3section_tunable
[title] Multi-Section MQW Tunable DBR Laser Simulation
[structure] MQW, passive phase tuning, passive DBR tuning sections.
[purpose] To illustrate the set up of a multi-sectional DBR laser.
------------------------
pics3d_examples\DBR\fiber_grating
[title] Simulation of fiber-grating DBR laser
[structure] MQW InGaAsP/InP emitting at 1.49 um with fiber-grating section.
[comment] Use of fiber/external-cavity option.
------------------------
pics3d_examples\DBR\sampled_grating
[title] Modeling sampled grating DBR laser.
[structure] InGaAsP MQW laser with sampled DBR grating.
[comment] Use of "ref_pitch" and "pitch" to define complex gratings.
------------------------
pics3d_examples\DFB_trans
[title] 3D Transient simulation.
[structure] 1D bulk DH laser emitting at 1.3 um.
[purpose] To illustrate the set up of transient simulation.
```

```
------------------------
pics3d_examples\EAM_DFB_trans
[title] Two-segment EAM-DFB simulation
[structure] MQW InGaAsP EAM-DFB laser emitting at 1.55 um with multi-section.
[comment] Use of self-consistent MQW in EAM modeling
------------------------
pics3d_examples\edge_LD_transient
[title] Transient simulation for bulk 1D DH laser.
[structure] 1D bulk heterojunction laser.
[comment] Demo of large signal modulation simulation.
------------------------
pics3d_examples\facet_effect
[title] Simulation of Laser Facet Effect Using 3D-Flow Model
[structure] Multiple segment MQW laser.
[comment] Full 3D laser simulation with material variation in z-direction.
------------------------
pics3d_examples\grating_2nd
[title] Second order grating DFB laser.
[structure] AlGaAs bulk DH laser at 0.86 um, 2nd order DFB gratings.
[comment] Set up of 2nd order grating; Surface emitting modes.
------------------------
pics3d_examples\hybrid_laser
[title] Hybrid silicon laser
[structure] InP based 1.3 micron laser grown on top of silicon waveguide
[comment] Structure built on Intel-UCSB announcement with taper added
------------------------
pics3d_examples\mplane_InGaN
[title] 3D simulation of InGaN/GaN LD with PICS3D
[device] MQW InGaN/GaN laser
[comment] Generation and import of m-plane gain data in PICS3D
------------------------
pics3d_examples\multicavity
[title] Simulation of a multiple cavity laser structure
[material] GaAs/AlGaAs at 0.81 and 0.84 microns.
[structure] 3-column, 2 SQW structures.
------------------------
pics3d_examples\pa_waveguide\EAM
[title] 3D Electro-Absorption modulator simulation.
[structure] SQW EAM of AlGaAs.
[comment] Use of "self_consistent" and "photo-absorbing waveguide" options.
------------------------
pics3d_examples\pa_waveguide\franz_keldysh
[title] 3D bulk electro-absorption modulator simulation.
```

```
[structure] Bulk InGaAsP material operating at 1.32 um as EAM.
[comment] Use of Franz-Keldysh model with "photo-absorbing waveguide" option.
------------------------
pics3d_examples\pa_waveguide\waveguide_pd
[title] 3D-Waveguide photodetector
[structure] Bulk DH InGaAsP for 1.3-micron detection.
[comment] Use of C-RTG method for waveguide PD modeling
------------------------
pics3d_examples\pump_laser
[title] 3D Pumped Edge Emitting Laser.
[structure] SQW of GaAs/AlGaAs.
[comment] Use of "light_power" to provide a shorter pump wavelength.
------------------------
pics3d_examples\ring_laser
[title] Simulation of a ring laser
[material] InGaAsP MQW for 1.55 emission
[structure] A single ring waveguide
------------------------
pics3d_examples\SLD\2peaks
[title] 3D model of super-luminescent diodes (SLD) emitting with 2 peaks
[structure] SQW SLD of GaAs/AlGaAs with two different SQW
[comment] Use of multiple wavelength to cover wide emissioin range
------------------------
pics3d_examples\SLD\ase_signal_linewidth
[title] 3D semiconductor optical amplifier modeling with input linewidth
[structure]  DH InGaAsP/InP 1.3 micron.
[comment] Demo of signal+ASE spectrum plotting
------------------------
pics3d_examples\SLD\ase_signal_spec
[title] 3D semiconductor optical amplifier model with input spectrum
[structure]  DH InGaAsP/InP 1.3 micron.
[comment] SOA with continuous input power spectrum.
------------------------
pics3d_examples\SLD\SLD_1mode
[title] 3D model of super-luminescent diodes (SLD)
[structure] Bulk DH InGaAsP/InP 1.3 micron
[comment] Based of 3D SOA with zero input power
------------------------
pics3d_examples\SLD\SLD_2mode
[title] Super-luminescent diodes (SLD) model with multi-lateral modes
[structure] Bulk DH InGaAsP/InP 1.3 micron, ridge waveguide
[comment] 2 lateral modes are included in simulation
------------------------
```

```
pics3d_examples\spiral_laser
[title] Simulation of a spiral laser
[structure] MQW InGaAsP/InP emitting at 1.49 with ring + straight waveguide
[comment] Use of mixed cylindrical and rectangle coordinates.
------------------------
pics3d_examples\thermal
[title] Thermal simulation of SQW GaAs/AlGaAs laser.
[structure] FP SQW GaAs/AlGaAs laser.
[comment] Please note the mode hopping effects.
------------------------
pics3d_examples\VCSELS\2mqw
[title] A VCSEL with two MQW regions
[structure] VCSEL with two MQW region joined by tunneling junction
[comment] Higher voltage is required to turn on current
------------------------
pics3d_examples\VCSELS\DBR_index_profile
[title] A simple VCSEL with imported DBR index profile
[structure] Simple VCSEL with MQW.
[comment] Index profile may contain linearly graded index profile.
------------------------
pics3d_examples\VCSELS\extern_cavity
[title] A VCSEL with external air-gap cavity
[structure] MQW VCSEL with air-gap mirror
[comment] Threshold depends on both DBR and external mirror
------------------------
pics3d_examples\VCSELS\GaN_VCSEL
[title] GaN-VCSEL template
[structure] Electrical pumped VCSEL at 0.41 um
[comment] Property senstive to choice of macro and polarization charge
------------------------
pics3d_examples\VCSELS\jim_vcsel
[title] A simple VCSEL with finely tuned MQW location.
[structure] Simple VCSEL with MQW.
[comment] The location of MQW is critical for tuning of wavelength.
------------------------
pics3d_examples\VCSELS\jim_vcsel_self
[title] A simple VCSEL with finely tuned MQW location.
[structure] Simple VCSEL with MQW. Also 3 layers per period in DBR mirror.
[comment] The location of MQW is critical for tuning of wavelength.
------------------------
pics3d_examples\VCSELS\lambda_as_layer_unit
[title] A simple VCSEL with finely tuned MQW location.
[structure] Simple VCSEL with MQW.
```

```
[comment] The location of MQW is critical for tuning of wavelength.
------------------------
pics3d_examples\VCSELS\nonsymmetric
[title] Full 3D multi-mode simulation of non-symmetric VCSEL
[structure] MQW VCSEL with non-symmetric top point contact.
[comment] Demo of multiple segment simulation in cylindrical system.
------------------------
pics3d_examples\VCSELS\pump_vcsel
[title] Simulation of an optically pumped VCSEL
[structure] Simple VCSEL with MQW.
[comment] Use of shorter wavelength to generate carriers in MQW.
------------------------
pics3d_examples\VCSELS\rectangle
[title] A simple rectangle VCSEL with finely tuned MQW position.
[structure] Rectangle VCSEL with MQW.
[comment] Comparison of cylindrical with rectangle VCSEL
------------------------
pics3d_examples\VCSELS\surface_relief
[title] Simulation of surface relieved VCSEL
[structure] MQW VCSEL with outer cavity part relieved.
[comment] Use of multicavity option to treat the relieved outer cavity
------------------------
pics3d_examples\VCSELS\VCSEL_2mode_trans
[title] Transient simulation of multiple lateral modes for a VCSEL.
[structure] AlGaAs MQW VCSEL.
[comment] Two lateral modes are almost equally pumped to lasing.
------------------------
pics3d_examples\VCSELS\VCSEL_2modes
[title] Multiple lateral modes for a VCSEL.
[structure] AlGaAs MQW VCSEL.
[comment] Two lateral modes are almost equally pumped to lasing.
------------------------
pics3d_examples\VCSELS\VCSEL_eim_oxide
[title] Fused junction VCSEL at 1.55 um with oxide layer confinement.
[structure] MQW VCSEL combining AlGaAs and InGaAsP system.
[comment] Use of EIM-VCSEL model.
------------------------
pics3d_examples\VCSELS\vcsel_macro_DBR
[title] A simple VCSEL with finely tuned MQW location.
[structure] Simple VCSEL with MQW. Also 3 layers per period in DBR mirror.
[comment] The location of MQW is critical for tuning of wavelength.
------------------------
pics3d_examples\VCSELS\VCSEL_thermal_oxide
```

```
[title] Thermal simulation of VVSEL with oxide layer confinement
[structure] MQW VCSEL with oxide layer, DBR with anisotropic properties
[comment] Use of tau_model for temp. dependent intraband scattering broadening
------------------------
pics3d_examples\VCSELS\VCSEL_transient
[title] Transient simulation for VCSEL using single lateral mode
[structure] Simple VCSEL with MQW
[comment] To demonstrate the lasing oscillation in VCSEL
------------------------
```

PARS OPTICS GROUP

Bibliography

[1] S. M. Zse. *Physics of semiconductor devices*. John Wiley & Sons, 2nd edition, 1981. ISBN 978-0-471-67324-8.

[2] E.M. Azoff. Energy transport numerical simulation of graded AlGaAs/GaAs heterojunction bipolar transistors. *IEEE Trans. ED*, 36(4):609–616, 1989. doi:10.1109/16.22464.

[3] E.M. Azoff. Close-form method for solving the steady-state generalised energy-momentum conservation equations. *COMPEL*, 6(1):25–30, 1987. doi:10.1108/eb010297.

[4] R. A. Smith. *Semiconductors*. Cambridge University Press, 2nd edition, 1978. ISBN 978-0521218245.

[5] D. Bednarczyk and J. Bednarczyk. The approximation of the fermi-dirac integral $F_{1/2}(\eta)$. *Physics Letters A*, 64:409–410, 1978. doi:10.1016/0375-9601(78)90283-9.

[6] E. F. Schubert. *Physical Foundations of Solid-State Devices*, chapter 16 - High doping effects. Rensselaer Polytechnic Institute, 2009. URL `http://www.rpi.edu/~schubert/`.

[7] Valentin O. Turin. A modified transferred-electron high-field mobility model for gan devices simulation. *Solid-State Electronics*, 49:1678Ű1682, 2005. doi:10.1016/j.sse.2005.09.002.

[8] S.S. Perlman and D.L. Feucht. p-n heterojunctions. *Solid State Electronics*, 7 (12):911–923, 1964. doi:10.1016/0038-1101(64)90070-X.

[9] R.J. Schuelke and M.S. Lundstrom. Thermionic emission-diffusion theory of isotype heterojunctions. *Solid State Electronics*, 27(12):1111–1116, 1984. doi:10.1016/0038-1101(84)90051-0.

[10] R.S. Varga. *Matrix iterative analysis*. Prentice-Hall, 1962. ISBN 978-3540663218.

[11] D.L. Scharfetter and H.K. Gummel. Large-signal analysis of a silicon read diode oscillator. *IEEE Trans. Electron Devices*, 16(1):64–77, January 1969. doi:10.1109/T-ED.1969.16566.

[12] S.E. Laux. Techniques for small-signal analysis of semiconductor devices. *IEEE Trans. Electron Devices*, 32:2028–2037, 1985. doi:10.1109/T-ED.1985.22235.

[13] Q. Li and R.W. Dutton. Numerical small-signal AC modeling of deep-level-trap related frequency-dependent output conductance and capacitance for GaAs MESFET's on semi-insulating substrates. *IEEE Trans. Electron devices*, 38: 1285–1288, 1991. doi:10.1109/16.81618.

[14] D.I. Blokhintsev. *Quantum mechanics*. D. Reidel Pubslishing Co., 1964. ISBN 978-9027701046.

[15] S.L. Chuang. Efficient band-structure calculation of strained quantum-wells. *Phys. Rev. B*, 43(12):9649–9661, 1991. doi:10.1103/PhysRevB.43.9649.

[16] S.R. Chinn, P.S. Zory, and A.R. Reisinger. A model for GRIN-SCH-SQW diode lasers. *IEEE J. Quantum Electron.*, 24:2191–2214, November 1988. doi:10.1109/3.8562.

[17] J. Nagle, S. Hersee, M. Krakowski, T. Weil, and C. Weisbuch. Threshold current of single quantum well lasers: The role of the confining layers. *Appl. Phys. Lett.*, 49(20):1325, 1986. doi:10.1063/1.97366.

[18] G. Fuchs, C. Schiedel, A. Hangleiter, V. Harle, and F. Scholz. Auger recombination in strained and unstrained InGaAs/InGaAsP multiple quantum well lasers. *Appl. Phys. Lett.*, 62(4):396–398, 1993. doi:10.1063/1.108941.

[19] S. Colak, R. Eppenga, and M.F.H. Schuurmans. Band mixing effects on quantum well gain. *IEEE J. Quantum Well Electron.*, 23:960–967, 1987. doi:10.1109/JQE.1987.1073466.

[20] R. Eppenga, M.F.H. Schuurmans, and S. Colak. New $k.p$ theory for GaAs/$Ga_{1-x}Al_xAs$-type quantum wells. *Phys. Rev. B*, 36(3):1554–1564, 1987. doi:10.1103/PhysRevB.36.1554.

[21] G. Bastard. Superlattice band structure in the envelope-function approximation. *Phys. Rev. B*, 24(10):5693–5697, 1981. doi:10.1103/PhysRevB.24.5693.

[22] L.C. Andreani, A. Pasquarello, and F. Bassani. Hole subbands in strained GaAs-$Ga_{1-x}Al_xAs$ quantum wells: Exact solution of the effective-mass equation. *Phys. Rev. B*, 36:5887–5894, 1987. doi:10.1103/PhysRevB.36.5887.

[23] J.M. Luttinger and W. Kohn. Motion of electrons and holes in perturbed periodic fields. *Physical Review*, 97(4):869–883, 1955. doi:10.1103/PhysRev.97.869.

[24] J.M. Luttinger. Quantum theory of cyclotron resonance in semiconductors: general theory. *Physical Review*, 102(4):1030–1041, 1956. doi:10.1103/PhysRev.102.1030.

[25] F. Wooten. *Optical properties of solids*. Academic Press, 1972. ISBN 0127634509.

[26] M. Yamada, S. Ogita, M. Yamagishi, and K. Tabata. Anisotropy and broadening of optical gain in a GaAs/AlGaAs multiquantum-well laser. *IEEE J. Quantum Electron.*, 21:640–645, June 1985. doi:10.1109/JQE.1985.1072712.

[27] R.H. Yan, S.W. Corzine, L.A. Coldren, and I. Suemune. Corrections to the expression for gain in GaAs. *IEEE J. Quantum Electron.*, 26(2):213–216, 1990. doi:10.1109/3.44950.

[28] E. Zielinski, H. Schweizer, S. Hausser, R. Stuber, M. H. Pilkuhn, and G. Weimann. Systematics of laser operation in GaAs/AlGaAs multiquantum well heterostructures. *J. Quantum Electron.*, 23(6):969–976, 1987. doi:10.1109/JQE.1987.1073463.

[29] E. Zielinski, F. Keppler, S. Hausser, M. H., R. Sauer, and W.T. Tsang. Optical gain and loss processes in GaInAs/InP MQW laser structure. *J. Quantum Electron.*, 25(6):1407–1416, 1989. doi:10.1109/3.29276.

[30] P.T. Landsberg. A contribution to the theory of soft X-ray emission bands of sodium. *Proc. Phys. Soc. A*, 62(12):806, 1949. doi:10.1088/0370-1298/62/12/307.

[31] P.T. Landsberg. Electron interaction effects on recombination spectra. *Physica Status Solidi B*, 15(2):623–626, 1966. doi:10.1002/pssb.19660150223.

[32] R.W. Martin and H.L. Stormer. On the low energy tail of the electron-hole drop recombination spectrum. *Solid State Comm*, 22(8):523–526, May 1977. doi:10.1016/0038-1098(77)91406-5.

[33] Shun Lien Chuang. *Physics of optoelectronic devices*. John Wiley & Sons Inc., 1995. ISBN 9-780471-109396.

[34] G.P. Agrawal and N.K. Dutta. *Long wavelength semiconductor lasers*. van Nostrand Reinhold, 1986. ISBN 0442209959.

[35] A. Yariv. *Optical Electronics*. Saunders College Publishing, 4th edition, 1991. ISBN 978-0030702891.

[36] C.H. Henry, R.A. Logan, and K.A. Bertness. Spectral dependence of the change in refractive index due to carrier injection in GaAs lasers. *J. Appl. Phys.*, 52 (7):4457–4461, 1981. doi:10.1063/1.329371.

[37] K. Vahala, L.C. Chiu, S. Margalit, and A. Yariv. On the linewidth enhancement factor α in semiconductor injection lasers. *Appl. Phys. Lett.*, 42:631, 1983. doi:10.1063/1.94054.

[38] W. W. Chow and S. W. Koch. Many-body coulomb effects in room-temperature IIŰVI quantum well semiconductor lasers. *Appl.Phys.Lett*, 66:3004, 1995. doi:10.1063/1.114258.

[39] H. Haug and S. Schmitt-Rink. Electron theory of the optical properties of laser-excited semiconductors. *Progress in Quantum Electronics*, 9(1):3, 1984. doi:10.1016/0079-6727(84)90026-0.

[40] H. Haug and S. Schmitt-Rink. Basic mechanisms of the optical nonlinearities of semiconductors near the band edge. *JOSA B*, 2(7):1135–1142, 1985. doi:10.1364/JOSAB.2.001135.

[41] S. Schmitt-Rink, C. Ell, and H. Haug. Many-body effects in the absorption, gain, and luminescence spectra of semiconductor quantum-well structures. *Phys. Rev. B*, 33(2):1183Ű1189, 1986. doi:10.1103/PhysRevB.33.1183.

[42] C.F. Hsu, P.S. Zory, C.H. Wu, and M.A. Emmanuel. Coulomb enhancement in InGaAs-GaAs quantum-well lasers. *IEEE J. Sel. Top. Quant. Elec.*, 3(2), 1997. doi:10.1109/2944.605649.

[43] Peter Blood. On the dimensionality of optical absorption, gain, and recombination in quantum-confined structures. *IEEE JOURNAL OF QUANTUM ELECTRONICS*, 36(3):354, Mar. 2000. doi:10.1109/3.825883.

[44] M. Sugawara, K. Mukai, Y. Nakata, K. Otsubo, and H. Ishilkawa. Performance and physics of quantum-dot lasers with self-assembled columnar-shaped and 1.3-μm emitting InGaAs quantum dots. *IEEE J. Select. Topics in Quant. Electron.*, 6:462, 2000. doi:10.1109/2944.865101.

[45] A. A. Dikshit and J. M. Pikal. Carrier distribution, gain, and lasing in 1.3-μm InAs InGaAs quantum-dot lasers. *IEEE J. Quant. Electron.*, 40(2):105–112, Feb. 2004. doi:10.1109/JQE.2003.821532.

[46] S. Selberherr. *Analysis and simulation of semiconductor devices*. Springer-Verlag, 1984. ISBN 0387818006.

[47] A.G. Chynoweth. Ionization rates for electrons and holes in silicon. *Phys. Rev.*, 109(5):1537–1540, 1958. doi:10.1103/PhysRev.109.1537.

[48] C.R. Crowell and S.M. Sze. Temperature dependence of avalanche multiplication in semiconductors. *Appl. Phys. Lett.*, 9(6):242–244, 1966. doi:10.1063/1.1754731.

[49] Thomas Lackner. Avalanche multiplication in semiconductors: A modification of chynoweth's law. *Solid-State Electronics*, 34(1):33–42, 1991. doi:10.1016/0038-1101(91)90197-7.

[50] Y. Okuto and C. R. Crowell. Threshold energy effect on avalanche breakdown voltage in semiconductor junctions. *Solid-State Electronics*, 18(2):161–168, 1975. doi:10.1016/0038-1101(75)90099-4.

[51] B. Jayant Baliga. *Fundamentals of Power Semiconductor Devices*. Springer, 1st edition, 2008. ISBN 0387473130.

[52] R. Van Overstraeten and H. De Man. Measurement of the ionization rates in diffused silicon *p-n* junctions. *Solid State Electronics*, 13(5):583–608, 1970. doi:10.1016/0038-1101(70)90139-5.

[53] W.N. Grant. Electron and hole ionization rates in epitaxial silicon at high electric fields. *Solid State Electronics*, 16(10):1189–1203, 1973. doi:10.1016/0038-1101(73)90147-0.

[54] A.A. Grinberg, M.S. Shur, R.J. Fischer, and H. Morkoc. An investigation of the effect of graded layers and tunneling on the performance of AlGaAs/GaAs heterojunction bipolar transistors. *IEEE Trans. Electron Device*, 31:1758–1765, 1984. doi:10.1109/T-ED.1984.21784.

[55] A. Yariv. *An introduction to theory and applications of quantum mechanics*. John Wiley & Sons, 1982. ISBN 978-0471060536.

[56] F.A. Padovani and R. Stratton. Field and thermionic-field emission in schottky barriers. *Solid-State Electronics*, 9(7):695–707, 1966. doi:10.1016/0038-1101(66)90097-9.

[57] A. Messiah. *Quantum mechanics*, page 242. John-Wiley & Sons, Inc., 1958. ISBN 9780486409245.

[58] E.O. Kane. Zener tunneling in semiconductors. *Phys. Chem. Solids*, 2:181–188, 1960. doi:10.1016/0022-3697(60)90035-4.

[59] C.B Duke. *Tunneling in Solids*, volume 10 of *Solid state physics: advances in research and applications*. Academic Press, 1969. ISBN 9780126077704.

[60] J.L Moll. *Physics of Semiconductors*, chapter 12. McGraw-Hill, 1964.

[61] I. Vurgaftman, J. R. Meyer, and L. R. Ram-Mohan. Band parameters for iiiŰv compound semiconductors and their alloys. *Journal of Applied Physics*, 89(11): 5815–5875, 2001. doi:10.1063/1.1368156.

[62] Shun Lien Chuang. Optical gain of strained wurtzite GaN quantum - well lasers. *IEEE JQE*, 32(10):1791–1800, October 1996. doi:10.1109/3.538786.

[63] S. L. Chuang and C. S. Chang. $k \cdot p$ method for strained wurtzite semiconductors. *Phys. Rev. B*, 54(4):2491–2504, July 1996. doi:10.1103/PhysRevB.54.2491.

[64] S. L. Chuang and C. S. Chang. Effective - mass hamiltonian for strained wurtzite GaN and analytical solutions. *Appl. Phys. Lett.*, 68(12):1657–1659, 1996. doi:10.1063/1.115896.

[65] Seoung-Hwan Park, Doyeol Ahn, and Shun-Lien Chuang. Electronic and optical properties of a- and m-plane wurtzite InGaNÚGaN quantum wells. *IEEE Journal of Quantum Electronics*, 43(12):1175 – 1182, 2007. doi:10.1109/JQE.2007.905009.

[66] Peng-Fei Qiao, Shin Mou, and Shun Lien Chuang. Electronic band structures and optical properties of type-ii superlattice photodetectors with interfacial effect. *Optics express*, 20(3):2319–2334, 2012. doi:10.1364/OE.20.002319.

[67] M. Asada. Intraband relaxation time in quantum well lasers. *IEEE J. Quantum Electronics*, 25(9):2019–2026, September 1989. doi:10.1109/3.35228.

[68] M. Asada. Intraband relaxation effect on optical spectra. In P. S. Zory, editor, *Quantum Well Lasers*, pages 97–130. Academic Press, Inc., 1983. ISBN 978-0127818900.

[69] B. Romero, J. Arias, I. Esquivias, and M. Cada. Simple model for calculating the ratio of the carrier capture and escape times in quantum-well lasers. *Appl. Phys. Lett.*, 76(12):1504–1506, 2000. doi:10.1063/1.126077.

[70] M. Grupen and K. Hess. Simulation of carrier transport and nonlinearities in quantum-well laser diodes. *IEEE J. Quant. Electronics*, 34(1):120, Jan. 1998. doi:10.1109/3.655016.

[71] C. S. Xia, W. D. Hu, C. Wang, Z. F. Li, X. S. Chen, W. Lu, Z. M. Simon Li, and Z. Q. Li. Simulation of InGaN/GaN multiple quantum well light-emitting diodes with quantum dot model for electrical and optical effects. *Optical and Quantum Electronics*, 38:1077–1089, 2006. doi:10.1007/s11082-006-9029-5.

[72] B. F. Levine. Quantum-well infrared photodetectors. *J. Appl. Phys*, 74:R1–R81, Oct. 1993. doi:10.1063/1.354252.

[73] G.K. Wachutka. Rigorous thermodynamic treatment of heat generation and conduction in semiconductor device modeling. *IEEE Trans. CAD*, 9(11):1141–1149, 1990. doi:10.1109/43.62751.

[74] A. Marshak and K. van Vilet. Electrical current in solids with position dependent band structure. *Solid State Electron.*, 21(2):417–427, 1978. doi:10.1016/0038-1101(78)90272-1.

[75] L. Liou, J. Ebel, and C Huang. Thermal effects on the characteristics of AlGaAs/GaAs heterojunction bipolar transistors using two dimensional numerical simulation. *IEEE Trans. Elec. Dev.*, 40(1):35–43, 1993. doi:10.1109/16.249421.

[76] R. Stratton. Semiconductor current flow equations (diffusion and degeneracy). *IEEE Trans. Elec. Dev.*, 19:1288–1292, 1972. doi:10.1109/T-ED.1972.17592.

[77] M. Stern. Finite difference analysis of planar optical waveguides. In W.P. Huang, editor, *Progress In Electromagnetics Research*, volume 10, pages 123–186. EMW Publishing, 1995.

[78] G. R. Hadley, D. Botez, and L. J. Mawst. Modal discrimination in leaky-mode (antiguided) arrays. *IEEE J. Quantum Electronics*, 27(4):921–930, April 1991. doi:10.1109/3.83327.

[79] Z. S. Sacks, D.M. Kingsland, R. Lee, and Jin-Fa Lee. A perfectly matched anisotropic absorber for use as an absorbing boundary condition. *IEEE Trans. Antennas Propagat.*, 43(12):1460–1463, Dec. 1995. doi:10.1109/8.477075.

[80] C. H. Henry. Theory of spontaneous emission noise in open resonators and its application to lasers and optical amplifiers. *J. Lightwave Technol.*, 4(3): 288–297, March 1986. doi:10.1109/JLT.1986.1074715.

[81] R.F. Kazarinov and C.H. Henry. Second-order distributed feedback lasers with mode selection provided by first-order radiation losses. *IEEE Journal of Quantum Electronics*, 21(3):144–153, 1985. doi:10.1109/JQE.1985.1072627.

[82] Pochi Yeh. *Optical Waves in Layered Media*. Wiley Interscience, 1988. ISBN 978-0471828662.

[83] H. Angus MacLeod. *Thin-Film Optical Filters*. IoP, 3rd edition, 2001. ISBN 978-0750306881.

[84] M. Bulsara. Strained silicon joins the drive to keep CMOS chips on course. *Compound Semiconductor magazine*, September 2002.

[85] B. M. Haugerud, L. A. Bosworth, and R. E. Belford. Elevated-temperature electrical characteristics of mechanically strained Si devices. *J. App. Phys.*, 95 (5):2792–2796, 1 March 2004. doi:10.1063/1.1644637.

[86] B. M. Haugerud, L. A. Bosworth, and R. E. Belford. Mechanically induced strain enhancement of metal-oxide-semiconductor field effect transistors. *J. App. Phys.*, 94(6):4102–4107, 15 March 2003. doi:10.1063/1.1602562.

[87] M.V. Fischetti and S.E. Laux. Band structure, deformation potentials, and carrier mobility in strained Si, Ge, and SiGe alloys. *J. Appl. Phys.*, 80:2234–2252, 1996. doi:10.1063/1.363052.

[88] C.G. Van De Walle and Richard M. Martin. Theoretical calculations of heterojunction discontinuities in the Si/Ge system. *Phys. Rev. B*, 34(8):5621–5634, 15 Oct. 1986. doi:10.1103/PhysRevB.34.5621.

[89] Soline Richard, Frederic Aniel, Guy Fishman, and Nicolas Cavassilas. Energy-band structure in strained silicon: A 20-band $k \cdot p$ and Bir-Pikus Hamiltonian model. *J. Appl. Phys.*, 94(3):1794, 1 Aug. 2003. doi:10.1063/1.1587004.

[90] M.A. Herman. Silicon-based heterostructures: strained-layer growth by molecular beam epitaxy. *Cryst. Res. Technol.*, 34:583–595, May-June 1999. doi:10.1002/(SICI)1521-4079(199906)34:5/6<583::AID-CRAT583>3.0.CO;2-X.

[91] K. Ismail and B.S. Meyerson. Si/SiGe quantum wells: fundamentals to technology. *J. Mater. Sci.: Materials in Electronics*, 6:306, 1995. doi:10.1007/BF00125885.

[92] M. M. Rieger and P. Vogl. Electronic-band parameters in strained Si(1-x)Ge(x) alloys on Si(1-y)Ge(y) substrates. *Phys. Rev. B*, 48(19):14276–287, 15 Nov. 1993. doi:10.1103/PhysRevB.48.14276.

[93] Shinichi Takagi, Judy L. Hoyt, Jeffrey J. Welser, and James F. Gibbons. Comparative study of phonon-limited mobility of two-dimensional electrons in strained and unstrained Si metal-oxide-semiconductor field-effect transistors. *J. Appl. Phys.*, 80(3):1567–1577, 1 Aug. 1996. doi:10.1063/1.362953.

[94] B.K. Ridley. *Quantum process in semiconductors*. Oxford Univ. Press, 4th edition, 1999. ISBN 9780198505792.

[95] K. Seeger. *Semiconductor Physics : An Introduction*. Springer, 7th edition, 1999. ISBN 9783540657866.

[96] N.W. Ashcroft and N.D. Mermin. *Solid State Physics*. Brooks Cole, 1976. ISBN 9780030839931.

[97] M. T. Currie, C. W. Leitz, T. A. Langdo, G. Taraschi, E. A. Fitzgerald, and D. A. Antoniadis. Carrier mobilities and process stability of strained Si n- and p-MOSFETs on SiGe virtual substrates. *J. Vac. Sci. Technol. B*, 19(6): 2268–2279, Nov/Dec. 2001. doi:10.1116/1.1421554.

[98] B. Ryhstaller, S.A. Carter, S. Barth, H. Riel, and W. Riess. Transient and steady-state behavior of space charges in multilayer organic light-emitting diodes. *J. Appl. Phys.*, 89(8):4575–4586, 15 April 2001. doi:10.1063/1.1352027.

[99] P. Langevin. L'ionization des gaz. *Ann. Chem. Phys.*, 28:289, 1903.

[100] I. Vragovic, R. Scholz, and M. Schreiber. Model calculation of the optical properties of 3,4,9,10-perylene-tetracarboxylic-dianhydride (PTCDA) thin films. *Europhysics Letters*, 57(2):288–294, 2002. doi:10.1209/epl/i2002-00574-3.

[101] M. Hoffmann and Z. G. Soos. Optical absorption spectra of the holstein molecular crystal for weak and intermediate electronic coupling. *Phys. Rev. B*, 66: 024305, 2002. doi:10.1103/PhysRevB.66.024305.

[102] M. Hoffmann, Z. G. Soos, and K. Leo. Absorption spectra and band structure of mixed frenkel-charge-transfer vibronic states in one-dimensional molecular crystals. *Nonlinear Optics*, 29(4–6):227–237, 2002.

[103] M.H. Hennessy, Z.G. Soos, R.A. Pascal Jr., and A. Girlando. Vibronic structure of PTCDA stacks: the excitonŰphonon-charge-transfer dimer. *Chem. Phys.*, 245:199–212, 1999. doi:10.1016/S0301-0104(99)00082-8.

[104] M. Hoffmann, K. Schimidt, T. Fritz, T. Hasche, V.M. Agranovich, and K. Leo. The lowest energy Frenkel and charge-transfer excitons in quasi-one-dimensional structures: application to MePTCDI and PTCDA crystals. *Chem. Phys.*, 258:73–96, 2000. doi:10.1016/S0301-0104(00)00157-9.

[105] Z.G. Soos, M.H. Hennessy, and G. Wen. Frenkel and charge transfer states of conjugated polymers and molecules. *Chem. Phys.*, 227:19–32, 1998. doi:10.1016/S0301-0104(97)00307-8.

[106] B. Tromborg, H. Olesen, and X. Pan. Theory of linewidth for multi-electrode laser diodes with spatially distributed noise sources. *IEEE J. Quantum Electron.*, 27:178–192, 1991. doi:10.1109/3.78219.

[107] C. H. Henry and R.F. Kazarinov. Stablilization of single frequency operation of coupled-cavity lasers. *IEEE J. Quantum Electron.*, 20:733–744, 1984. doi:10.1109/JQE.1984.1072465.

[108] S. McCall and P. Platzman. An optimized $\pi/2$ distributed feedback laser. *IEEE Journal of Quantum Electronics*, 21(12):1899–1904, Dec. 1985. doi:10.1109/JQE.1985.1072605.

[109] L.D. Landau and E.M. Lifshitz. *Quantum Mechanics; non-relativistic theory*. Pergamon Press, 3rd edition, 1977. ISBN 9780750635394.

[110] E. Snitzner. Cylindrical dielectric waveguide modes. *J. Opt. Soc. Amer.*, 51: 491–498, 1961. doi:10.1364/JOSA.51.000491.

[111] Ashish M. Vengsarkars, Ioannis M. Besieris, Amr M. Shaarawi, and Richard W Ziolkowski. Closed-form, localized wave solutions in optical fiber waveguides. *J. Opt. Soc. Am. A*, 9(6):937–949, 1992. doi:10.1364/JOSAA.9.000937.

[112] G. R. Hadley, K. L. Lear, M. E. Warren, K. D. Choquette, J. W. Scott, and S.W. Corzine. Comprehensive numerical modeling of vertical-cavity surface-emitting lasers. *IEEE J. Quantum Electronics*, 32(6):607–616, June 1996. doi:10.1109/3.488833.

[113] T. Makino. Spontaneous emission model of surface-emitting DFB semiconductor lasers. *IEEE J. Quantum Electron.*, 29(1):4–22, 1992. doi:10.1109/3.199240.

[114] B. Tromborg, H. Olesen, X. Pan, and S. Saito. Transmission line description of optical feedback and injection locking for Fabry-Perot and DFB lasers. *IEEE J. Quantum Electron.*, 23:1875–1889, Nov. 1987. doi:10.1109/JQE.1987.1073251.

[115] B. Tromborg, H. E. Lassen, H. Olesen, and X. Pan. Travelling wave method for calculation of linewidth, frequency tuning and stability of semiconductor lasers. *IEEE Photon. Technol. Lett.*, 4(9):985–988, 1992. doi:10.1109/68.157123.

[116] D.C Streit and F.G Allen. Silicon triangular diodes by mbe using solid-phase epitaxial growth. *IEEE Electron Device Letters*, 5(7):354–6, 1984. doi:10.1109/EDL.1984.25908.

[117] V. Fiorentini, F. Bernardini, and O. Ambacher. Evidence for nonlinear macroscopic polarization in iii-v nitride alloy heterostructures. *Appl. Phys. Lett.*, 80 (7):1204, 2002. doi:10.1063/1.1448668.

[118] Vladimir A. Fonoberov and Alexander A. Balandin. Optical properties of wurtzite and zinc-blende GaN/AlN quantum dots. *J. Vac. Sci. Technol. B*, 22 (4):2190–2194, Jul/Aug 2004. doi:10.1116/1.1768188.

[119] M.F. Khodr, P.J. McCann, and B.A. Mason. Effects of band nonparabolicity on the gain and current density in EuSe-$PbSe_{0.78}$ $Te_{0.22}$-EuSe IV-VI semiconductor quantum-well lasers. *IEEE Journal of Quantum Electronics*, 32(2):236 – 247, 1996. doi:10.1109/3.481871.

[120] H. Shen, M. Wraback, H. Zhong, A. Tyagi, S. P. DenBaars, S. Nakamura, and J.S. Speck. Unambiguous evidence of the existence of polarization field crossover in a semipolar ingan/gan single quantum well. *Appl. Phys. Lett.*, 95: 033503, 2009. doi:10.1063/1.3167809.

[121] L.A. Coldren and S.W. Corzine. *Diode Lasers and Photonic Integrated Circuits*. Wiley-Interscience, 1995. ISBN 978-0471118756.

[122] Chris Kocot. private communication, Jan. 2014.

[123] G. Masetti, M. Severi, and S. Solmi. Modeling of carrier mobility against carrier concentration in arsenic-, phosphorus- and boron-doped silicon. *IEEE Transactions on Electron Devices*, 30(7):764Ű769, July 1983. doi:10.1109/T-ED.1983.21207.

[124] N. D. Arora, J. R. Hauser, and D. J. Roulston. Electron and hole mobilities in silicon as a function of concentration and temperature. *IEEE Transactions on Electron Devices*, 29(2):292–295, February 1982. doi:10.1109/T-ED.1982.20698.

[125] S. Reggiani, M. Valdinoci, L.Colalongo, and G.Baccarani. A unified analytical model for bulk and surface mobility in si n- and p-channel mosfets. In *Proceeding of the 29th European Solid-State Device Research Conference (ESSDERC)*, volume 1, pages 240–243, September 1999. ISBN 2-86332-245-1. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1505484&isnumber=32274.

[126] E. Conwell and V. F. Weisskopf. Theory of impurity scattering in semiconductors. *Phys. Rev.*, 77:388–390, Feb 1950. doi:10.1103/PhysRev.77.388. URL http://link.aps.org/doi/10.1103/PhysRev.77.388.

[127] H Brooks. Scattering by ionized impurities in semiconductors. In *Physical Review*, volume 83, pages 879–879. AMERICAN PHYSICAL SOC ONE PHYSICS ELLIPSE, COLLEGE PK, MD 20740-3844 USA, 1951.

[128] D.B.M. Klaassen. A unified mobility model for device simulationŮi. model equations and concentration dependence. *Solid-State Electronics*, 35(7):953 – 959, 1992. ISSN 0038-1101. doi:10.1016/0038-1101(92)90325-7.

[129] Shin-ichi Takagi, Akira Toriumi, Masao Iwase, and Hiroyuki Tango. On the universality of inversion layer mobility in si mosfet's: Part i-effects of substrate impurity concentration. *Electron Devices, IEEE Transactions on*, 41(12):2357–2362, December 1994. doi:10.1109/16.337449.

[130] C. Lombardi, S. Manzini, A Saporito, and M. Vanzi. A physically based mobility model for numerical simulation of nonplanar devices. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 7(11):1164–1171, Nov 1988. ISSN 0278-0070. doi:10.1109/43.9186.

[131] Z. Yu, D. Chen, L. So, R. W. Dutton, et al. *PISCES-2ET and Its Application Subsystems*. Stanford University, 1984. URL http://www-tcad.stanford.edu/tcad/programs/pisces2et.html.

[132] C. Lombardi, S. Manzini, A. Saporito, and M. Vanzi. A physically based mobility model for numerical simulation of non-planar devices. *IEEE CAD*, 7 (11):1164–1171, Nov. 1988. doi:10.1109/43.9186.

[133] F. Szmulowicz et al. Effect of interfaces and the spin-orbit band on the band gaps of inas/gasb superlattices beyond the standard envelope-function approximation. *Phys. Rev. B*, 69:155321, 2004. doi:10.1103/PhysRevB.69.155321.

[134] Zhibin Ren. *NANOSCALE MOSFETS: PHYSICS, SIMULATION AND DESIGN*. PhD thesis, Purdue University, 2001. URL http://docs.lib.purdue.edu/dissertations/AAI3075713/. Also available at: http://nanohub.org/resources/1918/download/zhibin_ren_thesis.pdf.

[135] Z. M. Simon Li et al. Simulation of quantum cascade lasers. *J. Appl. Phys.*, 2011. doi:10.1063/1.3660207.

[136] G. A. Baraff. Semiclassical description of electron transport in semiconductor quantum-well devices. *Phys. Rev. B.*, 55(16):10745–10753, April 1997. doi:10.1103/PhysRevB.55.10745.

[137] C. S. Xia et al. Simulation of ingan/gan multiple quantum well light-emitting diodes with quantum dot model for electrical and optical effects. *Optical and Quantum Electronics*, 38:1077–1089, 2006. doi:10.1007/s11082-006-9029-5.

[138] Justin Iveland, Lucio Martinelli, Jacques Peretti, James S. Speck, and Claude Weisbuch. Direct measurement of auger electrons emitted from a semiconductor light-emitting diode under electrical injection: Identification of the dominant mechanism for efficiency droop. *Phys. Rev. Lett.*, 110:177406, Apr 2013. doi:10.1103/PhysRevLett.110.177406.

[139] Francesco Bertazzi, Michele Goano, Xiangyu Zhou, Marco Calciati, Giovanni Ghione, Masahiko Matsubara, and Enrico Bellotti. Comment on" direct measurement of auger electrons emitted from a semiconductor light-emitting diode under electrical injection: Identification of the dominant mechanism for efficiency droop"[phys. rev. lett. 110, 177406 (2013)]. *arXiv preprint arXiv:1305.2512*, 2013. URL http://arxiv.org/pdf/1305.2512.pdf.

[140] J. Kim et al. Theoretical and experimental study of optical gain and linewidth enhancement factor of type-i quantum-cascade lasers. *IEEE Journal of Quantum Electronics*, 40(12):1663 – 1674, 2004. doi:10.1109/JQE.2004.837666.

[141] Kale J. Franz. *New quantum cascade laser architectures: IIŰVI quantum cascade emitters, high k-space lasing and short injectors*. PhD thesis, Princeton University, 2009. URL http://search.proquest.com/docview/304987367.

[142] C. De Santi, M. Meneghini, M. La Grassa, B. Galler, R. Zeisel, M. Goano, S. Dominici, M. Mandurrino, F. Bertazzi, G. Meneghesso, and E. Zanoni. Role of defects in the thermal droop of ingan-based light emitting diodes. *Journal of Applied Physics*, 119(9):094501, 2016. doi:10.1063/1.4942438.

[143] Andreas Gehring. *Simulation of Tunneling in Semiconductor Devices*. PhD thesis, Vienna University of Technology, 2003. URL http://www.iue.tuwien.ac.at/phd/gehring/node61.html.

[144] Andreas Wacker. Semiconductor superlattices: a model system for nonlinear transport. *Physics Reports*, 357(1):1–111, Jan. 2002. doi:10.1016/S0370-1573(01)00029-1.

[145] Kenji Nakamura, Akira Shimizu, Masanori Koshiba, and Kazuya Hayata. Finite-element analysis of the miniband structures of semiconductor superlattices with arbitrary periodic potential profiles. *IEEE JQE*, 27(8):2035, Aug. 1991. doi:10.1109/3.83413.

[146] A. Schenk. Rigorous theory and simplified model of the band-to-band tunneling in silicon. *Solid-State Electronics*, 36(1):19–34, 1993. doi:10.1016/0038-1101(93)90065-X.

[147] Andreas Schenk Alexander Heigl and Gerhard Wachutka. Correction to the schenk model of band-to-band tunneling in silicon applied to the simulation of nanowire tunneling transistors. In *13th International Workshop on Computational Electronics*, pages 1–4, 2009. doi:10.1109/IWCE.2009.5091099.

[148] Raymond Woo. *Band-to-band tunneling transistor scaling and design for low-power logic applications*. PhD thesis, Stanford University, 2009.

[149] Mehmet Dogan, Christopher P Michael, Yan Zheng, Lin Zhu, and Jonah H Jacob. Two photon absorption in high power broad area laser diodes. In *Proc. SPIE*, volume 8965, page 89650P, 2014.

[150] S. Blaser et al. Characterization and modeling of quantum cascade lasers based on a photon-assisted tunneling transition. *IEEE Journal of Quantum Electronics*, 37(3):448–455, 2001. doi:10.1109/3.910456.

[151] Kazuya Matsuzawa, Ken Uchida, and Akira Nishiyama. A unified simulation of schottky and ohmic contacts. *IEEE TRANSACTIONS ON ELECTRON DEVICES*, 47(1):103–108, 2000. doi:10.1109/16.817574.

[152] Boni K. Boksteen. *A Simulation Study and Analysis of advanced Silicon Schottky Barrier Field Effect Transistors*. PhD thesis, University of Twente, 2010. URL http://purl.utwente.nl/essays/59426.

[153] S. Adachi. Material parameters of $In_{1-x}Ga_xAs_yP_{1-y}$ and related binaries. *J. Appl. Phys.*, 53:8775–8792, 1982. doi:10.1063/1.330480.

[154] L. Landolt-Bornstein. In O. Madelung, editor, *Numerical data and functions in science and technology*, volume 17a & 22a. Springer, 1982.

[155] L.D. Landau and E.M. Lifshitz. Electromagnetic flutuation. In *Electrodynamics of continuous media*, chapter 18. Addison-Wesley, 1960. ISBN 9780750626347.

PARS OPTICS GROUP

Index