

# PICWAVE

Version 3.0

© 2007-2009 Copyright Photon Design



*propagating ideas*

---

34 Leopold St  
Oxford, OX4 1TW  
United Kingdom  
Phone +44 1865 324990  
Fax +44 1865 324991  
info@photond.com  
www.photond.com

© 2007-2009 This manual is copyright of Photon Design.  
Unauthorised copying is strictly prohibited.

© 2007-2009 The program PICWAVE is  
copyright of Photon Design.  
Unauthorised copying in any form is strictly prohibited.

Photon Design  
34 Leopold St  
Oxford OX4 1TW  
United Kingdom

Phone +44 1865 324 990  
Fax +44 1865 324 991  
Email: [info@photond.com](mailto:info@photond.com)

# CONTENTS

<b>INTRODUCTION.....</b>	<b>1-1</b>
1.1.1 <i>New for Version 3.0</i> .....	1-1
1.1.2 <i>New for Version 2.0</i> .....	1-2
1.1.3 <i>New for Version 1.1</i> .....	1-3
1.1.4 <i>New for Version 1.01</i> .....	1-3
1.1.5 <i>Version 1.0</i> .....	1-3
<b>LEARNING PICWAVE.....</b>	<b>2-1</b>
2.1 SIMULATING A MACH ZEHNDER INTERFEROMETER .....	2-2
2.1.1 <i>Adding a device</i> .....	2-2
2.1.2 <i>Adding a waveguide</i> .....	2-3
2.1.3 <i>Constructing the MZI Device</i> .....	2-10
2.1.4 <i>Adding Monitors</i> .....	2-15
2.1.5 <i>Running the TDTW Calculator</i> .....	2-16
2.1.6 <i>Inspecting Results</i> .....	2-18
2.2 SIMULATING A FABRY-PEROT LASER .....	2-20
2.2.1 <i>Creating an active waveguide</i> .....	2-20
2.2.2 <i>Build the FP Laser Device</i> .....	2-21
2.2.3 <i>Adding some Optical Monitors</i> .....	2-24
2.2.4 <i>Adding some Active Layer Instruments</i> .....	2-24
2.2.5 <i>Running the TDTW Calculator for a laser simulation</i> .....	2-24
2.2.6 <i>Generating an Eye Diagram</i> .....	2-27
2.3 SIMULATING A BRAGG GRATING WITH AN FIR FILTER.....	2-28
2.3.1 <i>Preparing the circuit</i> .....	2-28
2.3.2 <i>Importing a scattering matrix spectrum</i> .....	2-29
2.3.3 <i>Setting the zStep and central simulation wavelength</i> .....	2-29
2.3.4 <i>Inspecting the FIR filter, FIR section properties</i> .....	2-30
2.3.5 <i>Finishing the circuit and running the TDTW simulation</i> .....	2-31
2.4 SIMULATING AN SOA WITH A MULTIPLE-LORENTZIAN GAIN MODEL .....	2-33
2.4.1 <i>Importing the gain data</i> .....	2-33
2.4.2 <i>Creating the RWG</i> .....	2-34
2.4.3 <i>Creating the SOA circuit</i> .....	2-36
2.4.4 <i>Fitting a multiple Lorentzian Gain model</i> .....	2-38
2.4.5 <i>Running the simulation and inspecting the device gain of the SOA</i> .....	2-41
2.5 OTHER EXAMPLES .....	2-42
<b>BASIC TOOLS .....</b>	<b>3-1</b>
3.1 THE SELECT TOOL.....	3-1
3.1.1 <i>Selecting Objects</i> .....	3-1
3.1.2 <i>Moving Objects</i> .....	3-2
3.1.3 <i>Nudging Objects</i> .....	3-2
3.1.4 <i>Deleting Objects</i> .....	3-3
3.1.5 <i>Copying Objects</i> .....	3-3
3.1.6 <i>Object Properties</i> .....	3-3
3.1.7 <i>Rotating Objects</i> .....	3-4

# CONTENTS

3.2	SCROLLING .....	3-4
3.3	ZOOMING .....	3-4
3.3.1	<i>The Zoom Tool</i> .....	3-4
3.3.2	<i>The Zoom to Fit Button</i> .....	3-5
3.3.3	<i>Other ways to Zoom</i> .....	3-5
3.4	UNDOING AND REDOING .....	3-6
3.5	THE GRID .....	3-6
3.5.1	<i>Grid Settings</i> .....	3-7
3.5.2	<i>Snap Settings</i> .....	3-7
<b>3.6</b>	<b>CIRCUITS .....</b>	<b>4-1</b>
4.1	CREATING A CIRCUIT .....	4-1
4.2	SHOWING CIRCUITS .....	4-1
4.3	SAVING CIRCUITS .....	4-2
4.4	RENAMING CIRCUITS .....	4-2
4.5	COPYING CIRCUITS .....	4-2
4.6	DELETING CIRCUITS .....	4-3
4.7	ADDING NOTES TO A PROJECT OR CIRCUITS .....	4-3
4.8	CIRCUIT PROPERTIES .....	4-5
4.9	SECTIONS .....	4-5
4.9.1	<i>The Waveguide Section</i> .....	4-5
4.9.2	<i>The Taper Section</i> .....	4-8
4.9.3	<i>The Y-Junction</i> .....	4-10
4.9.4	<i>The Directional Coupler</i> .....	4-10
4.9.5	<i>The Mach Zehnder Interferometer (MZI)</i> .....	4-11
4.9.6	<i>The Finite Impulse Response Filter Section (FIR Section)</i> .....	4-12
4.10	JOINS .....	4-12
4.10.1	<i>The Facet Join</i> .....	4-12
4.10.2	<i>The Power Splitter Join</i> .....	4-14
4.10.3	<i>The Power Coupler Join</i> .....	4-15
4.10.4	<i>The General Join</i> .....	4-16
4.10.5	<i>Join Scattering Matrices</i> .....	4-17
4.11	CONNECTING UP A CIRCUIT .....	4-21
4.12	ADDING AN OPTICAL SOURCE .....	4-22
4.13	ADDING AN ELECTRICAL DRIVE .....	4-23
4.14	MONITORS .....	4-25
4.15	INSTRUMENTS .....	4-25
4.15.1	<i>Monitor Instruments</i> .....	4-26
4.15.2	<i>Junction Instruments</i> .....	4-28
4.15.3	<i>Source Instruments</i> .....	4-30
4.15.4	<i>Drive Instruments</i> .....	4-30
4.15.5	<i>Contact Instruments</i> .....	4-30
4.16	SIGNAL EDITOR .....	4-31
4.16.1	<i>Creating a Signal Function</i> .....	4-31
4.16.2	<i>Further Details On Signals</i> .....	4-33

# CONTENTS

4.17 CONTACTS .....	4-33
4.17.1 Contact Properties .....	4-34
4.17.2 Travelling Wave Model Properties .....	4-35
4.18 MODE AND POLARISATION SETTINGS : GENERAL NOTES .....	4-35
<b>WAVEGUIDE CROSS-SECTIONS .....</b>	<b>5-1</b>
5.1 THE RWG WAVEGUIDE .....	5-1
5.1.1 General Usage within PICWave .....	5-1
5.1.2 The RWG Window .....	5-2
5.1.3 The RWG Editor Panel .....	5-4
5.1.4 The SWG Editor Panel .....	5-5
5.1.5 The RWG parameters. ....	5-5
5.1.6 The etching mechanism .....	5-6
5.1.7 Sharing the same SWG within the RWG .....	5-6
5.1.8 Sharing SWG's across RWG's: the SWG node .....	5-7
5.1.9 The SWG parameters .....	5-9
5.1.10 Using a material database to define material layer .....	5-10
5.1.11 Anisotropic waveguides .....	5-10
5.2 BOUNDARY CONDITIONS .....	5-11
5.2.1 The IMPEDANCE boundary condition .....	5-11
5.3 DEFINING WAVEGUIDES - GUIDELINES .....	5-11
5.3.1 Boundaries .....	5-11
5.4 THE MODE SOLVERS .....	5-12
5.5 THE MODE FINDER PANEL .....	5-12
5.6 THE MOLAB .....	5-14
5.6.1 The MOLAB Options .....	5-14
5.6.2 Finding and inspecting modes .....	5-15
5.6.3 Checking mode accuracy .....	5-16
5.6.4 Polishing a list of modes .....	5-16
5.7 THE INSPECT MODE PANEL .....	5-17
5.8 THE FDM MODE SOLVER .....	5-20
5.8.1 Options .....	5-20
5.8.2 Using Symmetry Planes .....	5-20
5.8.3 Applications .....	5-21
5.8.4 Capabilities and Limitations .....	5-21
5.8.5 Notes .....	5-22
5.8.6 The Molab and the FDM Solver .....	5-22
5.9 THE EFFECTIVE INDEX SOLVER .....	5-22
5.9.1 Options .....	5-23
<b>GRATINGS .....</b>	<b>6-1</b>
6.1 COUPLINGPARMS .....	6-1
6.2 CHIRPED GRATINGS .....	6-2
6.3 FACET PHASE AND GRATINGS .....	6-3
6.4 GRATING SET .....	6-3

# CONTENTS

6.4.1 Grating Set Editor .....	6-3
6.4.2 Grating Profiles.....	6-4
6.4.3 The user-defined grating file.....	6-5
6.4.4 Using a grating profile to define a grating in an RWG .....	6-6
6.4.5 The Kappa Calculator .....	6-6
<b>THE FIR SECTION.....</b>	<b>7-1</b>
7.1 OVERVIEW AND GENERAL USAGE.....	7-1
7.1.1 The Model.....	7-1
7.1.2 Usage .....	7-2
7.1.3 Properties .....	7-4
7.1.4 S-Matrix Spectrum Text Format.....	7-4
7.2 PICWAVE-FIMMPROP LINKS: CREATING S-MATRIX SPECTRA.....	7-5
7.2.1 Using the Script.....	7-6
7.2.2 S-Matrix Spectrum Script Options .....	7-7
<b>MULTIPLE LORENTZIAN GAIN FITTER.....</b>	<b>8-1</b>
8.1 OVERVIEW .....	8-1
8.2 MULTIPLE LORENTZIAN GAIN FITTER TOOL.....	8-2
8.2.1 Main interface .....	8-2
8.2.2 Pre-fit Analysis .....	8-4
8.2.3 Gain fitting parameters.....	8-5
8.3 FITTING A GAIN MODEL .....	8-7
8.3.1 Importing gain data .....	8-7
8.3.2 Fitting a gain model from imported data.....	8-10
8.3.3 Notes: Choice of fitting region .....	8-13
<b>THE TDTW CALCULATOR.....</b>	<b>9-1</b>
9.1 MAIN PANEL.....	9-1
9.1.1 The Run-Time Monitor Window.....	9-1
9.1.2 Simulation Control Parameters .....	9-1
9.1.3 Signals.....	9-2
9.1.4 Running a Simulation and Inspecting Results .....	9-2
9.2 INSTRUMENTS PANEL.....	9-3
9.3 CONTROL PANEL .....	9-3
9.3.1 Startup – Device State Control.....	9-4
9.3.2 Startup – Control Parameters .....	9-4
9.3.3 Noise Sources .....	9-5
9.3.4 Carrier Dynamics Acceleration .....	9-5
9.4 THE TIME DOMAIN RESULTS WINDOW .....	9-6
9.5 INSTRUMENT RESULTS .....	9-6
9.5.1 Plotting Instrument Results.....	9-7
9.5.2 X-Axis.....	9-7
9.5.3 Exporting Raw Optical Data.....	9-7
9.6 SPECTRA RESULTS.....	9-7
9.6.1 Optical spectra .....	9-8

# CONTENTS

9.6.2 Intensity and RIN Spectra.....	9-10
9.6.3 The Mode Tracker.....	9-10
9.6.4 SMSR.....	9-12
9.7 THE OSCILLOSCOPE.....	9-13
9.7.1 Syncing.....	9-14
9.7.2 Parameters.....	9-14
9.7.3 Print and Export.....	9-15
9.7.4 Pulse Analysis.....	9-15
9.7.5 Histogram Plots.....	9-16
9.8 THE LI FIT PANEL.....	9-16
9.8.1 Running a Fit.....	9-17
9.8.2 Fitted Parameters.....	9-17
9.9 INSPECTING THE DEVICE STATE.....	9-18
9.9.1 Z-Profiles.....	9-18
9.9.2 Cross-section State.....	9-21
<b>THE MATERIAL DATABASE.....</b>	<b>10-1</b>
10.1.1 Material Parameter Reference Tables.....	10-3
10.1.2 Refractive Index Functions.....	10-11
10.1.3 The Gain Model.....	10-12
10.1.4 Density Of States.....	10-12
10.1.5 Built-in AlGaAs model.....	10-13
10.1.6 Built-in InGaAsP model.....	10-13
10.1.7 Built-in InGaAlAs model.....	10-14
10.1.8 Metals.....	10-15
10.1.9 Inspecting Material Databases.....	10-15
<b>THE THEORETICAL MODEL.....</b>	<b>11-1</b>
11.1 THE ELECTRICAL MODEL.....	11-1
11.1.1 The Connection Model.....	11-1
11.1.2 The Current Flow Model.....	11-2
11.1.3 The Travelling Wave Electrode Model.....	11-2
11.2 THE OPTICAL MODEL.....	11-3
11.3 ALGORITHM FLOWCHART.....	11-5
11.4 THE GAIN MODEL.....	11-5
11.4.1 Single Lorentzian Model.....	11-5
11.4.2 Multiple Lorentzian Model.....	11-7
11.5 THE ELECTRO-ABSORPTION MODULATOR MODEL.....	11-7
11.6 KAPPA CALCULATOR – THEORY.....	11-8
<b>VARIABLES AND EXPRESSIONS.....</b>	<b>12-1</b>
12.1 EXPRESSIONS.....	12-1
12.1.1 Numbers in Expressions.....	12-2
12.1.2 Mathematical Constants in Expressions.....	12-2
12.1.3 Mathematical Operators in Expressions.....	12-2
12.1.4 Mathematical Functions in Expressions.....	12-3

# CONTENTS

12.1.5 Variables in Expressions .....	12-3
12.2 VARIABLES NODES.....	12-3
12.2.1 Creating a Variables Node.....	12-4
12.3 VARIABLES SCOPE.....	12-4
12.4 VARIABLES .....	12-5
12.4.1 Creating Variables.....	12-5
12.4.2 Setting Variables .....	12-7
12.4.3 Renaming Variables.....	12-8
12.4.4 Changing the Order of Variables.....	12-8
12.4.5 Deleting Variables .....	12-8
<b>COMMAND-LINE AND CLIENT/SERVER INTERFACE.....</b>	<b>13-1</b>
13.1 THE COMMAND-LINE WINDOW .....	13-1
13.2 USING THE COMMAND-LINE INTERFACE.....	13-2
13.2.1 The command syntax.....	13-2
13.2.2 Discovering variables and functions.....	13-2
13.2.3 Accessing subnodes.....	13-4
13.2.4 The log files .....	13-5
13.3 BATCH COMMANDS .....	13-5
13.3.1 An Example .....	13-5
13.3.2 Copying commands from a text editor.....	13-6
13.3.3 Running a script file .....	13-6
13.3.4 Managing scripts.....	13-6
13.4 COMMAND-LINE OPTIONS.....	13-7
13.5 SYNTAX REFERENCE.....	13-7
13.5.1 Basics .....	13-7
13.5.2 Variables.....	13-7
13.5.3 References and global objects .....	13-7
13.5.4 Commands.....	13-8
13.5.5 Comments.....	13-8
13.6 REMOTE TCP/IP INTERFACE.....	13-9
13.6.1 Operation.....	13-9
13.7 SCRIPTING .....	13-9
13.7.1 Installing Python.....	13-10
13.7.2 Example Python Script .....	13-11
13.7.3 pdApp Class Member Summary.....	13-16
13.7.4 Python Client.....	13-19
13.7.5 Using C++ .....	13-20
<b>TOOLS AND SETTINGS.....</b>	<b>14-1</b>
14.1 THE BRAGG WAVELENGTH CALCULATOR.....	14-1
14.2 THE MATERIALS INSPECTOR.....	14-1
14.3 THE MICROSTRIP CALCULATOR .....	14-2
14.4 THE FUNCTION EDITOR.....	14-4
14.5 APPLICATION SETTINGS.....	14-5



# CONTENTS

14.6 DEVICE PREFERENCES .....	14-5
<b>TASKS .....</b>	<b>15-1</b>
15.1 GENERATING AN LI CURVE .....	15-1
15.2 MODULATION RESPONSE .....	15-1
15.3 CALCULATING A RIN SPECTRUM .....	15-2
15.4 GENERATING AN EYE DIAGRAM.....	15-3
<b>SCIGRAPH.....</b>	<b>16-1</b>
16.1 SETUP .....	16-1
16.2 STATIC PLOT.....	16-1
16.2.1 Printing.....	16-2
16.3 THE XY CHART PLOT .....	16-3
16.3.1 Exporting to Microsoft Excel.....	16-4
16.4 THE 3D CHART PLOT .....	16-4
16.5 THE VECTOR FIELD PLOT.....	16-6
<b>APPENDIX .....</b>	<b>17-1</b>
17.1 VALIDITY TESTS.....	17-1
17.2 MODE PROPERTIES - DEFINITIONS.....	17-2
17.2.1 Mode Loss.....	17-2
17.2.2 Confinement Factor .....	17-2
17.2.3 Filling factor.....	17-3
17.2.4 TE Fraction .....	17-3
17.2.5 Intensity.....	17-3
17.2.6 Effective Core Area.....	17-3
17.3 DISPERSION AND GROUP INDEX .....	17-4
17.4 SPONTANEOUS EMISSION AND SPONTANEOUS COUPLING FACTOR (SPONBETANA) .....	17-4
17.5 THE ASCII MODE FORMAT FILE (.AMF) .....	17-5
<b>INDEX.....</b>	<b>18-1</b>

## Introduction

PICWAVE is a comprehensive photonic integrated circuit simulator including a full laser diode/SOA model for active as well as passive PICs. PICWAVE allows one to construct optical circuits of near arbitrary topology including Mach-Zehnder interferometers, ring resonators as well as the more traditional linearly connected components. It is based on a time domain engine enabling it to model both linear and non-linear elements. Unlike other commonly used device simulators, PICWAVE can model devices with optical path lengths of cm or even metres. It is also fully bi-directional, taking all reflections of the circuit into account.

In addition PICWAVE has a comprehensive laser diode/SOA model that takes a complete 3D description of the device structure as a starting point. The model will simulate the DC, time-domain and frequency domain characteristics of the device, including the effect of chirp, linewidth and spontaneous emission noise or phase noise. The laser/SOA takes into account many physical processes, including lateral carrier diffusion, spectral hole burning, spatial hole burning in the lateral and longitudinal direction, non-linear gain, inter-valence-band absorption.

PICWAVE uses a “slowly varying envelope approximation” to make calculations more efficient. So a time and space varying signal:

$$A(t,z)$$

is decomposed into a fast but time and space independent part plus a slow part that may vary in time and space:

$$A(t,z) = a(t,z) \cdot \exp(i(\beta_0 z - \omega_0 t))$$

where  $\omega_0$  is some optical frequency near the centre of the optical spectrum of interest and  $\beta_0$  is a characteristic propagation constant for  $\omega_0$  in the waveguide, typically the propagation constant of the zero order mode at  $\omega_0$ .

Changes Log

### 1.1.1 New for Version 3.0

- Multiple Lorentzian Gain Model – PICWAVE now has a self-contained multiple Lorentzian gain fitting tool which produces fitted gain models from a set of imported gain spectra. A multiple Lorentzian model enables greater fidelity to the imported gain spectra over a large portion of the free spectral range, allowing the gain to be modelled accurately away from the gain peak. This is in contrast to the original single Lorentzian model, which can accurately model the gain only in the region close to the gain peak.
  - Raw gain data imported in text file format
  - A range of fitting options available, including extrapolation and scaling of raw imported data

- Can plot raw spectra and fitted spectra results
- Can fit separate TE/TM gain models, in line with PICWave's new support of 2 polarisations.
- Taper Sections – PICWAVE can now model linear taper sections, both active and passive. The taper cross-section can be inspected at any point along the length of a taper section and the mode properties can be plotted as a function of position along the taper
- Lorentzian Noise Spectrum – PICWAVE now supports coloured noise with a Lorentzian spectrum (previously only white noise was available). The user can define the noise spectrum peak wavelength and half-width as functions of carrier density and temperature
- FDM Solver/RWG Editor Upgrade – A high performance full-vector 2D Finite Difference Mode Solver is now available in addition to the effective index mode solver. The RWG Editor interface has also been upgraded and is now identical to that of FIMMWAVE. The user can now select which modes to propagate in TDTW simulations
- Improved Multimode Functionality – In line with mode solver additions/improvements, waveguide (and taper) sections in circuits are no longer confined to propagating an equal number of TE and TM modes when modes of both polarisations are to be propagated.
- Grating Editor – A user can now define physical gratings for waveguide sections, from which grating coupling parameters are calculated automatically by the Grating Editor's kappa calculator.
- Merge Active Layers Feature – Simulation time can be reduced ( $\sim 2x/3x$ ) by computationally merging active layers in waveguides where all active layers are identical
- Time-Dependent Gain Saturation – The gain non-linearity ( $\epsilon$ ) can now decay according to a user-defined time constant.
- Can now plot Frequency Modulation (FM) spectra

#### 1.1.2 New for Version 2.0

- Multiple modes now supported
  - Passive waveguide and other sections can propagate one mode (TE or TM) or two modes (TE + TM);
  - Active waveguides and joins can support an arbitrary number of modes when polarisation is set to only TE or TM, and an arbitrary even number of modes for mixed polarisation (with half of the modes in each polarisation).
  - The general join now allows complete generality regarding the number and polarisation of modes of each of the adjoining section (i.e. the settings of all adjoining sections need not match), and allows the coupling between any and all of the modes of the attached waveguides to be specified. The use of general joins thus allows multiple modes and both polarisations to be supported throughout the *whole* circuit.
  - Multiple mode functionality also extends to instruments and Z-profile measurements.

- MOLAB options now supports both polarisations simultaneously, and is accessible directly from the device window.
- FIR filter sections can be inspected and adjusted from the device window (via right-click menu) before running a simulation.
- Spectra and monitor instrument results now accessible from (right-click menu) monitor menu in device window.
- Quantum Efficiency monitor instrument added.
- Sections can now be joined directly (without an intermediate facet)

#### 1.1.3 New for Version 1.1

- Can now propagate both TE and TM polarisations simultaneously through the circuit – all components can have polarisation dependent properties. Can define separate TE and TM gain in active sections.
- Import a wavelength dependent scattering matrix of arbitrary component – modelled in the time-domain using FIR filter. This provides a link for importing rigorous FIMMPROP (a Photon Design frequency domain model) simulations into PICWAVE.
- Additional passive components – directional coupler, Y-junction, MZI.
- Now a physical waveguide cross-section is optional – one can just enter the effective index etc of the mode directly if one prefers
- “Natural” joint – a facet join option which computes the reflection coefficients from the effective indices of the adjoining waveguides.
- Rotation of sections as well as joints now.

#### 1.1.4 New for Version 1.01

- improved spectral plotting, e.g. now with reference monitor.

#### 1.1.5 Version 1.0

- based on CLADISS-2D version 2.1

# Chapter

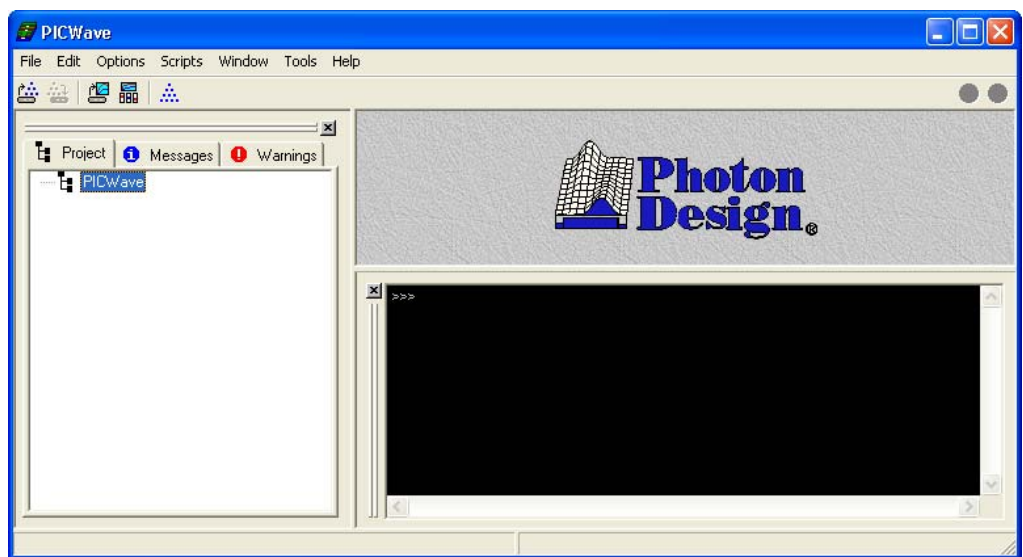
# 2


## Learning PICWAVE

This chapter will attempt to teach you to use the program by example. After following this you should be able to do most things and if not, know where to find out more.

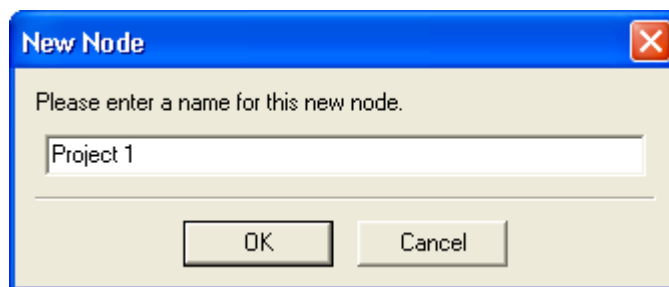
We will start by creating a *device*.

- Start up PICWAVE from the file picwave.exe. The Main Window will appear:





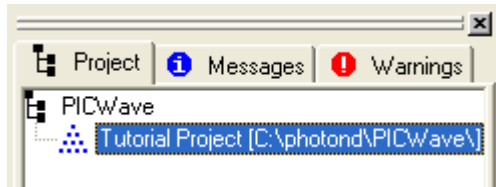
- To create a new *project*, click on the  Add Project button.

The *New Node dialog* will appear for you to enter the name of the new *project*.



- Type **Tutorial Project**, then click .



The *project* will now be shown with the project symbol  under the root of the *Project Tree*, which has the symbol  (you may have to scroll to the left in the *Project Window* to see these symbols):




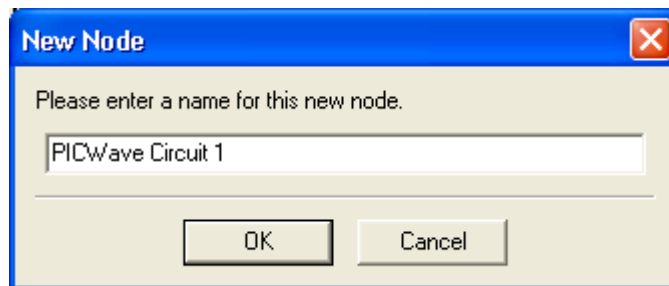
You can have as many different *projects* open in the application as you like.

## 2.1 Simulating a Mach Zehnder Interferometer



### 2.1.1 Adding a device

- To create a new *device* in a project, click on the  Tutorial Project in the *Project Tree*, then click on the  Add PICWave Circuit button

Note that this button will appear only if the  Tutorial Project is selected in the *Project Tree*. Again, the *New Node dialog* will appear for you to enter the name of the new device:




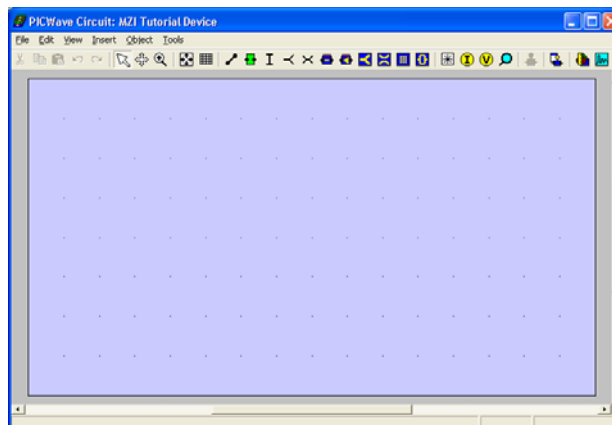
- Type **MZI Tutorial Device**, then click .

The *device* will be shown in the *Project Tree* with the device symbol  under the  Tutorial Project.



You can add as many different *devices* to a *project* as you like.


- To show the *device*, double-click on the  MZI Tutorial Device line in the *Project Tree*. A *device window* will appear showing an empty *device*:

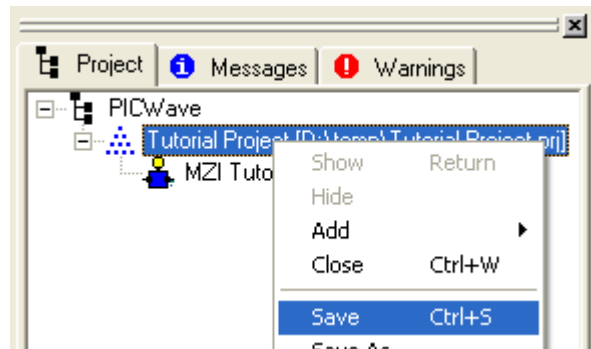



The *device window* is where we will later construct our *device*. This is done on the blue *circuit canvas*, which defines the x-z plane of the *device*: the x-direction being the vertical, the z-direction the horizontal. Here, various components from the toolbar are added and connected, creating a *circuit* which describes the *device*. (For more information on using *circuits* within a project, see §4.1-4.8)

### 2.1.1.1 Saving a project

Before we go any further, we should save the *project*.

Either select /File/Save Project in the *device window* or right-click on the  Tutorial Project in the *Project Tree* and select /Save.




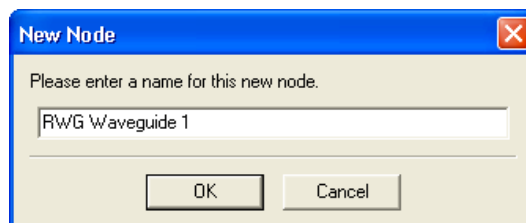
- Navigate to a convenient directory and enter **Tutorial Project** in the file selector and press . The *project* is now saved.

From here on you should save the project intermittently, when you see fit.

### 2.1.2 Adding a waveguide

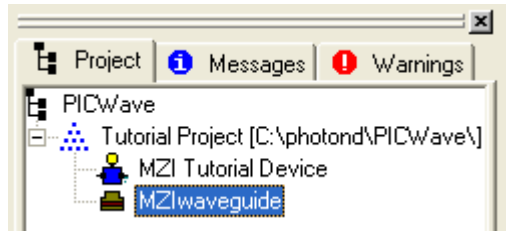
In a PICWave device, a *waveguide section* can be defined either just by its mode effective index and group indices for up to one TE and one TM mode, or by a full physical description of its cross-section in the x-y plane, which is uniform along the length of the *waveguide section* in the z-direction. The latter method allows the *waveguide section* to support multiple modes. Here we will create a waveguide cross-section, or *RWG*, which will later be used to define the cross-sections of waveguides in the MZI Tutorial device. *RWG* stands for “rectangular waveguide” – this format is convenient for describing epitaxially grown waveguides or any guide that can be described by a modest number of rectangles.


- Right click on the  Tutorial Project in the Project tree and select /Add/RWG Waveguide. The following will appear:

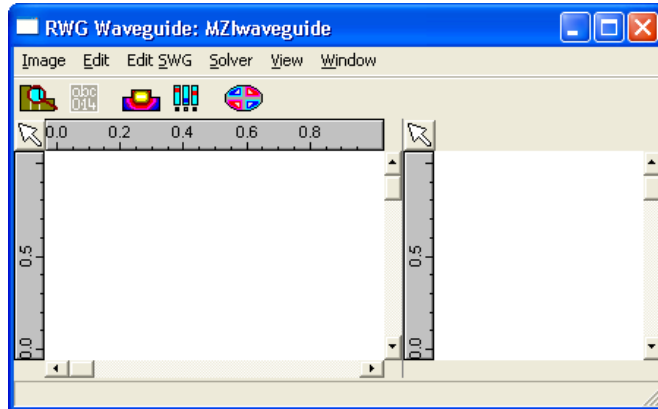


- Enter **MZIwaveguide** and click .


The waveguide icon  will appear in the Project Tree as shown:

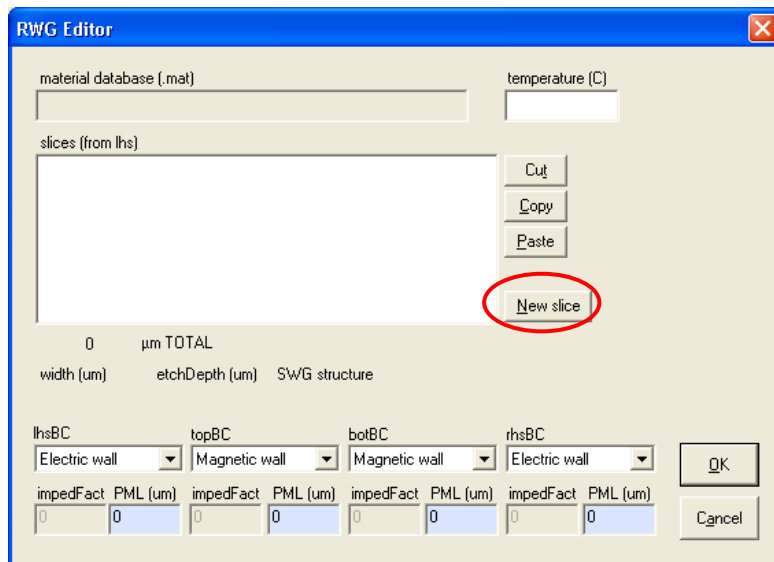


- Double-click the  MZlwaveguide line to open the *RWG Window*. The following should appear:



An *RWG* is constructed from one or more vertical *slices* arranged side by side, with each *slice* consisting of one or more horizontal *layers*. In addition, each slice can have an *etch depth*

- Click on the  button. This will open the *RWG Editor* panel.

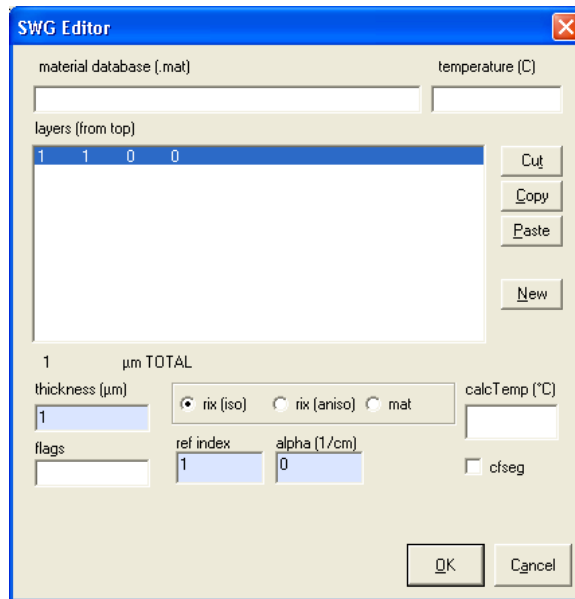


- To add a new *slice* click on the **New Slice** button (highlighted above). A new *slice* entry will appear in the slice list.

The epitaxial layer structure of a *slice* is described as an *SWG* (slab waveguide).

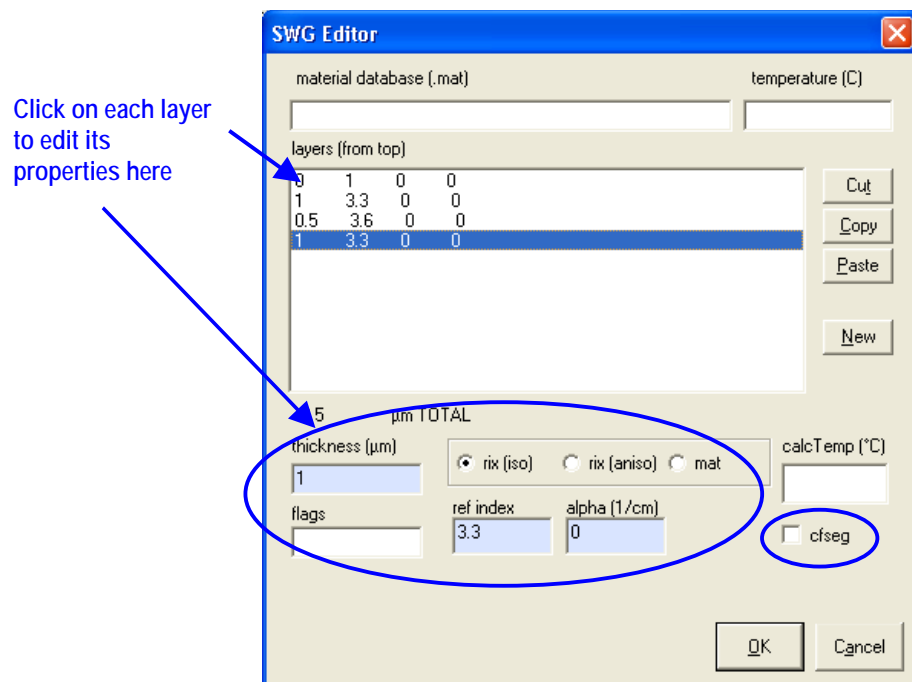
- Double click on the new *slice* entry to open the *SWG Editor*





Note that there is already one *layer* present in the *SWG*.

- Click on the **New** button 3 times to create 3 new *layers*,
- Select each of the *layers* in turn and set their **thickness** and **ref index** to what is shown here:

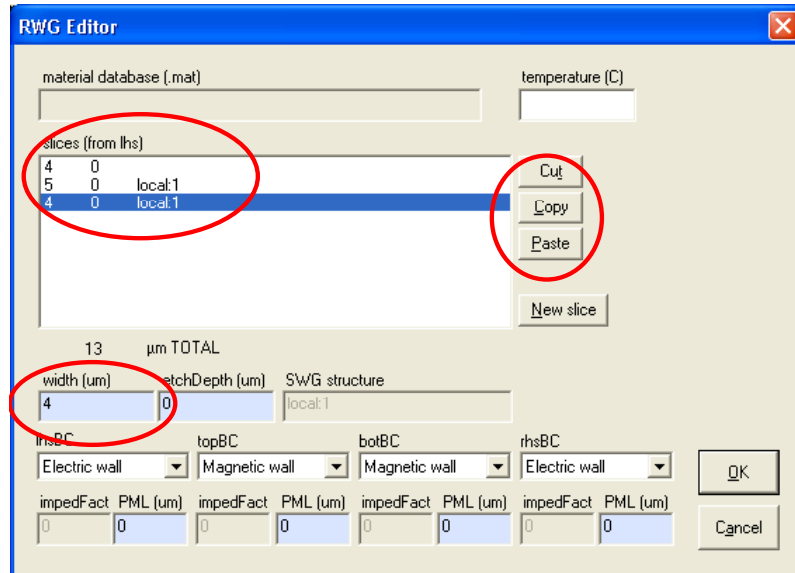


- Select the third *slice* down and then tick the **cfseg** box (highlighted above), to tell PICWAVE that it is this layer is to included in the confinement factor integral. The fourth number in the line of the third layer should change to 1, showing that **cfseg** has been ticked.

In active waveguides, ticking **cfseg** tells PICWAVE that the layer is an active layer. Although this waveguide will be used as a passive waveguide in our device, ticking the **cfseg** box will help to illustrate the calculation of the confinement factor of the waveguide later on.

We have now defined the epitaxial layer structure, or *SWG*, for this slice.

- Click **OK** when to return to the *RWG Editor*
- In the *RWG Editor*, click **Copy**, then click **Paste** twice so that you have 3 slices in total
- Set the width of slices 1, 2 and 3 to 4  $\mu\text{m}$ , 5  $\mu\text{m}$  and 4  $\mu\text{m}$ , respectively, by selecting each line in the list in turn and entering the value in **width** box as shown below.



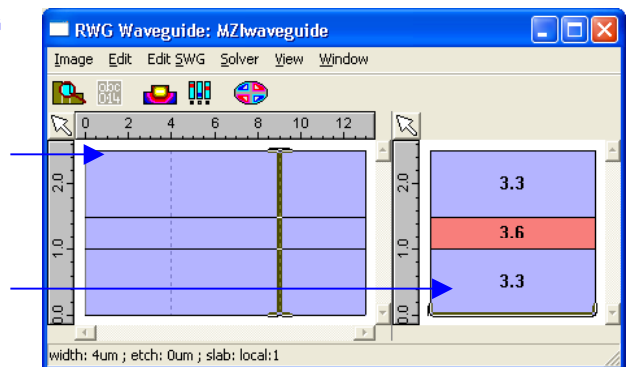
The local:1 label next to the second and third layers indicates that these layers are using the same local SWG as the first layer – this is because these layers were created by copy and pasting the first layer. Consequently, if the SWG of any of the three layers is changed, the SWG of the other two will be changed automatically.

- Click **OK** to return to the *RWG Window*

The *RWG Window* should now look like this:


**RWG view: An x-y plot of the RWG cross-section. Notice the 3 vertical slices (whose boundaries are indicated by dotted lines), all of which have the same layer structure (SWG).**

**SWG view: This shows the layers within the slice selected in the RWG view. Layers are labelled by their refractive indices; layers to be included in the confinement factor integral are coloured orange.**

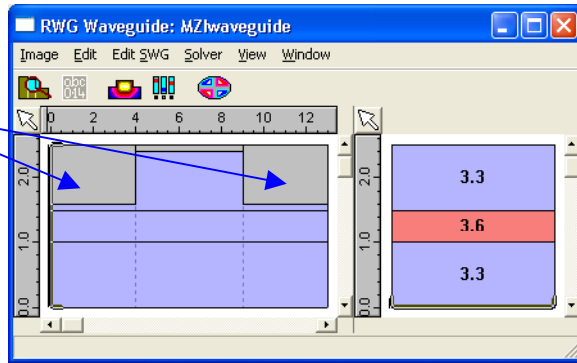


### 2.1.2.1 The Etch Mechanism

Next we need to define the ridge structure. We do this by *etching* the *slices*.

- Click on the  button again - this will show the *RWG Editor*.
- Set the **etchDepth** of *slices* 1, 2 and 3 to 0.9  $\mu\text{m}$ , 0.1  $\mu\text{m}$  and 0.9  $\mu\text{m}$ , respectively.
- Click **OK** when finished. You should now see the following.

Grey regions are slabs that have been etched.




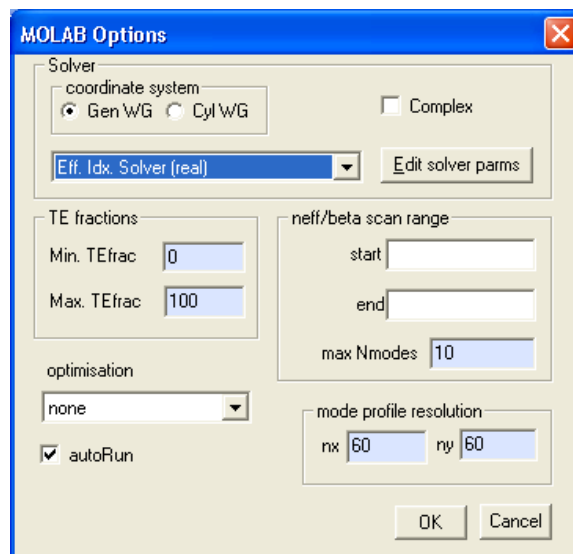
The first and third *slices* now have a grey region extending  $0.9 \mu\text{m}$  from the top surface; the second slice a region of  $0.1 \mu\text{m}$ . These indicate the *etched regions*. The refractive indices of these regions are defined by the refractive index of the top layer, which in this case is air (as the **ref index** was set to 1). Using this functionality, one can create quite complex waveguides with just one vertical *SWG* layer structure and different *etch depths* in each *slice*.


For more information on RWG waveguides and the RWG editor, see §5.1.

### 2.1.2.2 Working with waveguide modes – the Mode Finder

We will now proceed to investigate the modes of the waveguide. PICWAVE includes a simple *Effective Index Solver* and a Finite Difference Mode Solver (*FDM Solver*)

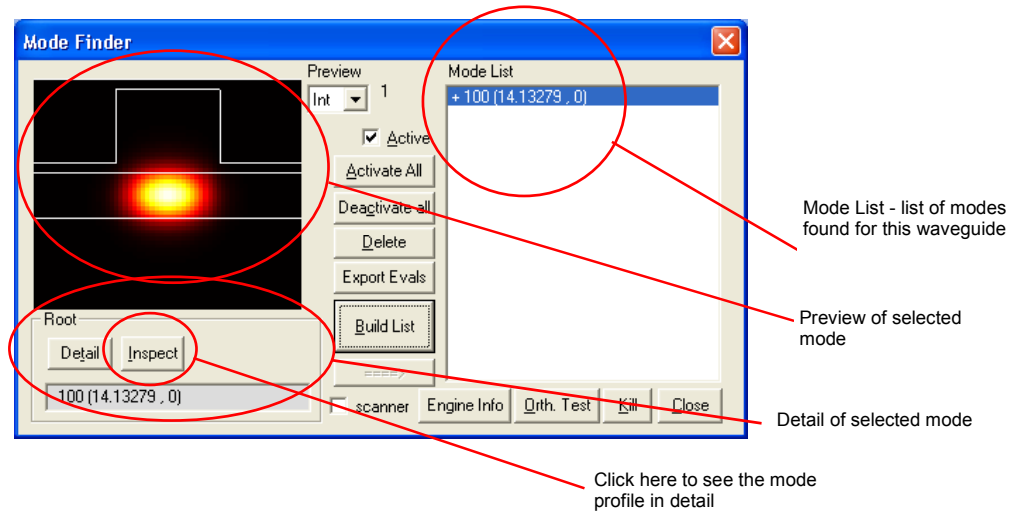
In the *RWG Window*, click on the  button. This will show the *MOLAB Options*.



- Select the *FDM Solver* from the drop-down list of solvers. Set **max Nmodes** to 1 and set **Min TEfrac** to 50. This configures the *FDM Solver* to find a single mode with a TE fraction between 50 and 100% i.e. the first TE-like mode.
- Click **Edit solver parms**. In the **Lambda** (um) box type in 1.55. This configures the *FDM Solver* to find modes at wavelength of 1.55um.
- Click **OK**.
- Click the  button, and the *Mode Finder* panel will appear.

The *Mode Finder* panel is used to locate/inspect modes of your waveguide, and to plot or print the mode profiles. Initially you will have no idea of the eigenmodes(s) of your waveguide. The *Mode Finder* works in a fully automatic way and will locate one or more modes already according to the criteria set in the *MOLAB Options*.

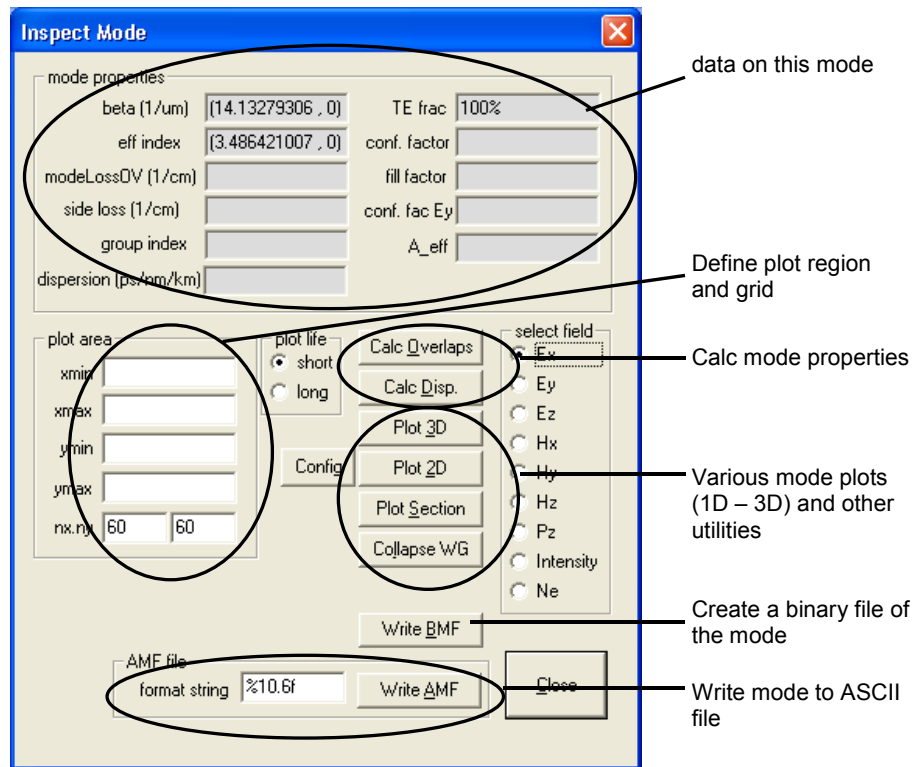
- Click **Build List**. This will open the *Build EV List Panel*.
- In the *Build EV List Panel*, click **Start** to start finding modes. When finished the *Mode Finder* panel should look as follows:



The *Mode List* on the right hand side of the *Mode Finder* panel lists *all* the modes that the *Mode Finder* has found for the RWG. In a given device, only these modes can be supported by a waveguide section that uses this particular RWG as its cross-section definition. Direct access to the *MOLAB Options* is conveniently provided in the device window, as you will see below. To inspect any of the listed modes, double-click on its list entry or select it and click **Inspect**.

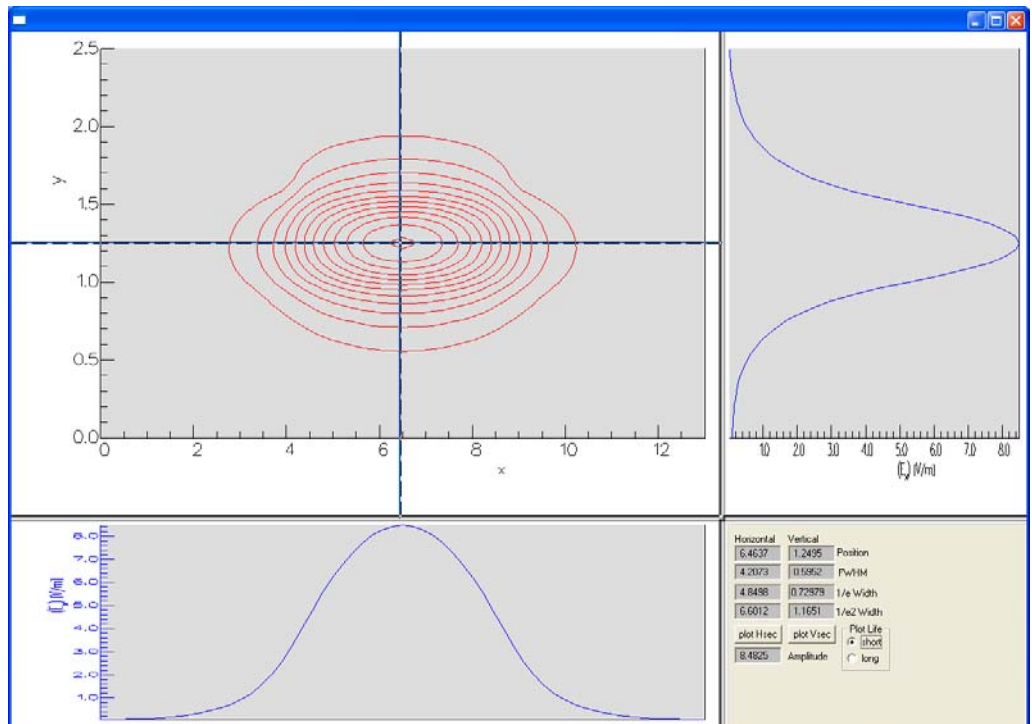
Given our *MOLAB Options* criteria, you should see one entry in the *Mode List* box. Double-click it and the *Inspect Mode Panel* should appear.

### 2.1.2.3 The Inspect Mode Panel



Among other things, you may use this panel to do any of the following:

- plot a 2D/3D profile of the mode - any field component or the mode intensity.
  - plot a 1D cross-section of the mode at a given horizontal or vertical position.
  - produce an ASCII file of either of the above
  - output 1D, 2D and 3D profiles to your printer.
  - calculate the confinement factor and/or loss of your mode.
- Select the  $E_x$  field, click **Config** next to **Plot 2D** and ensure that *Contour Plot* is selected from the **Plot Type** drop-down list, then click **OK**.
  - Click **Plot 2D** - a contour plot should appear, indicating that this is the zero order TE mode - if instead you plot  $E_y$ , you will see that there is field only at the sides of the ridge.
  - Click **Calc Overlaps** - after a short delay, some data will appear in the region titled *mode properties*. This will indicate a *confinement factor* of 0.84 - remember the **cfseg** flag you set in the vertical layer structure (*SWG*). It will also show *TE frac* = 100%, this has been calculated by analysing the whole mode profile and is an accurate value. It also calculates the *mode loss* - zero in this case. This is an overlap integral between the mode and the loss profile and is a reasonably accurate estimate for low loss waveguides.
  - Select the  $E_x$  field, click on **Plot Section**.
  - Move the cursor to the centre of the mode and left-click. 1-D cross-sections of the  $E_x$  field will be plotted for the horizontal and vertical planes indicated by the dotted lines passing through the cursor, as follows





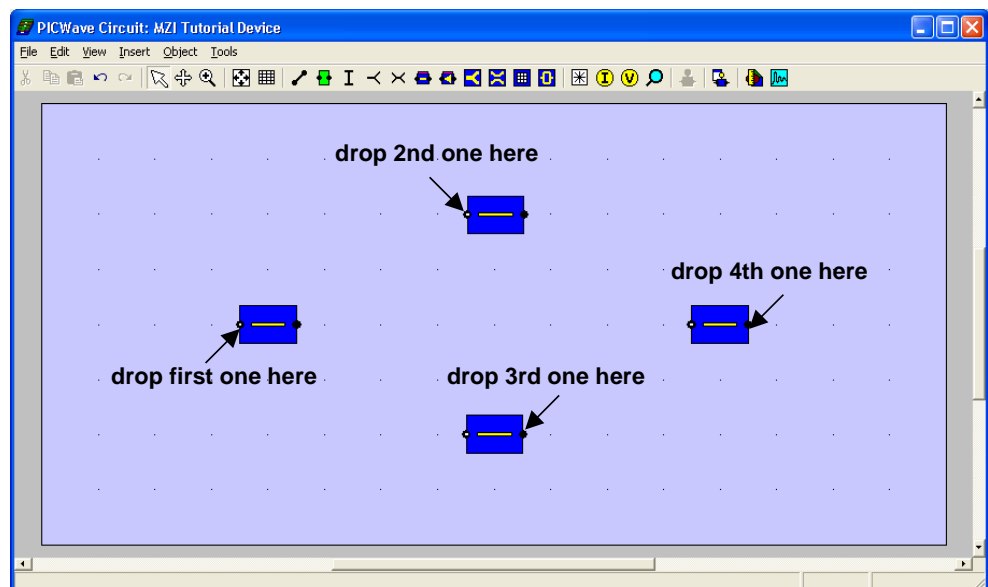
- Close the *Inspect Mode Panel*.

For more information on the mode solvers, *Mode Finder* and *Inspect Mode Panels*, see §5.5 ff.

### 2.1.3 Constructing the MZI Device

We are now going to construct the MZI Device.

- Open the  *MZI Tutorial Device* by double-clicking on its entry in the project tree.
- Click on the  icon in the *device window* for the *MZI Tutorial Device*. Now click on the *device window*, somewhere towards the left side to drop a *waveguide section* onto the device. Drop three more *waveguide sections* until you have a device view like:




- Right-click on the left most *waveguide section* and select *IProperties*. You will see the *Waveguide Section Properties* panel:


Waveguide Section Properties		
Parameter	Value	Units
sectionName	WGSection-1	
crossSectionDef	effIndex	
sectionType	passive	
modePol	TE-like	
length	10	μm
Tsub	25	Celcius
<b>Active properties</b>		
sponBetaNA	0.001	
numContacts	0	
CurrentFlowModel	more...	
ElectroAbsorptionModel	more...	
ncellW	0	
instruments	more...	
<b>Grating properties</b>		
gratPhaseShift	0	deg
CouplingParms	more...	
<b>TE properties</b>		
numTEModes	1	
effIndexTE	1	
effGroupIndexTE	1	
effLossTE	0	[1/cm]
<b>TM properties</b>		
numTMModes	0	
effIndexTM	1	
effGroupIndexTM	1	
effLossTM	0	[1/cm]

Notice the parameter **crossSectionDef**. This tells PICWAVE how to define the cross-section of the waveguide. As discussed in §2.1.2, we could use a reference to an *RWG* such as we created above, by setting **crossSectionDef** to *waveguide*. Alternatively, for simple things, we can set it to *effIndex* and then simply enter the effective index and group index of the zero-order mode of the waveguide in the boxes provided. We will use this latter approach for this section. So:

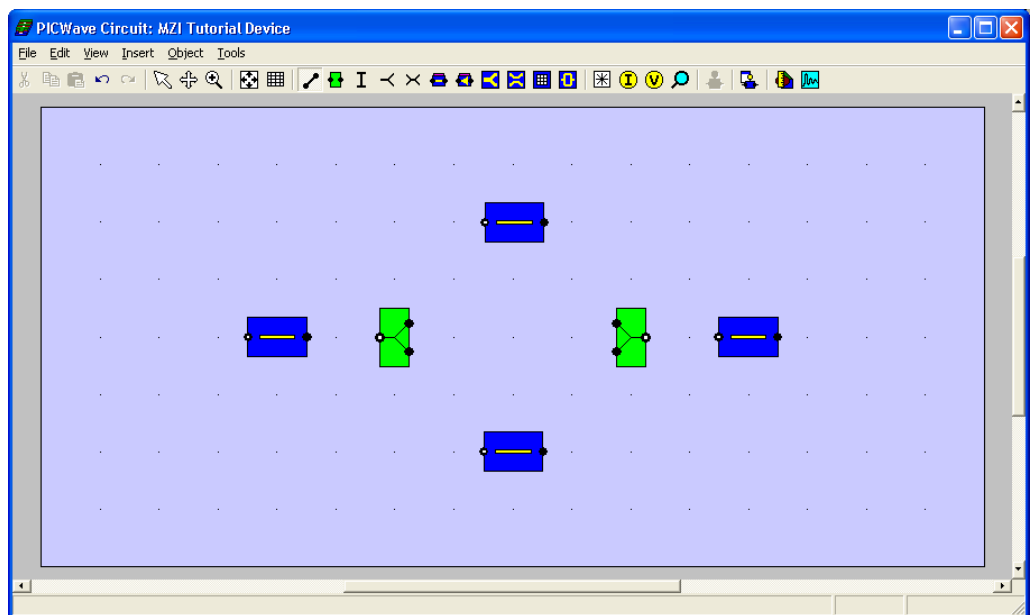
- Set **crossSectionDef** to *effIndex* (default). Notice that this setting requires the *waveguide section* to be passive (i.e. **sectionType** is *passive* and greyed out), as no active layers can be specified in this type of cross-section definition.
- Set the waveguide **length** (in the z-direction) to 100 μm.
- Set **modePol** to TE-like. This means that the section will support one TE mode.
- Then set **effIndexTE** to 1.5 and **effGroupIndexTE** to 1.5.
- Repeat the same changes for the top section.
- Repeat the same changes for the bottom section except set the length to 200um.

Let's use the *RWG* we created for the output *waveguide section*, so set the properties of the output (right-most) section as follows:

- Set the **crossSectionDef** of the right section to *waveguide*.
- Set the **sectionType** property to *passive* (default). Notice that, because our *RWG* has an active layer, the choice between *passive* and *gain* (and *ea-modulator*) is now available. It is also important to note that, when using an *RWG* to define the cross-section of a *waveguide section*, the mode supported by the section are determined by the choice made in the *MOLAB Options* for the *RWG* (§2.1.2.2).
- Set **length** to 100um.
- Find the  M2Iwaveguide object in the project tree and right-click /Copy to copy it to the clipboard.

- Back in the device window, right-click on the output *waveguide section* and select */Paste WG Ref.* Now the section will use the *RWG* cross-section description we defined above. To check this you can right-click on the output section and select */Edit WG* or */MOLAB Options*. The *RWG Window* and *MOLAB Options* for the *MZI* waveguide should appear in each case, respectively.
- Click on the  icon to add a *power splitter* and drop it to the right of the left *waveguide section*. This component will split a signal entering the left side into two outputs on the right side. It can split the power equally or unequally; it can reflect a part of the light; it can be lossless or lossy. Alternatively you can define a complete general scattering matrix for the component. Thus you can do simple things quickly but can also specify a completely arbitrary 3-port device. In our example we just want a lossless 3dB splitter. Since this is the default setting for the *power splitter* we have nothing more to do.
- Add another *Power Splitter* to the left of the right section. Right-click on it and select */Rotate/180 degrees* so that it has the correct handedness and will join two signals on the left side into one signal on the right side.
- Press Escape to stop dropping things on the device.

You should now have the following layout:



### Sections, Joins and Ports

The layout we have just created consists of blue and green components. All blue components represent *sections* – objects with a physical size, such as a length of waveguide (a *waveguide section*). All green components represent *joins* – elements that have no physical shape or size but rather allow us to define the coupling and splitting of an optical signal in an abstract manner (e.g. a *power splitter*).

You will notice that each component has one or more black dots on its end faces. The black dots on each component represent *ports*. A *port* is where light enters or exits the component. Typically each *port* corresponds to a waveguide entering or exiting the component. One port on each *section/join* is marked with a white dot; this is helpful in determining the relative orientations (after rotations) and also for identifying the correct *ports* when specifying coupling coefficients etc.

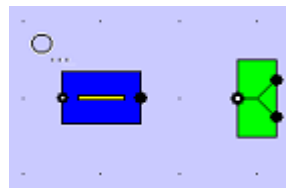


For further information on the *sections* and *joins* available in PICWAVE, see §4.9 and 4.10.

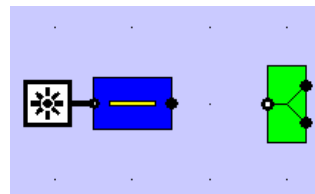
### Optical Sources

PICWAVE can inject an *optical signal* into any *port* of the device (except ports on a *general join*– see §4.10.4) You can independently define both the wavelength profile (time dependence) and the intensity profile. You can define signals that contain multiple wavelengths and for doing spectral analysis on a device you can define a delta function pulse – which has a wide band flat optical spectrum.

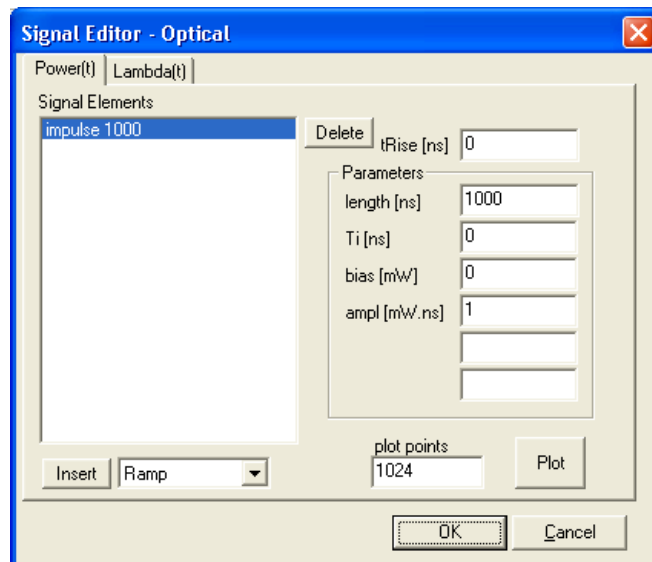
- Click on the  icon to add an *optical source* and move the cursor to the left of the left section:



- Now place the cursor circle over the left port of the section. When you have the correct position, the cursor will become white. Now click your mouse. The source should connect to the port:




- Right click on the source to edit the *Optical Source Properties* panel. Double click on **signal**. The *Signal Editor* will appear:





We want to do a wide band simulation so we need a short pulse that will contain a wide spectrum of optical frequencies (or wavelengths if you prefer). By default it has already inserted an optical impulse in the signal. However for instruction purposes this is what you would need to do to add it yourself:


- Select *Impulse* from the drop-down list next to the  button.

- Click 
- Set **length** to 1000 ns
- Set **ampl** to 1 mW.ns
- Press Plot to view your signal.

We have now defined the time dependence of the optical power. We still need to define the time dependence of the wavelength - again we can use the default settings, which would be added manually as follows:

- Click on the Lambda(t) tab.
- Select *Ramp* from the drop-down list next to the  button.
- Click 
- Set **length** to 1000 ns


The wavelength is set to be a constant zero nm (default) offset from the central wavelength of our calculation (to be defined later).

- Click  to close the *Signal Editor*.

For further information on adding an *optical source* and using the *Signal Editor*, see §4.12 and §4.16.

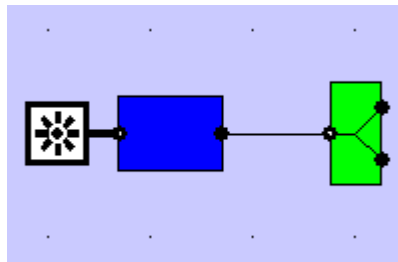
#### Connecting Up

Next we need to connect the different components of the *circuit* to define the optical paths between them.

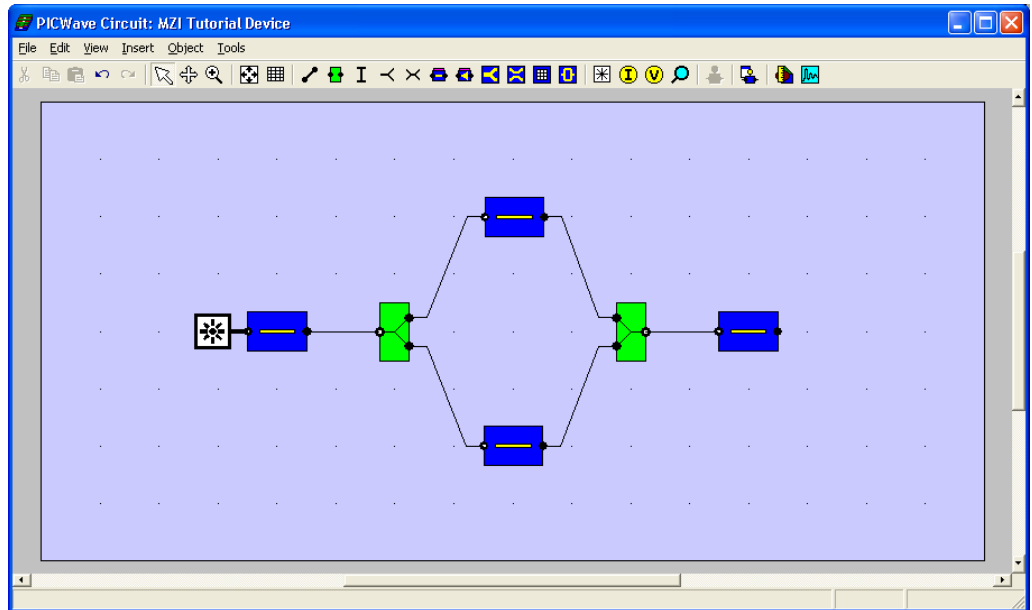
- Click the  icon to activate the *link* tool. The cursor will display:



- Move the cursor circle until it is on top of the right-hand *port* of the left section. Click your left mouse button. It should now display 2 instead of 1 indicating that it is ready to be moved to the second *port*.
- Move the cursor circle until it is on top of the left-hand *port* of the left power splitter. Click your left mouse button. You should now see:




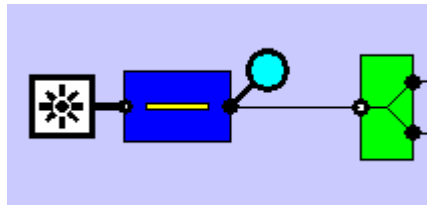
- Now connect up the rest of the *circuit* as shown in the figure below, pressing **Esc** when you have finished:



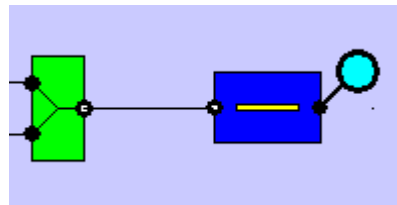
#### 2.1.4 Adding Monitors

We need to add some *monitors* to record the signals going into and out of our device. *Monitors* can be used to measure the instantaneous optical power or instantaneous wavelength. In addition they can record the time-evolving fields during the simulation so that you can do spectral analysis later.

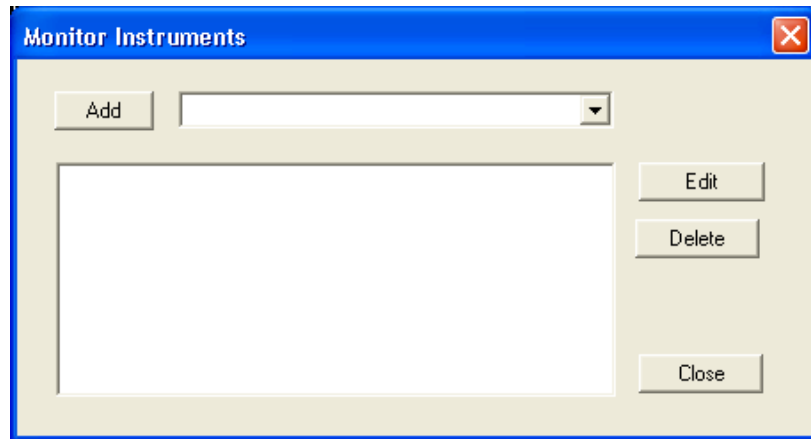
Click on the  icon in the toolbar. Now click on the right hand port of the input waveguide. You should now have a *monitor* attached to the input waveguide as shown:



Add another *monitor* to the output waveguide so that you have:



- Right-click on the input waveguide *monitor*, select */Properties...* to open the *Monitor Properties* panel, and set the **monitorName** to name to *Monitor-In*.
- Double-click on the **instruments** line to display the *Monitor Instruments* panel.




- Select *Facet Power* from the drop-down list and click **Add**. A *Facet Power Instrument* will allow us to measure the time-evolving power at the monitor. If you double-click on the new instrument, you will see a few settings we can adjust, including: **inOrOut**, to measure the power going *into* or *out* of the port; the **responseTime**; and **dlambda**, which together with a finite response time, causes the instrument to detect only a particular band of wavelengths. There is also the polarisation and index of the mode whose power is to be measured. We don't need to change any of the defaults however. Therefore click on **Close**.
- In the *Monitor Properties* panel, set **enableSpecMeas** to *Out*. This causes the monitor to record the field of the *outgoing* signal at every time step so that spectral measurements can be made once the simulation has completed. Close the *Monitor Properties* panel.
- Do the same procedure to set up the monitor on the output waveguide but this time name it *Monitor-Out*.

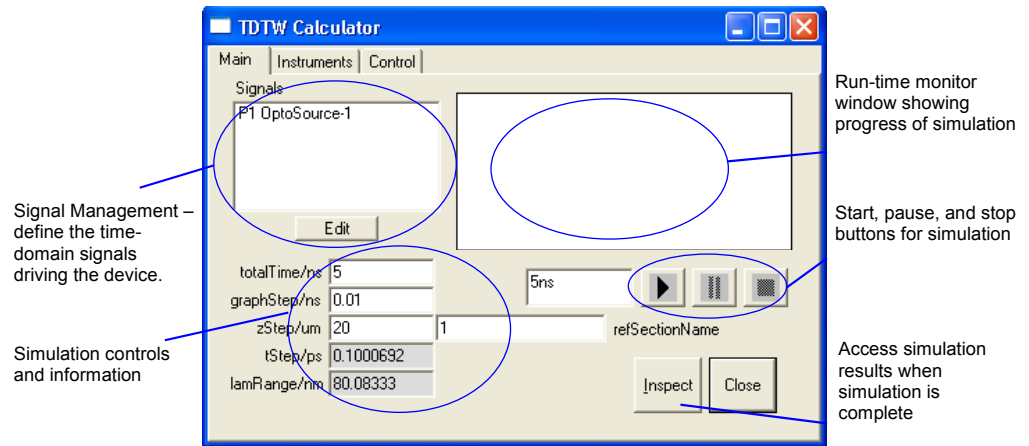
For further information on *monitors* and their associated instruments, see §4.14 and §4.15.

NOTE: Before proceeding to run the simulation, we note here that the mode/polarisation settings must be consistent throughout the whole circuit, that is, in every section, source and instrument. PICWAVE will warn you if this has not been done correctly when you try to run a simulation. The simulation will not run until the necessary corrections have been made. In this tutorial device we have confined ourselves to the first TE mode (the default). (See §4.18 for further details on consistency in mode/polarisation settings).

### 2.1.5 Running the TDTW Calculator

We have now completed setting up our device and are ready to set up the time-domain calculator.

- Click on the  icon in the tool-bar. The *TDTW Calculator* window should appear.



On the Main panel, note the following:

### Signals

This shows a list of all the signals driving the device, including both optical and electrical signals. We have no electrical signals and only one optical signal – the impulse, in our example – which is shown in the list.

### zStep and tStep

The **zStep** parameter is an important calculation parameter. It can be defined for only one section of the device, the *reference section* - whose number or name is specified in **refSectionName**. A value of  $10\mu\text{m}$  is generally fine for a section containing active materials. The simulation time step, denoted in PICWAVE by the dependent parameter **tStep**, is calculated from the zStep and group velocity of the *reference section*, **zStep** and  $v_{g,\text{ref}}$ , by the relation  $\mathbf{tStep} = \mathbf{zStep} / v_{g,\text{ref}}$ . Thus a signal propagating through the *reference section* will travel **zStep**  $\mu\text{m}$  in **tStep** ps. A corresponding relation holds for all the other sections in the device; their zSteps, which may be different from that of the *reference section*, are computed by  $\mathbf{zStep}_j = v_{g,j} \times \mathbf{tStep}$ , where  $j$  denotes the  $j$ th section. ( $v_{g,j}$  is either defined by **effGroupIndex** in the waveguide section's properties or, for waveguides with cross-sections defined by RWGs, is determined by the layer group indices and the mode profile(s)).

[NOTE: The length of the  $j$ th section must be an integer multiple  $\mathbf{zStep}_j$ ; and PICWAVE will adjust  $\mathbf{zStep}_j$  accordingly. This can introduce a discretisation error on the group velocity,  $v_{g,j}$ . PICWAVE will not allow this to exceed 10%. You should bear this in mind when choosing **zStep** – the smaller it is the better the program can choose the zSteps of the other sections i.e. the smaller the discretisation errors. (See §11.2)]


**tStep** is displayed in the *TDTW Calculator* window after the simulation has started, when it is paused or stopped (see below). A **zStep** of  $10\mu\text{m}$  corresponds approximately to a **tStep** of 0.1 ps, in GaAs, which is adequate for most calculations.





### Free spectral range and resolution

The free spectral range of the simulation is given by  $1/\mathbf{tStep}$ , so to get a simulation with a large free spectral range you need a small **zStep**. On the other hand, the resolution of any spectra you generate is given by  $1/\mathbf{totalTime}$ , so if you decrease **tStep** then you will need more time steps to get the same spectral resolution. If you want higher spatial resolution or a higher free spectral range calculation, then you will need to decrease


**zStep.** The free spectral range, expressed as a wavelength range, is computed and displayed in the *DTW Calculator* window as the read-only parameter **lamRange**. **lamRange** is given by  $\text{lamRange} = \lambda \cdot \text{c.tStep}$ , where  $c$  is the speed of light in a vacuum and **lambdaCentre** is the central simulation wavelength (see below).

The Instrument panel shows a list of *all* the *instruments* that we have defined – in each monitor. This is a convenient second way of editing the instruments, although you cannot add or remove instruments from here. To edit an instrument, double click on its entry in the list.

- Double-click on the *Monitor-Out-FacetPowerOUT* instrument and set **plotInMonitor** to *True*. This will cause the instrument’s measurement to be displayed in the Run-time Monitor Window of the *DTW Calculator*.
- Click on the Control panel and set **lambdaCentre** to 1.55 μm. The other parameters on the Control panel are concerned with active simulations so we ignore them here.
- Return to the Main panel. Set **zStep** to 20 μm. [leave **refSectionName** set to 1, so that the input waveguide is used as the *reference section*]
- Set **totalTime** to 5ns.
- Click the  button. The simulation will begin. You should see the *Monitor-Out-FacetPowerOUT* instrument’s measurement displayed in the *Run-time Monitor Window*.

At any time while the simulation is running, you can: click the  button to examine the progress of the calculation in detail (see below); stop (abort) a calculation by clicking on the Run/Kill button  button in the top-right of the Main Window, or the  button on the *DTW Calculator* window; or pause the simulation by clicking the  button, and resume by clicking it again.

### 2.1.6 Inspecting Results

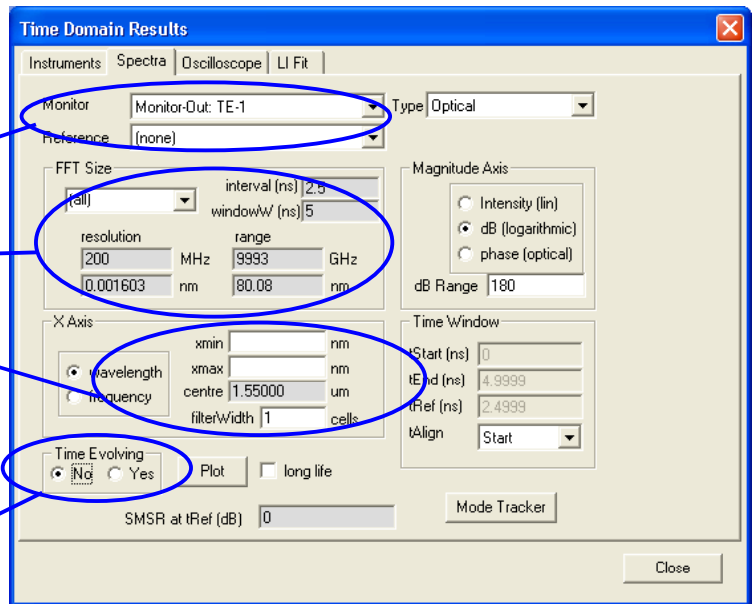
- When the simulation is finished, click , the *Time Domain Results* window should open with the *Instruments Results* panel selected.
- From the *Instruments Results* panel you can view the instrument readings. Double-click on “Monitor-Out-FacetPowerOUT” – you will see the impulse exiting the device with a small time-delay.
- Click on the *Spectra Results* panel. From here you can do various spectral analyses on the simulation results, such as generating optical spectra – both static and time-evolving, and intensity spectra – useful for generating RIN spectra of laser diodes.

Choose which monitor result you want to study

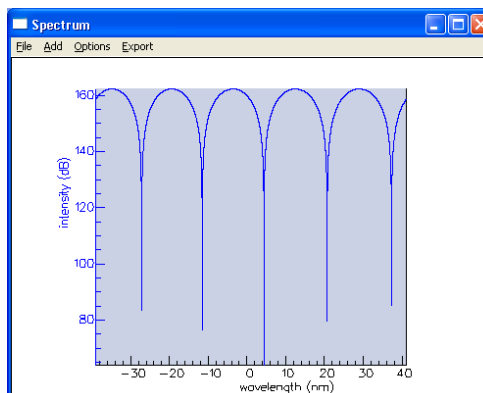
Configure the FFT

Configure the plot

Choose single or time-evolving spectrum



- Set **Type** (top-right) to *Optical*.
- Set **Monitor** (top-left) to *Monitor-Out: TE-1*
- Set **Time Evolving** (bottom-left) to *No*.
- Click the *wavelength* button in the **X-Axis** box so that the spectrum shows intensity against wavelength.
- Click **Plot**. The spectrum of the outgoing signal at the output waveguide plotted on a logarithmic scale should appear as shown below:



- Back in the *Spectra Results* panel of the *Time Domain Results* window, click on the **Intensity** option under **Magnitude Axis** box.
- Set **Reference** to *Monitor-In: TE-1*. This will cause the output spectrum to be divided by the input spectrum, when plotted.
- Click **Plot** again. The plot corresponds to transmission as a function of wavelength, on a linear scale.

Notice that the previous plot has disappeared. If you want to keep a plot, click on the **longLife** option next to the **Plot** button before you plot.

For further information on the *TDTW Calculator* and the *Time Domain Results* window, see §9.



**Congratulations you have completed your first PICWAVE simulation!**

## 2.2 Simulating a Fabry-Perot Laser

If you have the Active Module, you are able to simulate all sorts of active devices including SOAs and laser diodes, even tuneable lasers consisting of many different sections. In this section we will simulate a simple Fabry-Perot laser, generating its LI-curve and optical spectrum, and plotting an eye-diagram.

### 2.2.1 Creating an active waveguide

First we need to create a waveguide cross-section for our Fabry-Perot laser. You should run through §2.1.2 above before following this section, since that section goes into more detail of these procedures.

- Right-click on the project tree and add a new RWG waveguide. Name it *FPLaserWaveguide*.
- Open the waveguide's *RWG Window* by double-clicking on its entry in the project tree.
- Click on the  button to show the *MOLAB Options*. Select *FDM Solver* (default), set **max Nmodes** to 1.
- Click **Edit solver parms** and set **lambda** to 1.55 um and click **OK**.
- Back in the *RWG Window*, select */Edit/Set Material database (.mat)*. [If you have not done so already copy the standard material database supplied on your CD, rebase-picwave.mat into the directory where you've saved the Tutorial Project.prj]
- Select the rebase-picwave.mat *Material Database* file and click **Open**.
- Now click on the  button to open the *RWG Editor*.
- Click on **New slice**
- Select the new slice, then copy and paste two more slices
- Set the width and etch depths of slices according to the following table:

Slice	Width (um)	Etch depth (um)
1	2	0.68
2	2.5	0.1
3	2	0.68

- Next we need to define the layers of the waveguide. Double-click on one of the slices in the slice list. This will bring up the *SWG Editor* from where we can edit the layer structure (shared by all three slices).
- The *SWG* should already have one layer in it. Set the width of this layer to 0 um.
- Click on the **mat** radio-button:





This allows you to associate a material with the layer rather than giving it an explicit refractive index. The properties of the material are obtained from the material database file (refbase\_picwave.mat) file that we selected earlier.

- From the drop-down list that has appeared, select SiO2.
- Create 5 more layers using the **New** button.
- Set the thickness and material of each of the layers as shown below:

layers (from top)		
0	SiO2(0.000000)	0
0.7	InP(0.000000)	0
0.3	InGaAsP(0.400000)	0
0.05	InGaAsP(0.600000)	1
0.3	InGaAsP(0.400000)	0
1	InP(0.000000)	0



Certain materials have a composition parameter that can be set when you select the layer. You will see that the InGaAsP materials have a fraction 0.4, 0.6 etc. This is the As fraction (**As-frac**).. (After clicking the **.mat** button you may have to click on the layer again before the parameter box becomes enables.)

- Set the As fractions as shown.
- Select the InGaAsP(0.6) layer. This will be the quantum well. To tell PICWAVE that it is active, click the **cfseg** box.

We now have a waveguide cross-section describing our quantum-well structure.

### 2.2.2 Build the FP Laser Device

Next we will create a simple PICWAVE circuit representing our Fabry Perot laser device.

- Create a new device in the project by selecting the project in the project tree and clicking on the  Add PICWave Circuit button.
- Name it **FPlaser**.
- Double-click on the FPlaser entry in the project tree to display the new device.
- Click on the  icon in the FPlaser device window. Now click somewhere in the middle of the *circuit canvas* to drop a *waveguide section* onto the device.

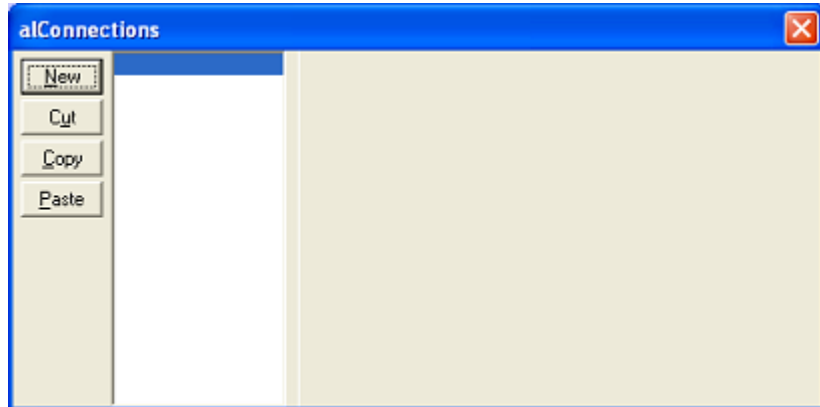
Right-click on the new *waveguide section* and select */Properties*. Set the *Waveguide Section Properties* as follows:

Parameter Name	Value	Notes
<b>sectionName</b>	<i>WGSection-1</i>	The section is given a name by default – you do not need to change it.
<b>crossSectionDef</b>	<i>waveguide</i>	We will use the RWG waveguide we generated earlier for this section
<b>sectionType</b>	<i>gain</i>	
<b>length</b>	<i>160 um</i>	
<b>numContacts</b>	<i>1</i>	Number of electrical contacts
<b>ncellIW</b>	<i>9</i>	This defines the lateral discretisation of the active layer(s) – see §11.2
<b>Tsub</b>	<i>25</i>	Substrate temperature

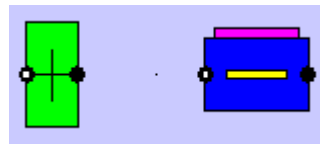
- Now right click on FPLaser RWG on the Project Tree and select /Copy. Then right-click on the *waveguide section* in the FPLaser device and select /Paste WG Ref.

Next we need to create a connection between the electrical contact and the active layer.

- In the *Waveguide Section Properties* panel for the *waveguide section*, double-click on the **CurrentFlowModel** property. The *CurrentFlowModel* panel will appear. Presently the model is very simple consisting just of a resistance between the contact and each active layer. (For more information on the *Current Flow Model*, see §11.1)
- Double-click on the **alConnections** property. The following panel will appear:



- Click **New** and set the **resistivity** of the new connection to 100 ohm.μm<sup>2</sup>.
- Close the property panels and return to the *device window*.
- Select the *facet* icon **I** from the device window. Click once to the left of the *waveguide section*. You should now see the following layout:




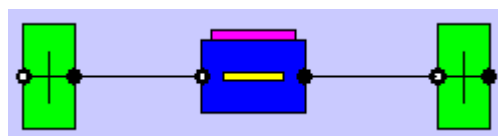
- Right-click on the new *facet*, select /Properties... and set **reflCoeffTE** to 0.3. This creates a facet with a 30% power reflection coefficient.

By default the facet is set to *Lossless* so that the TE transmission coefficient is automatically set to 70% for you.

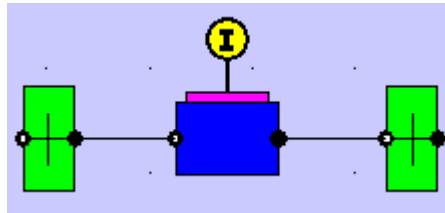
- Make sure the new *facet* is selected and type **Ctrl-C**, then **Ctrl-V** to create a copy.
- Move the copy to the right of the waveguide so you have:



- Click the  icon to activate the *link* tool. Use this to join up the *facets* to the *waveguide section*. When you are finished you should have:

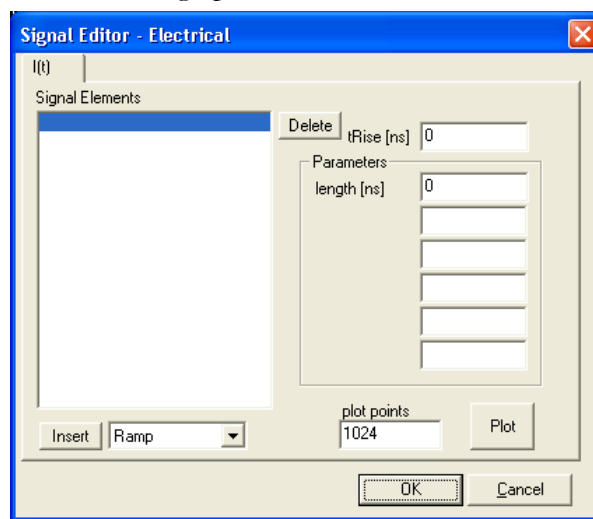


- Next we need to add a *current drive* to drive the laser. Click on the **I** icon. The cursor will change to a down arrow. Click on the contact on top of the *waveguide section*. You should now see:



This attached a *current drive*, from where you can inject a current signal of arbitrary time-evolving form.

- Right-click on the *current drive* and select */Properties...* To open the *Current Drive Properties* panel.
- Double click on **signal** to bring up the *Signal Editor*.



- Make sure element type is set to *Ramp* and click **Insert**.
- Set **length** to 10 ns, **startVal** to 0, **endVal** to 100 mA.
- Click **OK** to close the *Signal Editor*.


As can be seen from the *Current Drive Properties* panel, the current drive has two instruments automatically added to it, one to measure the current (**currInstrument**) and the other to measure the voltage of the drive signal (**voltInstrument**). These instruments can be inspected later when the simulation is running.

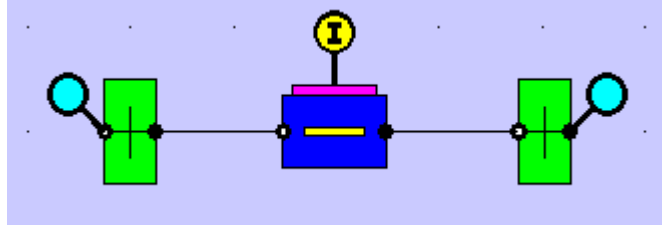
- Double click on the **currInstrument** property. The *Current Instrument Properties* panel should appear:

Parameter	Value	Units
enable	False	
valueName		
plotInMonitor	False	

- Set the **enable** property to *True* so that the current provided by the *current drive* can be inspected during and after the simulation.
- Close all the property panels.

### 2.2.3 Adding some Optical Monitors

- Click on the  icon in the toolbar. Now click on the left-hand port of the left facet and the right port of the right facet. You should now have two *monitors* attached to the device as shown:



- Open the *Monitor Properties* panel of the left hand monitor. Change its name to *Monitor-LH*.
- Set **enableSpecMeas** to *Out*, so that the spectrum of the outgoing signal will be measured.
- Add a *Facet Power Instrument* to the monitor. (Double-click the **instruments** property, select *Facet Power* from the drop-down list and click the **Add** button).
- Repeat the above for the right hand monitor, but set its name to *Monitor-RH*.

### 2.2.4 Adding some Active Layer Instruments


To monitor the internals of the laser such as the carrier density, temperature or gain we can add some *Junction Instruments*. When developing a laser model it is particularly useful to see the carrier density.

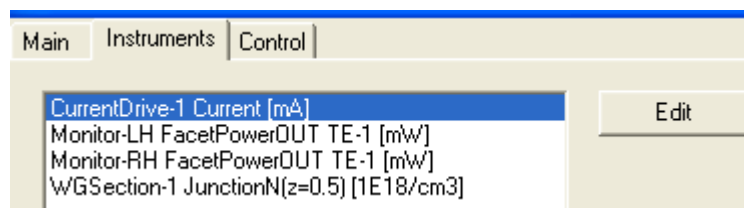
- From the *Waveguide Section Properties* panel of the *waveguide section*, double click on the **instruments** property. The *Waveguide Section Instruments* panel will appear. From this panel you will see a drop-down list of all the instruments you can add.
- Select Junction Ne and then click **Add**.
- Double click on the new instrument in the list and set **zFraction** to 0.5, this means that it will measure the carrier density half way along the section.
- Close all the property panels.

For more information on the *Junction Instruments*, see §4.15.

### 2.2.5 Running the TDTW Calculator for a laser simulation

We are now ready to set up the *TDTW Calculator*


- Click on the  icon in the tool-bar. The *TDTW Calculator* panel will appear.
- From the Main panel, set **totalTime** to 10 ns, **zStep** to 16  $\mu\text{m}$ .
- Go to the Instruments panel. You should see 4 instruments listed:





Recall that we added all these to the device at various stages above. They are all listed here together for convenience. You can also edit their properties from here.


- Double click on the *WGSection-1 JunctionN* instrument and set **plotInMonitor** to *True*, so the carrier density will be plotted in the *Run-time Monitor Window* of the *TDTW Calculator* when the simulation is running.
- Do the same for the *Monitor-RH FacetPowerOUT* instrument.
- Click on the Control panel and set **lambdaCentre** to 1.55  $\mu\text{m}$ .

The simulation is now ready to run.

- Click the  button. The simulation will begin. You should see the carrier density and facet power instruments' measurements displayed in the *Run-time Monitor Window*.

At any time while the simulation is running, you can click the  button and temporarily interrupt it.

- When the simulation has finished click the  button. The *Time Domain Results* window will appear.
- In the *Instruments Results* panel, double-click any of the instruments listed to plot their results.


- Select the *CurrentDrive-1* instrument and press the  button. This will set the current drive's current as the x-axis variable.

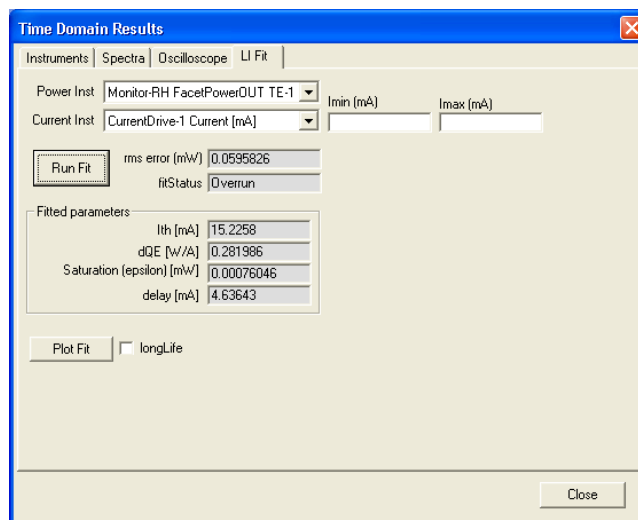
- Now double-click on the *Monitor-RH-FacetPowerOUT* instrument. You will see that you now have a graph of power versus current.

Next we will measure the *threshold current* (**lth [mA]**) and *differential quantum efficiency* (**dQE [W/A]**).

- Click on the *LI Fit* panel.

From here we can do an automatic fit to the LI curve to measure the **lth** etc.

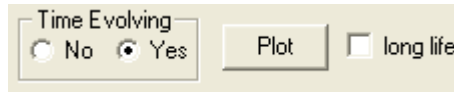
- Set the **Power Inst** and **Current Inst** choices to *Monitor-RH FacetPowerOut TE-1 [mW]* and *CurrentDrive-1 Current [mA]*, respectively.
- Click . You should see the **lth** and **dQE** displayed in the boxes, as follows:



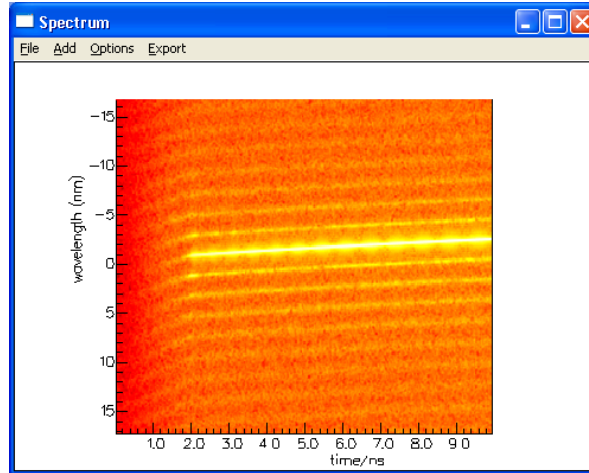
For more information on the *LI Fit* routine, see §9.8

Next we will take a look at the spectrum of the laser.

- Click on the *Spectra Results* panel.
- Select **Time Evolving** to *Yes* at the bottom left.

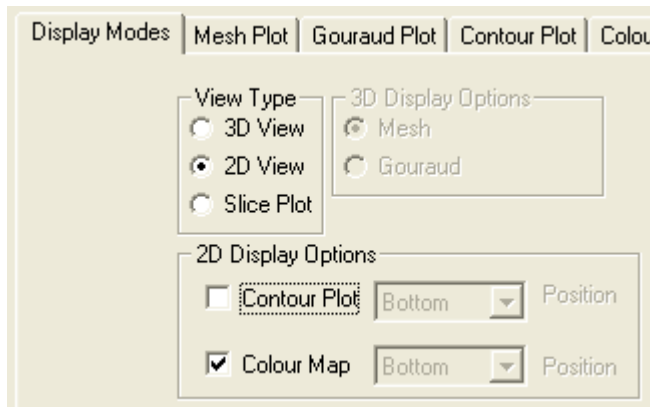


- Click **Plot**. You should see one of the following:



Here, one can see the shift in the cavity modes arising from chirp, which is parameterised by the linewidth enhancement factor in the material database (see LEF in **Table 10-1**.in §10-3). For more information on plotting spectra, see §9-7)

- [Note: If the graph does not appear to be the same type as that above, then
- Right-click on it and select */Display Options*. On the *Display Modes* tab, select *2D View* and *Colour Map* as shown:



- Then click **Set as Default** so that the plot always appears in this form in future. Click **OK** and you should have the Colour Map shown above on the left.]

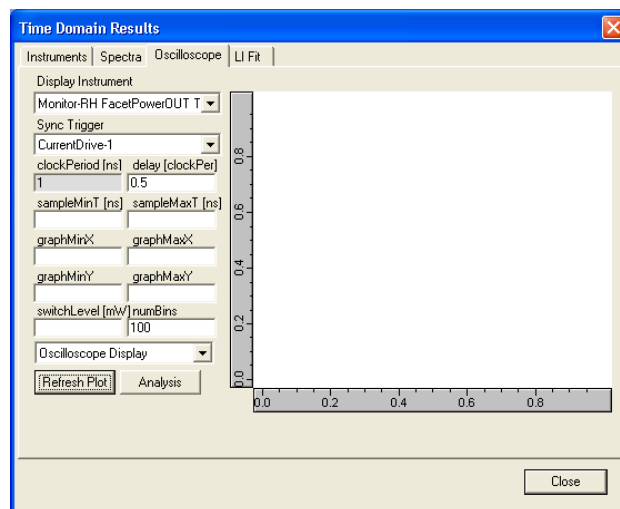
We have now created a *time-resolved spectrum*. This is very useful for monitoring the behaviour of laser diodes. If you see kinks in the *LI plot*, this *time-resolved spectrum* can often be used to quickly identify what the kink is due to. For example in this Fabry Perot laser you can see that initially, just above threshold, the spectrum is unstable but eventually it stabilises into one mode.

For more information on the spectra that PICWAVE can produce, and on the plotting options provided by SciGraph, PICWAVE's graph plotting and visualisation sub-system, see §9.6 and §16, respectively.

## 2.2.6 Generating an Eye Diagram

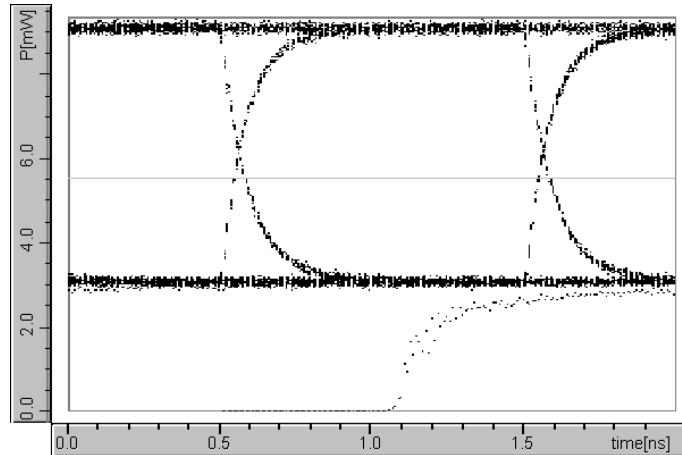
For digital transmission, the *eye diagram* is a very convenient diagnostic. To generate an *eye diagram*, we need to drive the laser with a pseudo-random bit pattern generated at some clock period  $T$  and then create a plot of the output signal with each period of the clock cycle plotted on top of the previous one. First we set up the bit pattern signal.

- Double-click on the *CurrentDrive-1* signal displayed in the Main panel of the *TDTW Calculator*. In the *signal elements* list delete the *ramp* entry.
- Set the element type to *NRZ* from the drop-down list and click **Insert**.
- Set **length** to 100 ns
- Set **Tp** to 1 ns
- Set **minVal** to 25mA
- Set **maxVal** to 50mA
- Set **seed** to 1
- Close the Signal Editor and set **totalTime** to 100 ns on the Main panel.
- Start the simulation.
- When it is finished, click **Inspect** and display the *Oscilloscope* panel:



The *Oscilloscope* can be used to study all sorts of signal and will conveniently generate an *eye diagram* for us. The **Display Instrument** setting tells the *Oscilloscope* what to plot and the **Sync Trigger** setting tells the *Oscilloscope* what to use as a trigger – the trigger determines at what part of the signal to start plotting. The **delay** setting allows you to shift the plot along the x-axis. By default the *Oscilloscope's* default settings will automatically set to the values required for an *eye diagram*, so:

- Click **Refresh Plot**. You should see something like:



In addition to generating *eye diagrams*, the *Oscilloscope* can provide many histograms and statistics of your switching signal.

For more information on the *Oscilloscope*, see §9.7.



## 2.3 Simulating a Bragg Grating with an FIR Filter

The *Finite Impulse Response Filter Section (FIR section)* allows one to model a component with an arbitrary spectral profile: it allows one to import a frequency domain model of a component from FIMMPROP or other compatible software (i.e. a wavelength dependent scattering matrix) from which it generates an equivalent time-domain model - an FIR filter. It thus allows a wide variety of passive components to be modelled accurately in FIMMPROP and then linked into a PICWAVE circuit for time-domain simulation.

In this example we demonstrate the use of an *FIR section* to model a Bragg grating which has been modelled in the frequency domain in FIMMPROP. The frequency domain model, or scattering matrix spectrum file, can be found on your CD at PICWave\Data\example\_smatspectrum.txt. The Bragg grating used to produce this scattering matrix spectrum file consisted of 70 layers with alternating refractive indices of 3.5 and 3.25. Its transmission was measured at 200 wavelengths between 1.47 and 1.57 $\mu$ m. [N.B FOR FIMMWAVE/FIMMPROP owners: an example demonstrating the FIMMPROP-PICWave link can be found, along with instructions, in PICWave\Examples\FMWX-PICWave link on your CD – this shows you how to generate your own scattering matrices using FIMMPROP – see §7.2].

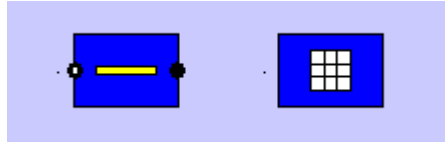
### 2.3.1 Preparing the circuit

To begin with, in an open project,

- Create a new device in the project by selecting the project in the project tree and clicking on the  Add PICWave Circuit button. Name it FIR Filter.
- Double-click on the FIR Filter entry in the project tree to open the device.
- Click on the  icon. Now click on the device canvas, somewhere in the middle, to drop an *FIR section* onto the device.
- Add a waveguide section to the device also. This dummy waveguide section will be needed later on as the *reference section* from which the TDTW simulation time-step (**tStep**) will be calculated; a *FIR section* cannot be used as the *reference section*.

The circuit should look as follows:





### 2.3.2 Importing a scattering matrix spectrum

The next stage is to import the scattering matrix spectrum (*S-Matrix spectrum*). An *S-Matrix spectrum* contains a definition of the ports on an *FIR section* along with a set of spectra (*S-Matrix elements*) which describe the wavelength-dependent mode coupling between the ports.

- In a text editor, open the aforementioned scattering matrix spectrum file on your CD. It should look as follows:

```
begin <SMatSpectrum(1.1)> "sMatrix"
200 2 1 // nlambda nports nspectra
1 1 1 "ID1" "descr1" // side nModesPort nEModesPort wgID description
2 1 1 "ID2" "descr2" // side nModesPort nEModesPort wgID description
(1, 1, 2, 1) // list matrix elements included in this file
// s-matrix spectra follow
1.47 { -0.8123068, -0.570303}
1.470391 { -0.8843665, -0.457635}
1.470781 { -0.9403692, -0.3348072}
1.471172 { -0.9784735, -0.2043644}
1.471563 { -0.9973695, -0.06931855}
1.471953 { -0.9964303, 0.0670182}
1.472344 { -0.9757914, 0.201255}
1.472734 { -0.9363383, 0.330162}
1.473125 { -0.8796035, 0.4508843}
1.473516 { -0.8075945, 0.5610953}
1.473906 { -0.7225881, 0.6590691}
1.474297 { -0.6269283, 0.7436695}
1.474687 { -0.5228596, 0.8142728}
```

Details on the *S-Matrix spectrum* format can be found in §7.1.4. The key things to note about this *S-Matrix spectrum* are that it defines one left-hand and one right-hand port, and that it defines the wavelength-dependent coupling between these ports for the first TE mode, at 200 discrete wavelengths lying between 1.47 and 1.57um.

- Select all the text in the file and copy it to the clipboard.
- In PICWAVE, right-click on the *FIR section* and select *IPaste Smatrix* to import the *S-Matrix spectrum*.

Note that the *S-Matrix spectrum* of a given *FIR section* can be viewed and edited in text format by right-clicking on the *FIR section* and selecting *IProperties*, and then double-clicking on **SMatrix**.

- If the FIR section is still selected click off it or press **Esc** and two ports should appear.

### 2.3.3 Setting the zStep and central simulation wavelength

An *FIR filter*, modelling the time-domain response of the Bragg grating, will be generated from the *imported S-Matrix spectrum*. The response of the *FIR filter* can be inspected before running a TDTW simulation. This requires, however, that the wavelength range of the TDTW simulation (**lamRange**) be defined first. This means that the zStep (**zStep**) of the reference section (in this case the dummy waveguide section) and central simulation wavelength (**lambdaCentre**) must be set (see §2.1.5).

- Open the TDTW Calculator window, in the Control panel set **lambdaCentre** to 1.52 um. (This happens to be approximately the centre of the stop-band band in the *imported S-Matrix spectrum*.)
- In the device window, open the properties panel of the waveguide section and select the **sectionName** text (e.g. *WGSection-1*). Copy this to the clipboard and paste it into the **refSectionName** text box in the Main panel of the TDTW Calculator. The

waveguide section “WGSection-1” will now be used as the reference section used to define the **zStep**.

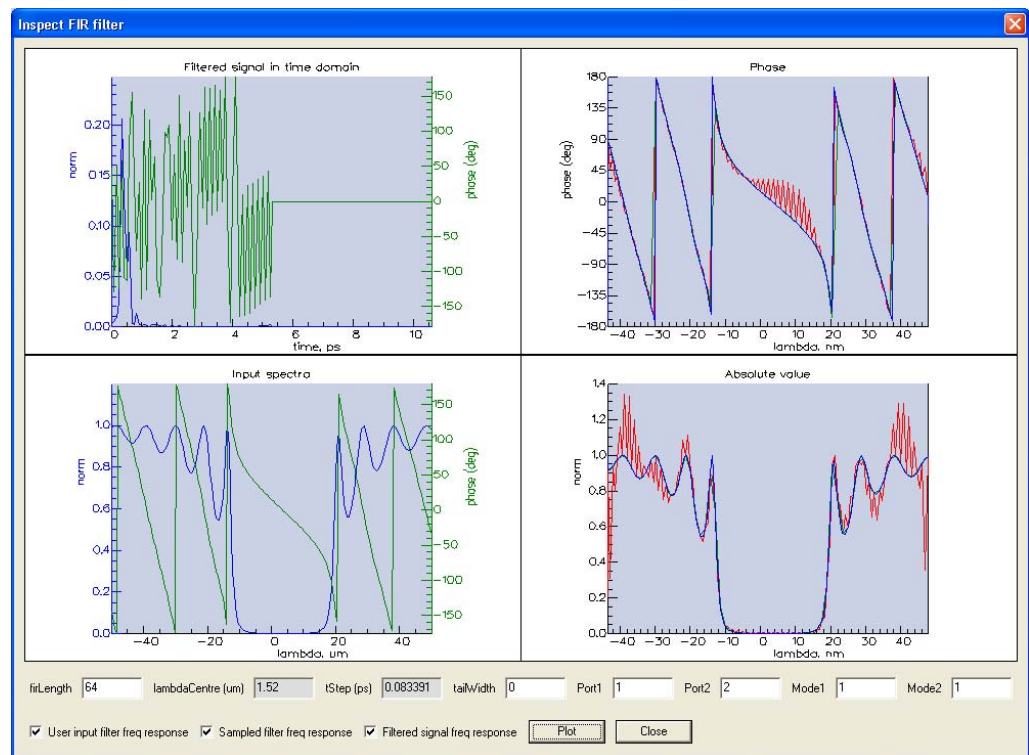
As was seen from the S-Matrix spectrum file, the *imported S-Matrix spectrum* covers a limited wavelength range. The free spectral range (**lamRange**) of the TDTW simulation cannot exceed this range. This places a lower limit on the value of **zStep** that can be used; PICWave will warn you if this limit is exceeded. In this case, the wavelength range covered by the *imported S-Matrix spectrum* is 100nm; the lowest allowable **zStep** can be computed to be approx. 24um. [Calculated from the expressions for **tStep** step and **lamRange** given in §2.1.5]

- Set the value of **zStep** to 25um
- Open the properties panel of the waveguide section and set the **length** to 25um (the reference section should have a **length** equal to **zStep**, at least)

### 2.3.4 Inspecting the FIR filter, FIR section properties

With the free spectral range defined the *FIR filter* can be inspected.

- In the device window, right-click on the *FIR section* and select *Inspect filters*.
- In the *Inspect FIR filter* window, click **Plot**. The window should then look as follows:



This window allows one to see the *FIR filter* that will be generated from the *imported S-Matrix spectrum*, i.e. to see its response in both the time and frequency domains. The S-Matrix element being plotted is defined by the indices Port1, Port2, Mode1, Mode2, so that all the spectra shown are for the coupling between the *Mode1*th mode of the *Port1*th port and the *Mode2*th mode of *Port2*th port. The parameters **lambdaCentre** and **tStep** from TDTW Calculator are also shown.

Plotted on the bottom left is the *imported S-Matrix spectrum*, which corresponds to the Bragg grating’s original frequency (wavelength) response as calculated by FIMMPROP; it is plotted over its entire wavelength range. The *FIR section* converts this

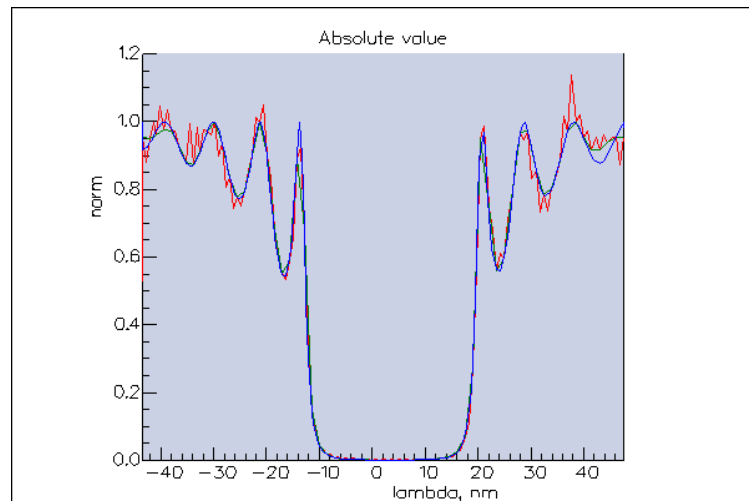
frequency (wavelength) response into a time-domain response by generating an *FIR filter*, whose coefficients are given by a forward-Fourier transform of a *sampled S-Matrix spectrum*. A *sampled S-Matrix spectrum* is produced by sampling the *imported S-Matrix spectrum* at number of wavelengths equally spaced over the free spectral range. The number of sampling wavelengths, which also corresponds to the number of *FIR filter* coefficients, is defined by the *filter length* (the **firLength** parameter)

Plotted on the right, for direct comparison, are the amplitude and phase of: the *imported S-Matrix spectrum* (blue); the *sampled S-Matrix spectrum* (green); the *FIR filter* frequency response spectrum (red). These are plotted over what will be the wavelength range of the TDTW simulation (**lamRange**). The *FIR filter* frequency response spectrum is that which will be observed in a TDTW simulation. The additional rapid oscillations, which are not present in the *imported S-Matrix spectrum*, are due to the truncation of the imported spectrum and the periodic nature of the *FIR filter*.

Plotted on the top left is the time domain response of the *FIR filter* to an impulse (delta function) signal. An *FIR filter* calculates the signal amplitude at a given time step from a linear combination of the amplitudes of a number, **firLength**, of previous time steps (see §7.1.1 for more details). Thus it can be seen that the length of the time-domain response is given by **firLength** x **tStep** = 64 x 0.083ps = 5.3ps.

Besides the *S-Matrix spectrum*, an *FIR section* has two other properties: the **firLength**, which has been discussed already, and the **tailWidth**. The rapid oscillations in the frequency response of an *FIR filter* can be reduced by modifying (distorting) the *sampled S-Matrix spectrum* so as to force it to be periodic over the free spectral range. This distortion takes place within two equally sized wavelength windows at each end of the free spectral range. The size of these windows, expressed as a fraction of the free spectral range, is specified by the **tailWidth**.

- Set the **tailWidth** to 0.1 and click **Plot**. The bottom right plot should then look as follows:



Note that the *sampled S-Matrix spectrum* (green) is now periodic and departs from the *imported S-Matrix spectrum* (blue) at each end of the free spectral range; and that the rapid oscillations in the *FIR filter* response (red) are much reduced.

- Click **Close** then click **Yes** when asked if you want to save the parameters.

### 2.3.5 Finishing the circuit and running the TDTW simulation

Now that the *FIR section* has been set up, the circuit can be completed.

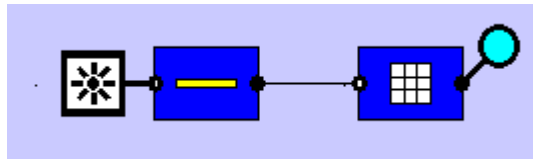
- Connect the right-hand port on the waveguide section to the left-hand port on the *FIR section*
- Add an optical source to the left-hand port of the waveguide section. By default this will inject an impulse, which has a flat-band optical spectrum, at  $t=0$ .

As the ingoing signal has a flat spectrum, the spectrum of the outgoing signal, exiting the right-hand port of the *FIR section*, will reveal the wavelength response of the Bragg grating as modelled by the *FIR section*.

To measure the spectrum of the outgoing signal:

- Add a monitor to the right-hand port of the *FIR section*.
- Right-click on the monitor, select *Properties*, and in the monitor properties panel set **enableSpecMeas** to *Out*.

Your circuit should now look as follows:



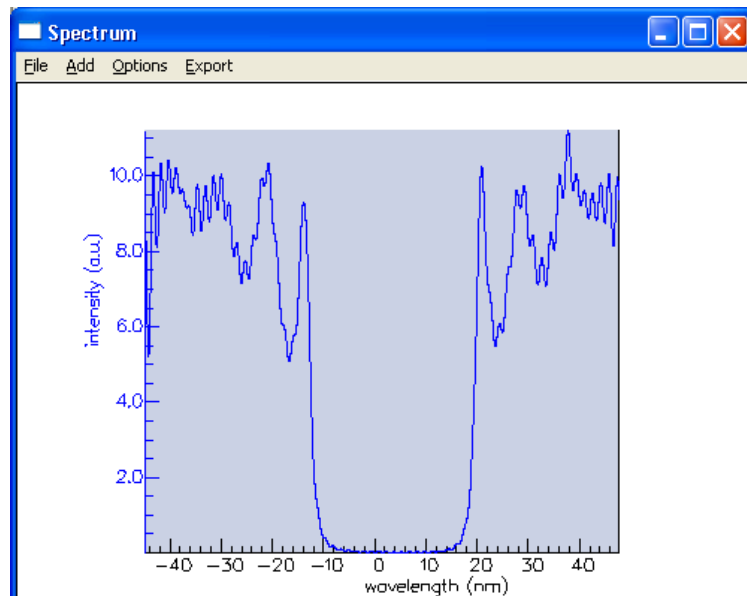
The simulation is now ready to run.

- Click the  button. The simulation will begin.

When the simulation has finished,

- Click the **Inspect** button. The Time Domain Results window will appear.
- In the Spectra tab, ensure that the **Monitor** is set to *Monitor-1: TE-1*. Select *Intensity (lin)* in the **Magnitude Axis** box, then click **Plot**.

The spectrum of the output signal, which shows the wavelength response of the Bragg grating as modelled by the *FIR filter*, should look as follows:



For more information on the FIR section, see §7.

## 2.4 Simulating an SOA with a multiple-Lorentzian Gain Model

The *Multiple Lorentzian Gain Fitter* tool in PICWAVE allows one to import gain data from an external text file and fit it with a multiple Lorentzian gain model. A *multiple Lorentzian gain model* offers two advantages over PICWAVE's *single Lorentzian gain model* (which is specified by parameters in the material database system): 1) it provides an easy way of producing a gain model that is valid over a range of carrier densities (and temperatures) without the need for manual calibration of the gain parameters in the material database, 2) it allows one to model the gain accurately over a larger portion of the free spectral range and, importantly, away from the gain peak. This example illustrates the process of importing gain data, fitting a gain model, and using the gain model in an SOA simulation.

The gain data to be imported is contained on your CD at: PICWave\Examples\HRLDLink\example\_gaindata.txt. This material gain data was produced by HAROLD from the following epi-layer structure:

Layer number	Material	Thickness (um)	Doping (cm <sup>-3</sup> )	Comments
1	AlGaAs	0.5	n = 1.5x10 <sup>19</sup>	-
2	Al <sub>0.6</sub> Ga <sub>0.4</sub> As – GaAs (linear alloy comp variation)	0.25	n = 1.5x10 <sup>19</sup>	-
3	Al <sub>0.6</sub> Ga <sub>0.4</sub> As	1.47	n = 1.5x10 <sup>19</sup>	-
4	Al <sub>0.26</sub> Ga <sub>0.74</sub> As - Al <sub>0.6</sub> Ga <sub>0.4</sub> As (linear alloy comp variation)	0.13	-	barrier
5	Al <sub>0.1</sub> Ga <sub>0.9</sub> As	0.012	-	QW
6	Al <sub>0.6</sub> Ga <sub>0.4</sub> As - Al <sub>0.26</sub> Ga <sub>0.74</sub> As (linear alloy comp variation)	0.13	-	barrier
7	Al <sub>0.6</sub> Ga <sub>0.4</sub> As	1.58	p = 1x10 <sup>18</sup>	-
8	GaAs - Al <sub>0.6</sub> Ga <sub>0.4</sub> As	1.5	p = 1x10 <sup>18</sup>	-
9	GaAs	0.2	p = 1x10 <sup>18</sup>	-
10	GaAs	100	-	substrate

The gain data consists of material gain spectra at a set of 50 carrier densities, in this case at a temperature of 25°C. All the spectra cover a wavelength range 0.78-0.82um. The file can be opened for inspection in a text editor; details on the data format can be found in §8.3.1.

[N.B FOR HAROLD owners: an example demonstrating the HAROLD-PICWave link can be found, along with instructions, in PICWave\Examples\HRLD-PICWave link on your CD – this shows you how to generate your own material gain data files c.f. HAROLD manual Chapter 6].

### 2.4.1 Importing the gain data

To begin with, a new material database file must be created, which will be set to import the gain data for a particular material from an external file.

- Make a copy of the `refbase_picwave.mat` material database file that is on your CD and put it in the directory where your project file will be saved.
- Rename the material database file to `ml_example.mat`.
- Copy the gain data file (`example_gaindata.txt`) from you CD and put it in the same directory as the new material database file.
- Open `ml_example.mat` and find the AlGaAs material definition:

```
BEGIN AlGaAs(x)    // material name and template
.
.
.
END
```

An AlGaAs epi-structure was used to produce the gain data, and so the gain data will be imported for the AlGaAs material.

- Insert a new line before the line: `END` as follows:

```
BEGIN AlGaAs(x)    // material name and template
.
.
.
IMPORT_GAIN_SPECTRA example_gaindata.txt
END
```


This extra line will tell the material database system to import material gain data for the AlGaAs material from the file `example_gaindata.txt`. That which follows the `IMPORT_GAIN_SPECTRA` flag is a relative path and file name of the gain data file, in this case only the file name is needed as the gain data file is in the same directory as the material database file. The `IMPORT_GAIN_SPECTRA` will also tell the material database system that a multiple Lorentzian gain model is to be used for the AlGaAs material; the gain model in the AlGaAs material definition (`GAIN_POLYL`, `GAIN_POLYN` parameters etc) will be ignored in TD'TW simulations in which this material is used.

- Save and close `ml_example.mat`. (you might have to change the read-only status before you are able to save it).

#### 2.4.2 Creating the RWG

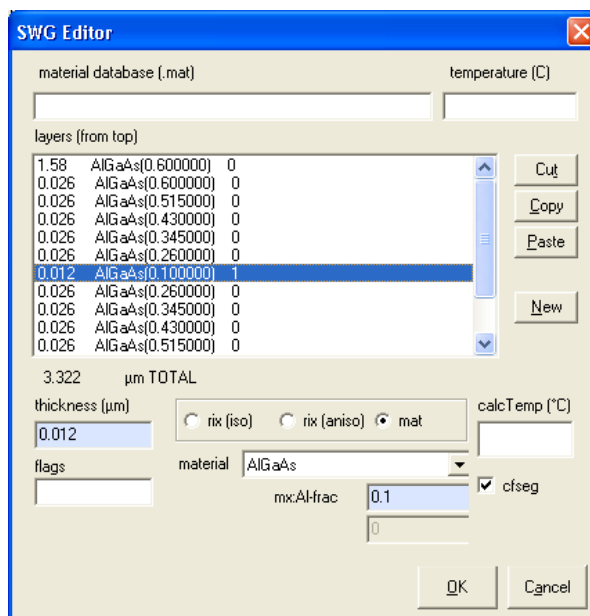
The next stage is to create an RWG that will use the AlGaAs material in an active layer. This RWG will be used to define the cross-section of the waveguide in the SOA device that will be created later. All active materials used in a given device (via RWGs) which are set to import gain data are accessible to the *Multiple Lorentzian Gain Fitter* Tool in the device window, as will be shown later.

- Create a new project and save it in the directory containing the material database and gain data files.
- Add a new RWG to the project and name it SOA WG.
- Double-click on SOA WG node in the project tree to open its RWG window

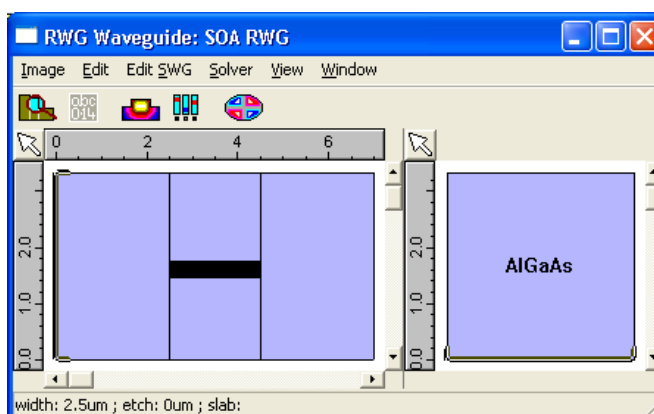
- Select **/Edit/Set Material database (.mat)** , then select `ml_example.mat`
- Next, click  to open the RWG Editor
- In the RWG Editor, click **New Slice** three times. Set their **width** values to 2.5, 2 and 2.5  $\mu\text{m}$  respectively.
- Double-click on the first slice in the list of slices to open its SWG Editor. By default there is already one layer in this slice: set its thickness to 3.322  $\mu\text{m}$ , click the **mat** radio button then select the AlGaAs material from the drop-down list, finally set the **mx:Al-frac** to 0.6 (you may have to click on the layer again for this to become enabled)
- Click **OK** to return to the RWG Window and repeat the previous step for the third slice in the list.
- Next, double click on the second slice in the RWG Window and set up the following layers:

Layer number	thickness ( $\mu\text{m}$ )	mx:Al-frac	cfseg
1	1.58	0.6	-
2	0.026	0.6	-
3	0.026	0.515	-
4	0.026	0.43	-
5	0.026	0.345	-
6	0.026	0.26	-
7	0.012	0.1	enabled
8	0.026	0.26	-
9	0.026	0.345	-
10	0.026	0.43	-
11	0.026	0.515	-
12	0.026	0.6	-
13	1.470	0.6	-

These layers correspond to epi-layer structure given above, the layers with linearly varying composition have been approximated using series of thinner layers. In the SWG Editor the layers should look as follows:



- Click **OK** to return to the RWG Editor, then click **OK** to return to the RWG Window. The RWG should now look as follows:



- Open the MOLAB options and set **max Nmodes** to 1  
The RWG is now complete.

### 2.4.3 Creating the SOA circuit

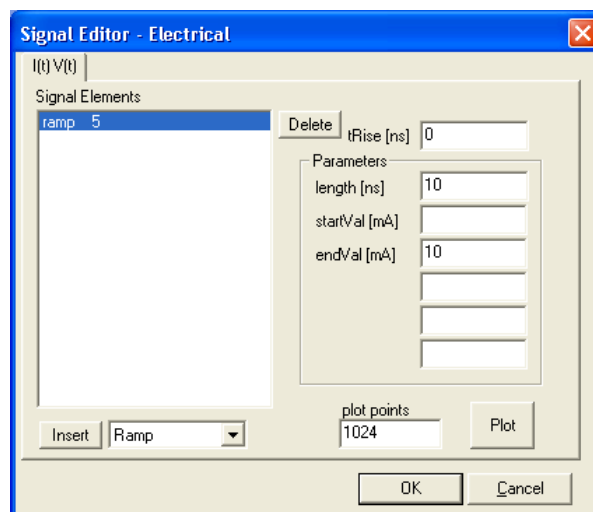
The next stage is to create the SOA circuit:

- Add a new PICWave circuit to the project and name it *SOA device*
- Double-click on the SOA device circuit in the project tree to open its device window.
- In the device window add a waveguide section and set its properties to the following:



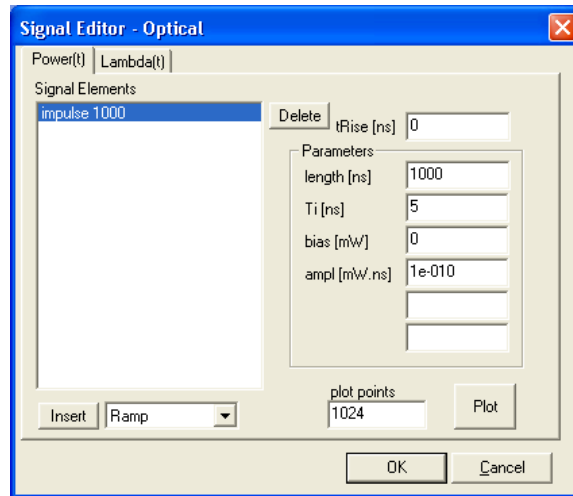
Parameter	Value	Units
sectionName	WGSection-1	
crossSectionDef	waveguide	
sectionType	gain	
modePol	TE-like	
length	500	$\mu\text{m}$
Tsub	25	Celcius
<b>Active properties</b>		
sponBetaNA	0.001	
numContacts	1	
CurrentFlowModel	more...	
ElectroAbsorptionModel	more...	
ncellW	1	
instruments	more...	
<b>Grating properties</b>		
gratPhaseShift	0	deg
CouplingParms	more...	
<b>TE properties</b>		
numTEModes	1	
effIndexTE	1	
effGroupIndexTE	1	
effLossTE	0	[1/cm]
<b>TM properties</b>		
numTMModes	0	
effIndexTM	1	
effGroupIndexTM	1	
effLossTM	0	[1/cm]

- Double-click on the **CurrentFlowModel** value (*more...*) and add a new connection with resistivity 1000 Ohm  $\mu\text{m}^2$
- Double-click on the **instruments** value (*more...*) to open the Waveguide Section Instruments panel. Add a Junction Ne instrument and, double-clicking on it, set **zFraction** to 0.5 so that the carrier density is measured halfway along the waveguide.
- Close the Waveguide Section Properties panel when finished
- In the device window add a current drive to the waveguide section.
- Press **Esc** then right-click on the current drive and select **/Properties**. Then click on the **signal** property to open the Signal Editor. Add a ramp element and set it as follows:



- Click **OK** when finished and return to the device window.

- Attach an optical source to the left-hand port of the waveguide section. Open its signal editor (in the same manner as with current source). An impulse signal already exists, edit its Power (t) signal to the following:

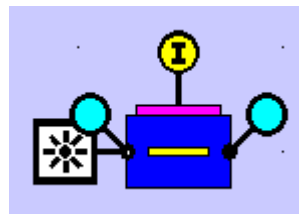


- Click **OK** and return to the device window.

The impulse signal will be used to measure the small-signal gain of the SOA device later on.

- Next, right click on the SOA RWG in the project tree and select */Copy*, then right-click on the waveguide section in the SOA device window and select */Paste WG Ref*.
- Finally, add a monitor to the right-hand port of the waveguide section. Right-click on the monitor and select */Properties*. In the Monitor Properties panel set: **monitorName** to *Monitor-Out*, and **enableSpecMeas** to *Out*, so that the spectrum of the outgoing amplified optical signal will be measured.
- Similarly add a monitor to the left-hand port of the waveguide section; set **monitorName** to *Monitor-In*, and **enableSpecMeas** to *In*, so that the spectrum of the ingoing optical signal will be measured.

The SOA device circuit is now complete and should look as follows:



#### 2.4.4 Fitting a multiple Lorentzian Gain model

Setting **tStep** and **lambdaCentre**

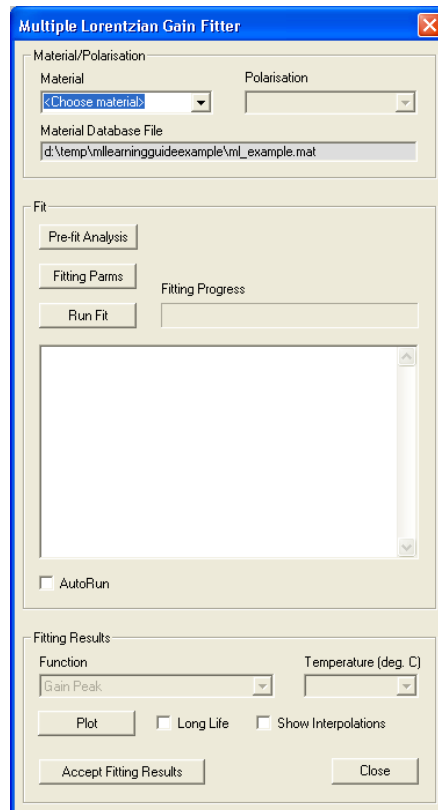
In order to fit a *multiple Lorentzian gain model*, **tStep** and **lambdaCentre** must be defined. These parameters determine the free spectral range of the TDTW simulation and thus determine the wavelength window over which the imported gain spectra must be fitted. This means that a fitted multiple Lorentzian gain model is specific to the particular values of **tStep** and **lambdaCentre** for which it was fitted. If these parameters are changed, the gain model will have to be re-fitted. As **tStep** and **lambdaCentre** are associated with the simulation of a *particular device*, fitted multiple Lorentzian gain

models can be used only with the device for which they were fitted i.e. they cannot be shared with other PICWave circuits

- From the device window, open the TDTW Calculator and in the Main panel set **zStep** to 10  $\mu\text{m}$ ; then, in the Control panel, set **lambdaCentre** to 0.805  $\mu\text{m}$ , which is approximately equal to the peak gain wavelength for the imported material gain spectra.

We are now ready to fit the gain model.

- From the device window, select */Tools/Multiple Lorentzian Gain Fitter* to open the *Multiple Lorentzian Gain Fitter* tool.



Selecting the material and polarisation

The **Material** drop-down list at the top of the window contains all the active materials used by the device which are set to import gain data and use *multiple Lorentzian gain models*.

- Select *AlGaAs* from the **Material** drop-down list. The **Polarisation** drop-down list should become enabled.

The *Multiple Lorentzian Gain Fitter* allows gain models for both the TE and TM polarisation to be fitted for each active material. Separate gain data files are needed for each polarisation. The **Polarisation** drop-down list lists all polarisations for which gain models can be fitted (i.e. for which gain data is available). This example uses only one gain data file, which contains TE gain data.

- Select *TE* from the **Polarisation** drop-down list. Note that it is labelled as “not-fitted” as the gain model has yet to be fitted.

Pre-fit Analysis

- Click **Pre-fit Analysis** to open the *Pre-Fit Analysis* window.

The *Pre-Fit Analysis* window allows one to see the imported spectra that are to be fitted and also gives the carrier density and wavelength ranges (the parameters: **minN**, **maxN**, **minLambda** and **maxLambda**) covered by the set of imported spectra, and the transparency carrier density (**NO**).

- Click **Plot**. This will plot, on one graph, the set of imported gain spectra contained in the gain data file, over their entire wavelength range. Note that this plot option is denoted by *Raw Spectra* in the **Plot** drop-down list.
- From the **Plot** drop-down list select, *Sampled Spectra*. Then click **Plot**. This will plot the portion of the imported spectra lying within the wavelength “window” that corresponds to the free-spectral range, that is, the portion that will be fitted to produce the gain model. The free spectral range wavelength window is shown underneath the plot (**lamRange**).

The *Pre-Fit Analysis* window allows one to make an informed choice about the *gain fitting parameters*, which are briefly discussed below.

#### Gain Fitting Parameters

- In the *Pre-Fit Analysis*, click **Fitting Parms**. This will open the *Gain Fitting Parameters Panel*, in which the *gain fitting parameters* are set.

The *gain fitting parameters* control the fitting process in which the imported spectra are fitted with a function comprised of the several pseudo-Lorentzian curves. Amongst other things these settings allow one to specify: the carrier density range over which to do the fitting, the portion of the free spectral range over which to fit the spectra, extrapolation settings and parameters for scaling the imported spectra data. These parameters are discussed further in §8.2.3. For this example the default gain fitting parameters can be used.

#### Running the fit

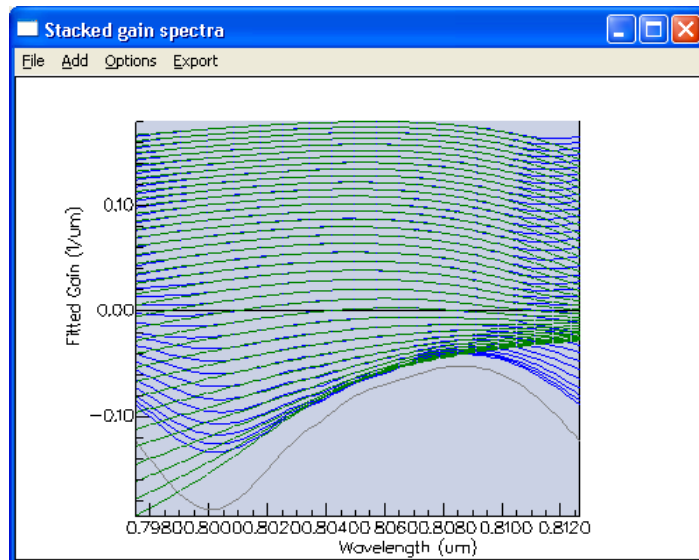
- Return to the main window of the Multiple Lorentzian Gain Fitter.
- Click **Run Fit**. This will start the fitting process, the fitting progress can be observed from the progress bar and the fitting log.

The fitting algorithm fits the imported gain spectra according to the gain fitting parameters. Each imported spectrum is fitted with a number of pseudo-Lorentzians, which are periodic over the free spectral range. The periodicity of the pseudo-Lorentzians is required by the TDTW algorithm employed by PICWave. Consequently the fitted gain spectra are periodic over the free spectral range; this places limitations on the fitting accuracy at each end of free spectral range as will be seen.

#### Inspecting the fitting results

When the fitting is completed the **Function** and **Temperature** drop-down lists in the Fitting Results section of window will become enabled.

- From the **Temperature** drop-down list select 25.
- From the **Function** drop down list, select *Stacked Spectra* and then click **Plot**. This will plot the set of fitted spectra (blue) on top of the imported spectra (green) over the wavelength range corresponding to the free spectral range. The plot should look as follows:




Notice that the fitting is accurate across about 70% of the free spectral range; in particular, note that the fitting is accurate even away from the gain peak – an advantage of a *multiple Lorentzian gain model* as opposed to *single Lorentzian gain model*, with which one could only expect accuracy around the gain peak. Notice also the periodicity of the fitted spectra and the evident limits it places on the fitting accuracy at edges of the free spectral range: the TDTW algorithm requires that the material gain be the same at each end of the free spectral range as discussed above.

Other parameters of the fitted and imported spectra can be viewed here, they are discussed in §8.2.1. For now,

- Click **Accept Fitting Results** to accept the fitted gain model and close the *Multiple Lorentzian Gain Filter*.

The multiple Lorentzian gain model for the AlGaAs material has now been created and the TDTW simulation can be run.

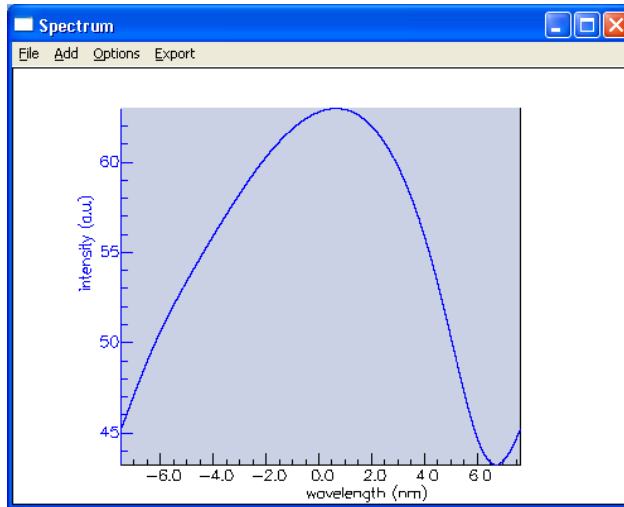
#### 2.4.5 Running the simulation and inspecting the device gain of the SOA

- From the device window, open the TDTW Calculator. In the Main panel set **totalTime** to 10ns and in the Control panel switch off all noise sources. Switching off the noise sources will allow the underlying device gain to be measured.
- Click  to start the simulation

The simulation has been set up so that the SOA will be driven to a steady state before the low power impulse signal is injected. The impulse signal has a flat-band optical spectrum, so the spectrum of the amplified signal will correspond to the device gain spectrum of the SOA.,

- When the simulation has completed, click **Inspect** and in the Instruments Results panel double click on the JunctionN instrument to display the carrier density as function of time. Here you can verify that the carrier density was level by the time that the impulse was injected at 5ns, and that the pulse itself was of sufficiently low power to leave the carrier density unchanged; a changing carrier density would lead to spurious features on the device gain spectrum which will be plotted next.

- Go to the Spectra Results panel. Select *Monitor-Out* from the **Monitor** drop-down list, select *Monitor-In* from the **Reference** drop-down list. Then select Intensity (lin) in Magnitude Axis box and click **Plot**. You should obtain the following:



This plot shows the spectrum of the outgoing signal, the amplified impulse, divided by the spectrum of the ingoing impulse signal i.e. the device gain spectrum.

Notice the asymmetry of the device gain; a multiple Lorentzian model is needed to model such asymmetry. Notice also the periodicity of the device gain spectrum over the free spectral range and the correspondence between its shape and shapes of the fitted material gain spectra (well above transparency) which were plotted earlier.

This example thus illustrates how a multiple Lorentzian gain model can be used when modelling SOAs over a broad band.

For further details on the *Multiple Lorentzian Gain Fitter* and on the process of *multiple Lorentzian gain models*, see §8.

## 2.5 Other Examples

Your CD contains a number of other examples that you may wish to explore. Open the file:

PICWave\Examples\ActiveExamples.prj

This project contains the following examples:

### FabryPerot

This is a model of a Fabry Perot single section laser. There are three simulations for this device:

1. FPDevice-switchOn – a simple steady state time domain simulation with 100mA current being switched on at time  $t=200\text{ps}$ . Plot a time-resolved spectrum and notice how the FP laser settles quickly into one mode. (We recommend you plot this as a colour-map – right-click on the plot and select /Display Options, then enable **Colour Map**, then **2D View**.)
2. FPDevice-EyeDiagram – this generates an eye diagram from an NRZ bit pattern applied to the drive contact. Run the simulation, and from the TD Results/Eye Diagram tab, set Instrument to FacetPowerOutRHS and click Plot. You should see a nice noisy eye diagram.

3. FPDevice-ImpulseResponse – the device is biased at 24mA and after a short while a 1mA.ns impulse is injected via the electrical contact. Since the input is a delta-function, this spectrum should be a good approximation to the modulation response spectrum.
4. FPDevice-GainSwitching – illustrates gain switching of a laser. The laser is driven at 20mA bias and modulated with a 1GHz +/- 20mA current signal. Run simulation and see laser pulsing strongly.

For full details and instruction for this simulation see the “Project Notes” in the “FabryPerot” project.

### QshiftDBR

This is a model of a quarter-wave shifted DBR laser. There are two simulations for this device:

1. qShiftDevice-LI-Plot – generates a light-current curve. Start the simulation and when it is complete create an LI curve as follows
  - Open the TD Results/Instruments panel
  - Select DrvNode1 instrument and then click the ➔ arrow. This tells the program to use DriveNode1 for the x-axis
  - Double click on the FacetPowerOUT(RH) instrument. You should now have a (somewhat noisy) LI curve. To reduce the noise you could increase the response time of the FacetPower instrument and re-run the simulation.
  - PICWAVE also includes an LI curve analysis routine for measuring the threshold and differential quantum efficiency:
  - Select the TD Results/LI Fit panel. Click the **Run Fit** button. You should see the threshold current and differential QE displayed.
  - Click **Plot Fit** to see the fitted curve.
2. qShiftDevice-InjectTest – In this case we simulate the effect of an external optical signal injected into the DBR laser. The signal is a constant 0.1mW but sweeping from -8nm to +8nm about the central wavelength in 10ns.
  - Start the "InjectTest" simulation. Wait for it to complete. You should see green and red lines in the Calculator Run-time Monitor Window. These are the optical powers exiting left and right facets.
  - Click on the [Inspect] button and then plot the time-evolving spectrum as you did before. You will see a line moving diagonally across the image, this is the amplified injection signal. The laser mode crosses it horizontally. You can also see faint additional lines at higher angles - these are generated by 4-wave mixing.

### FP\_BuriedHet-ExtCavity

This is a buried heterostructure FP laser with a 1mm external cavity.

- Double click on the ExtCavityFPDevice in the project tree on the left to display the structure.
- /View/Toggle real/symbolic view to show the device in real view

- Run the FPExtCavityTD simulation and plot a time-resolved spectrum. This will run an LI simulation – ramping up the current to 100mA. You will see that the structure is strongly multi-moded.

Read the notes attached to the **FP\_BurriedHet** project for further information.

### ModeLocking

This is an example of a mode locked laser. The device is a simple FP laser with a short saturable absorber section at the right hand end. To simulate the short carrier lifetime in the reverse biased absorber section, a special material has been defined for the active layer of this section – InGaAsP\_SA. This material has a high non-radiative spontaneous recombination rate to simulate the rapid speed at which the carriers are swept out of the active region by the reverse bias.

Read the notes attached to the **ModeLocking** project for further information.

### 4Wave Mixing

This project illustrates 4-wave mixing in an SOA. The device is an SOA with low reflectivity facets. 3 optical channels are injected into the SOA:

Channel 1: 1mW CW , +0nm

Channel 2: 1mW NRZ, +1nm

Channel 3: 1mW NRZ, +4nm

Channel 1 is a steady state signal and channels 2 & 3 are modulated by an NRZ bit pattern.

Run the "MultiLambdaInjection" simulation and wait until it completes. Click the Inspect button and then plot the time-resolved spectrum - ([Spectrum vs time] button in "Spectra" tab.) In the spectra you can clearly see not only the 3 channels but also many other wavelengths generated by four-wave mixing.

A further active device example, a regenerator, with accompanying notes can be found in:

PICWave\Examples\2R-Regenerator.prj

Various passive device examples with accompanying notes can be found in:






PICWave\Examples\PassiveExamples.prj




# Chapter 3

## Basic Tools

PICWAVE has a number of basic tools for manipulating and viewing objects such as sections and joins in the *device window*.

- the  *select tool* can be used to select and move objects;
- the  *scroll tool* can be used to scroll the device in the window;
- the  *zoom tool* can be used to zoom in and out;
- the  zoom to fit button can be used to zoom so that the device fits in the window;
- the  grid settings button can be used to change the grid and snapping settings;

### 3.1 The Select Tool

The  *select tool* can be used to select and move objects.


➤ To select the *select tool*:

click on the  *select tool*

or press the **Escape** (or **Esc**) key on your keyboard

or select **Tools/Select**

or right-click anywhere on the device and select **/Select**.

The cursor will change to a  *select cursor*.

#### 3.1.1 Selecting Objects

You will need to select objects before you can perform certain operations on them, such as deleting, copying and moving multiple objects together.

➤ To select an object, select the  *select tool* (see §3.1) then click on the object.

The object will be shown with a black and white rectangular border to indicate that it is selected. In addition, some objects will appear with various *handles*.

For example, a selected section is shown with a black and white rectangular border:



Sometimes you will need to select several objects at the same time, for example if you want to move them all together.

- To select more than one object, click on the first object, then hold down the **Shift** key and click on each of the other objects in turn.
- To select a number of objects in the same area of the device, click somewhere on the device where there is no object, hold the mouse button down, then drag out a rectangle to enclose the objects.
- To add an object to the selection, hold down the **Shift** key and click on the object.
- To add a number of objects in the same area of the device to the selection, hold down the **Shift** key, click on the device where there is no object, hold the mouse button down, then drag out a rectangle to enclose the objects.
- To remove an object from the selection, hold down the **Shift** key and click on the selected object.
- To unselect all objects, click anywhere on the device where there is no object.

Sometimes you may need to select an object that is hidden behind another object.

- To select an object that is hidden behind another object, move the mouse pointer over the hidden object, click once to select the front object, then hold down the **Alt** key and click again.


Note that if there are more than two objects below the mouse pointer, you can hold down the **Alt** key and keep clicking to cycle through the objects.

- To select all the objects in the device:

select **Edit/Select All...**

*or* press **Ctrl+A**.


### 3.1.2 Moving Objects

- To *move* an object, select the  *select tool* (see §3.1), click on the object, hold the mouse button down, and drag the object to its new position.

If *snapping to grid* is enabled, the *origin* of the object will be snapped to the *snap spacing*. See §3.5.2 for how to enable and disable snapping.

- To *move* a number of objects together, select the objects (see §3.1.1 for how to select several objects at the same time), then click on any of the selected objects, hold the mouse button down, and drag the objects to their new positions.


### 3.1.3 Nudging Objects

- To *nudge* an object a fixed distance up, down, left or right, select the  *select tool* (see §3.1), click on the object to select it, then use the arrow keys on the keyboard to *nudge* the object up, down, left or right.

The distance the object is moved when it is nudged in this way is equal to the *snap spacing* in the z- or x-direction. See §3.5.2 for how to change the *snap spacing*.

- To *nudge* an object by 10 times the *snap spacing*, hold down the **Shift** key while pressing the arrow keys on the keyboard.
- To *nudge* an object by 1/10th the *snap spacing*, hold down the **Ctrl** key while pressing the arrow keys on the keyboard.



### 3.1.4 Deleting Objects



- To *delete* an object, select the  *select tool* (see §3.1), click on the object to select it, then
  - select **Edit/Delete**
  - or* press the **Delete** (or **Del**) key.

Note that if there are any *links* between the deleted object and other objects, these will be deleted too (see §4.11 for details of *links*).

- To *delete* a number of objects at the same time, select the objects (see §3.1.1 for how to select several objects at the same time), then delete them as above.


### 3.1.5 Copying Objects

- To copy an object to the clipboard, select the  *select tool* (see §3.1), click on the object to select it, then
  - click on the  copy button
  - or* select **Edit/Copy**
  - or* press **Ctrl+C**.

- To cut an object to the clipboard, select the  *select tool* (see §3.1), click on the object to select it, then
  - click on the  cut button
  - or* select **Edit/Cut**
  - or* press **Ctrl+X**.

- To cut or copy a number of objects at the same time, select the objects (see §3.1.1 for how to select several objects at the same time), then cut or copy them as above.

- To paste the object or objects on the clipboard into the device:

- click on the  paste button
- or* select **Edit/Paste**
- or* press **Ctrl+V**.

If the objects were cut or copied from another device, they will be pasted in exactly the same position as they were in the other device. If, however, they were cut or copied from this device, they will be pasted at a slightly offset position so that they are distinct from the original objects.

### 3.1.6 Object Properties

- To change the *properties* of an object:
  - right-click on the object, then select **/Properties...**
  - or* click on the object to select it, then select **Object/Properties**
  - or* click on the object to select it, then press **Alt+Return** or **Alt+Enter**.


The relevant *properties* panel will appear, allowing you to enter precise values for the properties of the object.


### 3.1.7 Rotating Objects


- To *rotate* an object (a section or a join):  
right-click on the object, then select */Rotate../Right, Left or 180 degrees*.

One *port* on each *section/join* is marked with a white dot; this is helpful in determining the relative orientations (after rotations).

## 3.2 Scrolling

The  *scroll tool* can be used to scroll the device in the window.

- To select the *scroll tool*:
  - click on the  *scroll tool*
  - or* select **Tools/Scroll**
  - or* right-click anywhere on the device and select */Scroll*.


The cursor will change to a  *scroll cursor*.


- To scroll the device in the window, click anywhere on the device, hold the mouse button down and drag.

You can also use the scroll bars to scroll the device in the window in the usual way.

## 3.3 Zooming

### 3.3.1 The Zoom Tool

The  *zoom tool* can be used to zoom in and out.


- To select the *zoom tool*:
  - click on the  *zoom tool*
  - or* select **Tools/Zoom**
  - or* right-click anywhere on the device and select */Zoom/Zoom tool*.

The cursor will change to a  *zoom in cursor*.


- To zoom in, click on the device.

The device will be shown at twice magnification centred on the point where you clicked. You can click again to zoom in to four times magnification, and again to zoom in to eight times magnification, and so on.

- To zoom in on a particular area, click on the device, hold the mouse button down, and drag out the area you want to zoom in on.
- To zoom out, hold down the **Shift** key and click on the device.

When you hold down the **Shift** key, the cursor will change to a  *zoom out cursor*. After clicking, the device will be shown at half the previous magnification, centred on the point where you clicked.

### 3.3.2 The Zoom to Fit Button

- To zoom so that the device fits the window, click the  zoom to fit button.


### 3.3.3 Other ways to Zoom

- To zoom in:
  - press **Ctrl+]**
  - or* select **View/Zoom/In**
  - or* right-click on the device and select **/Zoom/In**
  - or* use the *zoom tool* as described in §3.3.1.

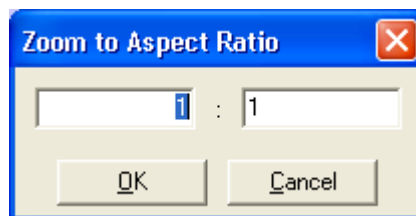
The device will be shown at twice the previous magnification.


- To zoom out:
  - press **Ctrl+[**
  - or* select **View/Zoom/Out**
  - or* right-click on the device and select **/Zoom/Out**
  - or* use the *zoom tool* as described in §3.3.1.

The device will be shown at half the previous magnification.

- To zoom so that the device fits the window:
  - click the  zoom to fit button
  - or* press **Ctrl+}**
  - or* select **View/Zoom/Fit window**
  - or* right-click on the device and select **/Zoom/Fit window**
- To zoom so that the device is shown at the correct aspect ratio, i.e. so that the z- and x-scales are the same:
  - select **View/Zoom/1:1**
  - or* right-click on the device and select **/Zoom/1:1**
- To zoom so that the device is shown at an aspect ratio other than 1:1:
  - select **View/Zoom/Aspect ratio...**
  - or* right-click on the device and select **/Zoom/Aspect ratio...**



The *Zoom to Aspect Ratio dialog* will appear:



- Type the required aspect ratio in the text boxes then click .
- To return to the previous magnification:
  - press **Ctrl+{**
  - or* select **View/Zoom/Previous**

or right-click on the device and select */Zoom/Previous*

## 3.4 Undoing and Redoing

The  undo button and  redo button can be used to undo and redo any changes you make to a device, up to a maximum of 20 changes.

➤ To undo a change:

click on the  undo button

or select **Edit/Undo**

or press **Ctrl+Z**.

➤ To redo a change:

click on the  redo button

or select **Edit/Redo**

or press **Ctrl+Y**.

## 3.5 The Grid

The *Grid Settings dialog* can be used to change the appearance of the *grid* and to enable or disable *snapping to grid*.

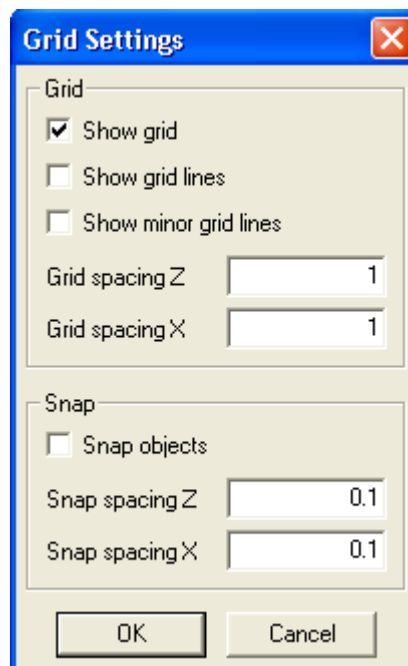
➤ To show the *Grid Settings dialog*:


click on the  grid settings button

or select **Edit/Grid Settings...**

or right-click on the device and select */Grid Settings...*

The *Grid Settings dialog* will appear:



➤ When you have made changes as described in the following sections (§3.5.1 and 3.5.2), click  to apply them.

### 3.5.1 Grid Settings

The grid is a visual aid for aligning objects with device coordinates. It corresponds to the x-z plane: the x-direction being vertical; the z-direction horizontal.

- To show the grid, check the **Show grid** checkbox; if the box is unchecked, no grid lines or points will be shown.
- To show grid lines, check the **Show grid lines** checkbox; if the box is unchecked, no lines will be shown, only points at the intersections of the lines.
- To show minor grid lines halfway between the main gridlines, check **the Show minor grid lines** checkbox; if the box is unchecked, no minor grid lines will be shown, only the main grid lines; if the main gridlines are not enabled, the minor gridlines cannot be enabled either.
- To change the spacing between the grid lines or points, enter values in  $\mu\text{m}$  in the Grid spacing Z and Grid spacing X fields.

### 3.5.2 Snap Settings

*Snapping to grid* is an aid for aligning objects with device coordinates. It can be useful for setting up devices quickly and precisely using the graphical tools.

For example, if *Snapping to grid* is enabled with a *snap spacing* of  $5\ \mu\text{m}$  in both the z- and x-directions, you will only be able to place, move, resize, deform, etc. objects to z- and x-coordinates that are exact multiples of  $5\ \mu\text{m}$ .

- To enable Snapping to grid, check the **Snap objects** checkbox; if the box is unchecked, you will be able to place and move objects etc without snapping.
- To change the snap spacing, enter values in  $\mu\text{m}$  in the Snap spacing Z and Snap spacing X fields.
- To disable snapping to grid temporarily while placing and moving objects etc, hold down the **Ctrl** key while dragging.

## Chapter

# 4

## Circuits

A PICWAVE *circuit* contains a complete description of a device made up of *sections* and *joins*. A *section* represents a physical optical element with a finite length such as a waveguide or directional coupler, whereas a *join* is a virtual object used to connect up *sections* and has no physical length.

A circuit is constructed on the *circuit canvas* where your editing is done.



You may create any number of *circuits* in a *project*.

Note that if the *Project Tree* is not visible, you can show it by clicking on the *Project tab* in the Main Window:



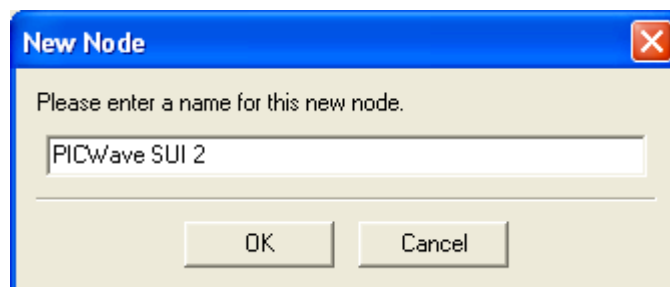
### 4.1 Creating a Circuit

➤ To create a new *circuit* in a *project*:

click on the  *project* in the *Project Tree*, then click on the  Add PICWave Circuit button

or right-click on the  *project* in the *Project Tree*, then select /Add/PICWave Circuit.

The *New Node dialog* will appear for you to enter the name of the new circuit:




➤ Type a name for the *circuit*, then click .

The new *circuit* will be shown in the *Project Tree* with the circuit symbol .




### 4.2 Showing Circuits

➤ To show a *circuit*:






double-click on the  *circuit* in the *Project Tree*

or right-click on the  *circuit* in the *Project Tree*, then select /Show





- or click on the  *circuit* in the *Project Tree*, then press **Return** or **Enter**.
- To hide a *circuit*:
  - click the close box  of the circuit window
  - or click on the circuit window to make it the front window, then press **Alt+F4**
  - or right-click on the  *device* in the *Project Tree*, then select **/Hide**.

## 4.3 Saving Circuits

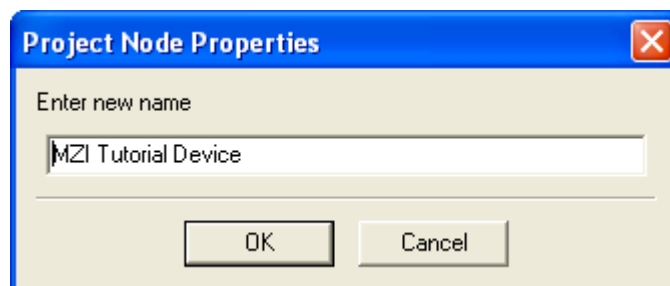
- To save a *device* to file:
  - select **File/Save Project** or **File/Save Project As...** from the *circuit* window
  - or press **Ctrl+S** from the *circuit* window
  - or right-click on the  *circuit* in the *Project Tree*, then select **/Save** or **/Save As...**
  - or click on the  *circuit* in the *Project Tree*, then press the  Save project button
  - or click on the  *circuit* in the *Project Tree*, then select **File/Save Project** or **File/Save Project As...**
  - or click on the  *circuit* in the *Project Tree*, then press **Ctrl+S**.


Note that this will save the entire project, not just the device.

## 4.4 Renaming Circuits




- To rename a *circuit*:
  - right-click on the  *circuit* in the *Project Tree*, then select **/Rename**
  - or click on the  *circuit* in the *Project Tree*, then press **Ctrl+R**.




The *Project Node Properties dialog* will appear for you to enter the new name for the *circuit*.



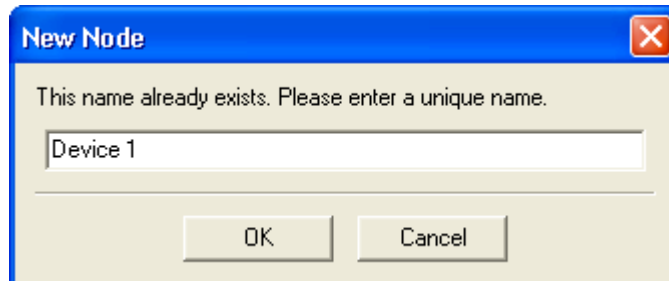
- Type the new name for the *circuit device*, then click .

## 4.5 Copying Circuits

- To copy a *circuit* to the clipboard:
  - right-click on the  *circuit* in the *Project Tree*, then select **/Copy**
  - or click on the  *circuit* in the *Project Tree*, then press **Ctrl+C**.
- To cut a *circuit* to the clipboard:
  - right-click on the  *circuit* in the *Project Tree*, then select **/Cut**


- or click on the  *circuit* in the *Project Tree*, then press **Ctrl+X**.
- To paste a *circuit* from the clipboard into the same project or another project:
  - right-click on the  *project* in the *Project Tree*, then select */Paste*
  - or click on the  *project* in the *Project Tree*, then press **Ctrl+V**.

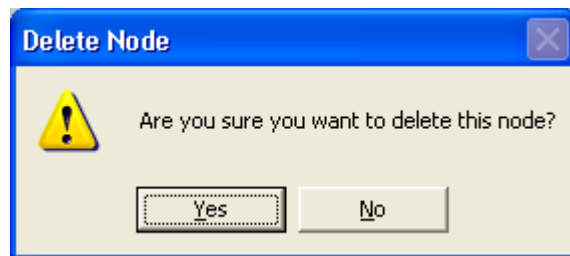
If there is already a *circuit* with the same name in the project, for example if you are making a copy of the *circuit* in the same project, the following dialog will appear:



- Type a unique name for the device, then click **OK**.

## 4.6 Deleting Circuits






- To delete a *circuit*, right-click on the  *circuit* in the *Project Tree*, then select */Delete*. The *Delete Node dialog* will appear:



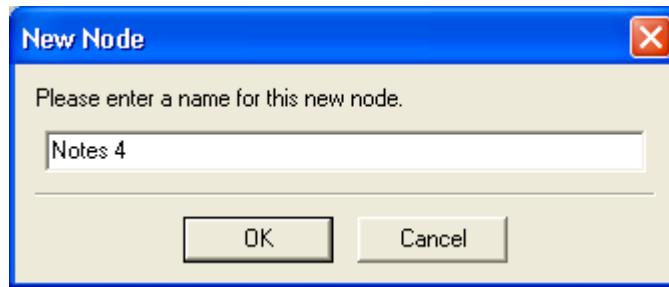
Click **Yes** to delete the *circuit*.


## 4.7 Adding Notes to a Project or Circuits

You can add *notes* to a *project* or *circuit* so that your documentation can be saved with your *project*.

- To add *notes* to a *project* or *circuit*:
  - click on the  *project* or  *circuit* in the *Project Tree*, then click on the  add notes button
  - or right-click on the  *project* or  *circuit* in the *Project Tree*, then select */Add/Notes*.

The *New Node dialog* will appear for you to enter the name for the new *notes*:




- Type a name for the *notes*, then click .

The new *notes* will be shown in the *Project Tree* with the notes symbol .

You can add as many different sets of notes to a *project* or *circuit* as you like.

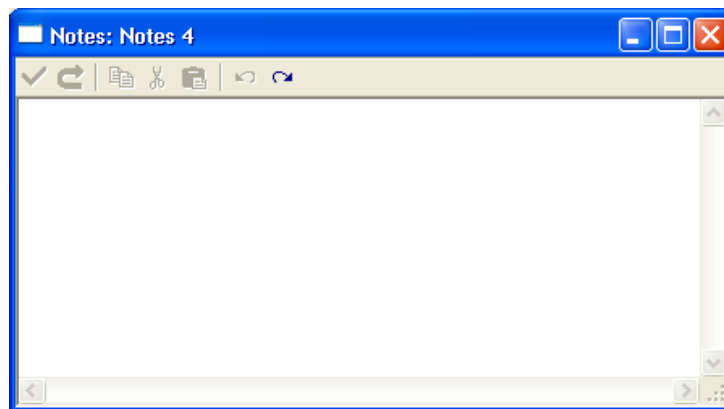
- To show a set of *notes*:

double-click on the  *notes* in the *Project Tree*








or right-click on the  *notes* in the *Project Tree*, then select **/Show**

or click on the  *notes* in the *Project Tree*, then press **Return** or **Enter**.


The *notes* will appear as follows:




You can type any text into the notes window to document the circuit or project to which the notes belong.

- Click on the  commit changes button to commit any changes you have made to the *notes*. Note that this does not save the notes to disk; any committed notes will be saved when you save the project (see §2.1.1.1).
- Click on the  revert button to discard any changes you have made to the *notes*. Note that the changes will be lost.
- Click on the  copy button to copy the highlighted text to the clipboard.
- Click on the  cut button to cut the highlighted text to the clipboard.
- Click on the  paste button to paste the text on the clipboard into the *notes*.
- Click on the  undo button to undo the last change you made to the text.
- Click on the  redo button to redo the last change you undid.
- To hide a set of *notes*:

click the close box  of the notes window

- or click on the notes window to make it the front window, then press **Alt+F4**
- or right-click on the  notes in the *Project Tree*, then select */Hide*.

## 4.8 Circuit Properties

- To change the properties of a *circuit*, show the *circuit* (see §4.2), then:
  - click on the  circuit properties button
  - or select **Edit/Device Properties...**
  - or right-click anywhere on the *circuit* and select */Device Properties...*

The *Circuit Properties dialog* will appear.

The *Circuit properties* are:

Parameter	Description	Units
Geometry		
<b>ZMin</b>	z-coordinate of the left edge of the device	
<b>ZMax</b>	z-coordinate of the right edge of the device	
<b>XMin</b>	x-coordinate of the bottom edge of the device	
<b>XMax</b>	x-coordinate of the top edge of the device	

These allow you to increase or decrease the area of you circuit canvas, for example to make room for more circuit elements.

## 4.9 Sections


A *section* is a circuit element with a physical length. This version of PICWAVE supports five types of *section* - *waveguide sections*, *Y-Junctions*, *directional couplers*, *Mach Zehnder Interferometers (MZI)* and *Finite Impulse Response Filter Sections (FIR sections)*. In previous versions, *Y-Junctions*, *directional couplers* and *MZIs* were (and still can be) simulated by equivalent circuits, the tutorial example (§2.1) being an example of the latter.

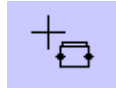
All *sections* can be *rotated*, as described in §3.1.7, by right-clicking on them and selecting */Rotate./Right, Left* or *180 degrees*. Rotation may be needed out of necessity (e.g. to join two signals together using a rotated Y-Junction) or just for layout convenience.

### 4.9.1 The Waveguide Section

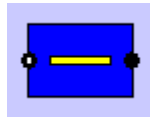
A *waveguide section* can be used for both active and passive waveguides. A waveguide can be defined either just by its (TE and/or TM) mode effective indices and group indices, assuming a single-mode model for each polarisation, or by a full physical description of its cross-section in the x-y plane (an *RWG*), which is uniform along the length of the waveguide in the z-direction and allows the waveguide to support multiple modes in each polarisation (see §5 on *RWGs* in general and §5.1.1 in particular on using *RWGs* to define the cross-section of a *waveguide section*).

To add a *waveguide section* to a *circuit*:

- Click on the  icon in PICWAVE's device window toolbar, or select `/Insert/WGSection` from the menu-bar. The cursor will change to:



- Now left-click on the circuit window where you want to drop your *waveguide section*. If you want to add more sections then left-click at additional positions. Press **Esc** when you have finished dropping new sections. Your new section should appear as:



The black circles on each end of the section are *ports*. A *port* is where an optical signal enters or exits a section.

When an *RWG* has been used to define the cross-section of a *waveguide section*, the *RWG* can be opened for editing from the device window by right-clicking on the *waveguide section* and selecting and `/Edit WG` (see §5.1.1).

The properties of a given *waveguide section* are edited in the *Waveguide Section Properties* panel (see §3.1.6 on how to access this). These properties are discussed below.

#### 4.9.1.1 Waveguide Section Properties – General

- sectionName** you can assign a name to the section here. The name will be used in various places to reference the section. Sections will be given a name automatically on creation.
- crossSectionDef** if you set this to *waveguide* then you need to assign a waveguide cross-section, such as an *RWG*, to the section (see §5). The waveguide section will then support *all* the modes of the *RWG*. If you set this to *effIndex*, then you do not need to give the section a physical cross-section but just define the properties of the optical mode in one or both of the polarisations – via the properties **modePol**, **effIndexTE** (**effIndexTM**) and **effGroupIndexTE** (**effGroupIndexTM**) - see below.
- modePol** (if **crossSectionDef** is set to *effIndex* only). Set this to the polarisation of the mode of the waveguide section: *TE-like*, *TM-like* or *Both* (see §4.18 for further details on polarisation settings). This setting will determine which effective indices and group indices need defining (see below for mode properties).
- sectionType** this can have one of three values, the last two are only available if **crossSectionDef** is set to *waveguide* - see above:
  - passive* – the waveguide section is a simple passive one.
  - gain* - (requires Active Module) the waveguide can have an active layer modelling a diode junction with injected carriers for generating optical gain. Gain sections are intended for forward biased diodes (see below for active properties).

*ea-modulator* – (requires Active Module) enables the electro-absorption modulator model. Ea-modulator sections are intended for reverse-biased diodes (see below for active properties).

<b>length</b>	[ $\mu\text{m}$ ] the <i>section length</i>
<b>Tsub</b>	[ $^{\circ}\text{C}$ ] the <i>substrate temperature</i> (only available if <b>sectionType</b> set to <i>gain/ea-modulator</i> )

#### 4.9.1.2 Active Properties

If you have the Active Module then you can add sections for modelling active media i.e. waveguides with injected carriers creating optical gain (forward biased diode) or for modelling an electro-absorption modulator (reverse biased diode). See §11.4 and 11.5 for details of the active models; §4.17 and 11.1 for further details about the properties and modelling of electrical contacts needed for active waveguide sections; and §10.1.1 and 10.1.3 for details on the active material settings and the gain model.

<b>sponBetaNA</b>	spontaneous coupling factor (geometric part only), see §17.4.
<b>ncellW</b>	the number of computation cells ( <i>CoreCells</i> ) in transverse direction. This defines how finely the lateral carrier distribution is resolved (see §11.2)
<b>numContacts</b>	number of electrical contacts on the section. An electrical contact is used for injecting current into the diode junction. Typically only one is needed.
<b>instruments</b>	this brings up the <i>Waveguide Section Instruments</i> panel from which you can add various <i>Junction Instruments</i> to the monitor – see §4.15.2.
<b>CurrentFlowModel</b>	a set of parameters defining the current flow model between the contact and the active layer(s). Currently the model consists simply of a resistive connection from the contact to each active layer. (see §11.1).

**Electro absorption Model** this provides access to the parameters defining the electro-absorption model. The model is currently defined by three functions:

- **escapeRateFunc** - the photo-generated carrier escape rate in (1/ns) – the time required for photo-carriers to reach the electrodes.
- **gainPeakFunc** – defines how the gain peak is modified by the applied reverse bias voltage.
- **gainPeakPosnFunc** – defines the shift in the gain peak with reverse bias voltage.

Each function can be edited using the *Function Editor* (see §14.4 for more information). See §11.5 for more details on the Electro-Absorption Model.

#### 4.9.1.3 Grating Properties

A waveguide section may have one or more *gratings*. For waveguides whose cross-sections are defined by an RWG, gratings may be physically defined by a *grating profile*

from a *grating set*, otherwise they are defined manually by their coupling coefficients. The waveguide section grating properties are:

<b>gratPhaseShift</b>	[degrees] an additional phase shift added to any grating
<b>CouplingParms</b>	<i>grating</i> coupling parameters editor, used to view/edit the coupling parameters of all gratings belonging to the section

For more information on *gratings*, *grating sets* and **CouplingParms**, see §6.

The *Bragg Wavelength Calculator* can be used to calculate required grating periods (see §14.1).

#### 4.9.1.4 Mode Properties

**numTEModes** (if **crossSectionDeff** set to *waveguide* only) number of active TE-like modes to propagate through the section; i.e. the first **numTEModes** TE modes will be propagated, where modes are in order of decreasing effective index (see §5.1.1 for details on *active modes*)

**numTMModes** as **numTEModes** but for the TM polarisation

The following can be set only if **crossSectionDeff** is set to *effIndex*:

**effIndexTE** set these to the effective (phase) indices of the TE mode of the waveguide section

**effIndexTM** as **effIndexTE** but for the TM polarisation

**effGroupIndexTE** set these to the effective (group) indices of the modes of the waveguide section

**effGroupIndexTM** as **effGroupIndexTE** but for the TM polarisation

**effLossTE** [1/cm] set this to the effective losses of the modes of the waveguide section

**effLossTM** as **effLossTE** but for the TM polarisation

#### 4.9.2 The Taper Section

The *taper section* is a waveguide section whose cross-section varies along its length. Two *RWGs* define the cross-section at each end of the taper section, and the cross-section in between is computed by interpolating the physical parameters of the two *RWGs* as a function of position,  $z$ , along the taper (see §5 on *RWGs*, and §5.1.1 in particular for how to use them to define the cross-section at each end of the taper). The current version of PICWAVE supports linear adiabatic tapers only.

Taper Model

Analytically, the cross-section as a function of position  $z$  along the length of a taper section is defined such that any parameter  $p_j$  describing it (e.g. a length or an alloy fraction) is given by the interpolation function:

$$P_j = P_{j,LHS} + (P_{j,RHS} - P_{j,LHS}) \times w(z)$$

where LHS and RHS denote the cross-sections at the left and right-hand ends of the taper, Lhs WG and Rhs WG, respectively; and  $w(z)$  is a weighting function that varies from  $w(z=0) = 0$  to  $w(z=L) = 1$ , where  $L$  is the length of the *taper section*.

PICWAVE models *taper sections* by discretising them into a user-defined number of sub-sections, each sub-section having a unique but constant cross-section, characterised by a weight. A sub-section's weight is given by the average of  $n(z)$  over its length. The cross-section is generated from interpolation function above, using the weight in place of  $n(z)$ .


PICWAVE divides *taper sections* into sub-sections such that: each sub-section contains an integer number of z-elements; the z-elements are distributed as equally as possible between the sub-sections; the group velocity error over the whole *taper section* is no more than that introduced by its division into z-elements (see §11.2 on z-elements and group velocity error). The maximum number of sub-sections is automatically limited to the number of z-elements. The z-element length,  $zStep$ , may vary from one sub-section to the next, reflecting the corresponding change in group velocity.

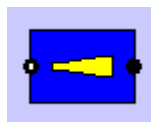
In the current version of PICWAVE  $n(z)$  is linear; therefore only linear taper sections can be modelled i.e. all cross-section parameters (dimensions etc) vary linearly with the position  $z$  along the taper. Taper sections are also assumed to be adiabatic.

#### Taper Section Usage

The user need only define the cross-section at each end face of the *taper section* (Lhs WG and Rhs WG), and the number of sub-sections into which the taper is to be divided. Lhs and Rhs always denote the ends of the taper which correspond to the left and right-hand ends in the *default* orientation.

Taper generation, i.e. their division into sub-sections and the generation of their cross-sections, is automatic. For a given *taper section*, it happens at the start of the first TDTW simulation in which the taper section is used. Taper sections are generated anew only when parameters determining their division into sub-sections are changed. These parameters are: the parameters defining the RWGs (Lhs WG and Rhs WG) and their *active modes*; the number of sub-sections (**numSubSections**); the taper section length (**length**); and the TDTW simulation **zStep**. See §4.18 for details on *active modes* with regard to *taper sections*.

A *taper section* is added to a circuit in the same manner as a waveguide section, using the  icon or **/Insert/Taper Section** from the menu-bar. It should appear as:



The waveguide cross-sections Lhs WG and Rhs WG are defined by RWGs as detailed in §5.1.1. The RWGs must be topologically equivalent i.e. have the same number and arrangement of layers, slices etc.

The properties of a given *taper section* are edited in the *Taper Section Properties* panel (see §3.1.6). These properties are discussed below. When a TDTW simulation is paused or completed, a taper's modal data and cross-sections can be inspected by right clicking on the taper and selecting **/Z-profiles/Mode Data** (see §9.9.1) and **/Edit WG at...** (see §5.1.1), respectively. This allows one to see how the taper has been divided up into sub-sections. (Note that the cross-section of the first and last sub-sections will not in general be identical to Lhs WG and Rhs WG, which define the cross-section at the end *faces* of the taper).

If the **zStep** in the TDTW Calculator is defined using a taper section as the *reference section* (see §9.1.2), the **tStep** of the simulation is calculated using the group velocity in the Lhs WG. This **tStep** is then used to determine the value of  $zStep$  within each of the



taper's sub-sections in the same way as it used to determine the value of  $zStep$  for the other sections in the device.

#### Taper Section Properties


All but one of the *taper section* properties are shared by the waveguide section. The only property unique to the *taper section* is:

**numSubSections** the number of sub-sections into which the taper will be divided (provided it does not exceed the number of z-elements - see above)

### 4.9.3 The Y-Junction

The *Y-Junction* is a section that splits a waveguide into two, or that, after rotation (see §4.9 introduction), joins two waveguides into one. It splits the power between the two waveguides according to a user-specified coupling coefficient. This takes place over a user-specified section length, in contrast to the *power splitter* (§4.10.2), in which the splitting occurs over zero length (no optical time delay).

The *Y-Junction* supports up to two modes, one in each polarisation, in the same manner as an *effIndex* waveguide section.

A *Y-Junction* is added to a circuit in the same manner as a waveguide section, using the  icon or `/Insert/Y-Junction` from the menu-bar. It should appear as:



The properties of a given *Y-Junction* are edited in the *Y-Junction Properties* panel (see §3.1.6). These properties are discussed below.

#### Y-Junction Properties

All but one of the *Y-Junction* properties are shared by the waveguide section, namely **sectionName**, **length**, **modePol**, **effIndexTE** (**effIndexTM**), **effGroupIndexTE** (**effGroupIndexTM**) and **effLossTE** (**effLossTM**) (see §4.9.1.1 for a description of these properties). The only property unique to the *Y-Junction* is:

**couplingCoeff** the fraction of the power to be coupled from the LHS port to the *upper* RHS port in the default orientation (before losses)

### 4.9.4 The Directional Coupler

The *directional coupler* is a section modelling two parallel waveguides, which are coupled so that a fraction of the input power of one guide couples to the other guide for every unit length travelled. This fraction is a user-specified parameter and may be polarisation dependent. Coupling takes place over the user-specified section length, in contrast to the *power coupler* (§4.10.3), for which the coupling occurs in zero length (no optical time delay). . In PICWAVE, the *directional coupler* is defined to be symmetric.

#### The Model


If A and B are the amplitudes of the modes travelling forward in guide 1 and guide 2 and the coupling coefficient is K then the coupling is defined as:

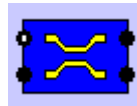
$$dA = j.K.B.dz$$

$$dB = j.K.A.dz$$

This is integrated along the length of the coupler. The *directional coupler* supports up to two modes, one in each polarisation, in the same manner as an *effIndex* waveguide section. TE-like modes do not couple to TM-like and vice versa.

Usage

A *directional coupler* is added to a circuit in the same manner as a waveguide, using the  icon or `/Insert/Directional Coupler` from the menu-bar. It should appear as:



The properties of a given *directional coupler* are edited in the *Directional Coupler Properties* panel (see §3.1.6). These properties are discussed below.

Properties


All but two of the *directional coupler* properties are shared by the waveguide section, namely **sectionName**, **length**, **modePol**, **effIndexTE** (**effIndexTM**), **effGroupIndexTE** (**effGroupIndexTM**) and **effLossTE** (**effLossTM**) (see §4.9.1.1 for a description of these properties). The only properties unique to the *directional coupler* are:

- coeffKappaTE** [1/μm] as defined by K in the coupling equations above, this is the coupling coefficient which specifies the coupling between the TE mode in one guide and the TE mode in the other.
- coeffKappaTM** as **coeffKappaTE** but for the TM polarisation

#### 4.9.5 The Mach Zehnder Interferometer (MZI)

A *Mach Zehnder Interferometer (MZI)* splits an input signal equally between two arms with different optical path lengths and rejoins them into one output signal for the purpose of achieving interference effects. *MZIs* can be easily constructed from individual parts in PICWAVE, as in the tutorial (§2.1), but can also be conveniently inserted as a complete single section. Circuits can thus be constructed more quickly and have a much smaller and simpler visual layout in the device window.

An *MZI* supports up to two modes, one in each polarisation, in the same manner as an *effIndex* waveguide section.

An *MZI* is added to a circuit in the same manner as a waveguide section (see §4.9.1), using the  icon or `/Insert/MZI` from the menu-bar. It should appear as:



The properties of a given *MZI* are edited in the *Mach Zehnder Interferometer Properties* panel (see §3.1.6 on how to access this). These properties are discussed below.

Properties

All but two of the *MZI* properties are shared by the waveguide section, namely **sectionName**, **length**, **modePol**, **effIndexTE** (**effIndexTM**), **effGroupIndexTE** (**effGroupIndexTM**) and **effLossTE** (**effLossTM**) (see §4.9.1.1 for a description of these properties). The only properties unique to the *directional coupler* are:

<b>arm1Length</b>	[ $\mu\text{m}$ ] the length of the upper arm of the MZI
<b>deltaLength</b>	[ $\mu\text{m}$ ] the difference by which the length of the lower arm of the MZI exceeds that of the upper arm

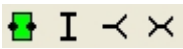

#### 4.9.6 The Finite Impulse Response Filter Section (FIR Section)

A *Finite Impulse Response Filter Section (FIR section)* is a section that can model a device with an arbitrary spectral profile. It is different from the other sections described above in that it does not have a physical size defined by the user, but instead provides a way of importing a frequency-domain model of a component from FIMMPROP or another compatible product. The FIR section and its use is fully treated of in §6.

## 4.10 Joins

The *ports* on adjacent sections can either be connected (see §4.11) directly, in which case the signal is fully transmitted from one section to the next without reflection, or connected explicitly with a *join*. *Joins* are elements that have no physical shape or length but which allow one to define the coupling (reflection/transmission/loss) and splitting of an optical signal between sections in an abstract manner. In general, *joins* by themselves do not correspond to physical structures. The exception to this is the 2-port or *facet* join, which can correspond to a waveguide facet so that you can define a facet reflectivity. As well as *facet* joins, you can construct 3-port *joins* (power splitter), 4-port *joins* (power coupler) and general joins with an arbitrary number of ports – see below.

Usage

To add a join to your circuit, click on one of the join icons:  in the toolbar, or select one of the joins listed in the **/Insert** menu from the menu-bar, then move your cursor to the position where you would like your join and left-click to place it. Press **Esc** or click the  icon when you have finished placing joins.

All *joins* can be *rotated*, as described in §3.1.7, by right-clicking on them and selecting **/Rotate..Right, Left** or **180 degrees**. Rotation may be needed out of necessity (e.g. to join two signals together using a rotated power splitter) or just for layout convenience.

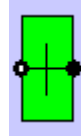
The join properties panel for all joins is accessed by right-clicking on the join and selecting **/Properties**.

The coupling for all joins can be defined by a *general scattering matrix (S-Matrix)*. For further information on using join *S-Matrices*, see §4.10.5.

#### 4.10.1 The Facet Join

A waveguide facet is defined by a *facet* join, where the reflection/transmission/loss can be set with varying degrees of generality for each polarisation; *by default* they are non-reflecting. The modes traversing a *facet* join are determined by the sections attached to it, therefore a *facet* join must have *at least one* section attached to it. All sections attached to a *facet* must have the same number of modes in each polarisation; coupling can only occur between corresponding modes in each attached section (see §4.18), and the specified coupling coefficients are the same for all modes of a given polarisation.

A *facet* join has two ports and is shown on the circuit canvas as:



The properties of a given *facet* join are edited in the *Facet Properties* panel, they are as follows:

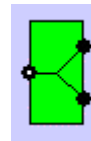
Facet properties

<b>isGeneral</b>	if <i>true</i> then you need to explicitly specify the scattering matrices (see §4.10.5) of the <i>facet</i> via the <b>genSMatTE</b> ( <b>genSMatTM</b> ) properties - see below. Otherwise you can define various specialisations as outlined below.
<b>type</b>	if <b>isGeneral</b> is set to <i>false</i> , a <i>facet</i> join can be one of:  <i>Lossless</i> , - you need only specify the reflection coefficient(s) <b>reflCoeffTE</b> ( <b>reflCoeffTM</b> ) - see below; the transmission coefficient(s) is computed automatically.  <i>Lossy</i> - you must define the reflection coefficient(s) <b>reflCoeffTE</b> ( <b>reflCoeffTM</b> ) and the insertion loss(es) <b>insLossTE</b> ( <b>insLossTM</b> ) – see below; the transmission coefficient(s) is computed from these parameters.  <i>Defined</i> - this allows you to independently define the left-left reflection(s) <b>rLLTE</b> ( <b>rLLTM</b> ), right-right reflection(s) <b>rRRTE</b> ( <b>rRRTM</b> ) and transmission coefficient(s) <b>tLRTE</b> ( <b>tLRM</b> ) – see below. It is assumed that <b>tRLTE</b> is equal to <b>tLRTE</b> etc.  <i>Natural</i> – this defines a natural joint where the reflection/transmission is calculated from the effective (phase) indices of the adjoining sections.
<b>reflCoeffTE</b>	reflection coefficient for <i>Lossless</i> , and <i>Lossy</i> models
<b>reflCoeffTM</b>	as <b>reflCoeffTE</b> but for the TM polarisation
<b>insLossTE</b>	insertion loss for <i>Lossy</i> model
<b>insLossTM</b>	as <b>insLossTE</b> but for the TM polarisation
<b>tLRTE</b>	TE transmission coefficient
<b>tLRM</b>	as <b>tLRTE</b> but for the TM polarisation
<b>rLLTE</b>	left-left TE reflection
<b>rLLTM</b>	as <b>rLLTE</b> but for the TM polarisation
<b>rRRTE</b>	right-right TE reflection
<b>rRRTM</b>	as <b>rRRTE</b> but for the TM polarisation
<b>genSMatTE</b>	if <b>isGeneral</b> = <i>True</i> , then the TE scattering matrix is specified here – see §4.10.5 for details
<b>genSMatTM</b>	as <b>genSMatTE</b> but for the TM polarisation

#### 4.10.2 The Power Splitter Join

The *power splitter* is a join that splits the power in one waveguide on the left (or right) into two waveguides on the right (or left). In this it is like the Y-Junction (§4.9.3), except that splitting occurs over zero length (i.e. there is no optical time delay). It splits the input power between the two output signals according to a user-specified coupling coefficient, which can be set for each polarisation. *By default* it splits the power equally between the two output waveguides. The other settings for the *power splitter* can be set with varying degrees of generality. The modes traversing a *power splitter* are determined by the sections attached to it, therefore a *power splitter* must have *at least one* section attached to it. All sections attached to a *power splitter* must have the same number modes in each polarisation; coupling can only occur between corresponding modes in each attached section (see §4.18), and the specified coupling coefficients are the same for all modes of a given polarisation.

The *power splitter* has three ports and is shown on the circuit canvas as:



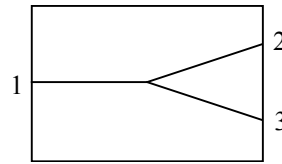
The properties of a given *power splitter* are edited in the *Power Splitter Properties* panel, they are as follows:

##### Power Splitter Properties

<b>isGeneral</b>	if <i>true</i> then you need to explicitly specify the scattering matrices (see §4.10.5) of the <i>power splitter</i> via the <b>genSMatTE</b> ( <b>genSMatTM</b> ) properties - see below. Otherwise you can define various specialisations as outlined below.
<b>isSymmetric</b>	if true, then coupling to both branches is equal. Otherwise you need to specify <b>coupCoeffL1R1TE</b> ( <b>coupCoeffL1R1TM</b> ) – see below
<b>isLossy</b>	if false, then the join is assumed to be lossless. Otherwise you need to specify the insertion loss via <b>insLossTE</b> ( <b>insLossTM</b> ) – see below.
<b>isReflecting</b>	if false, then the join is non-reflecting, i.e. no light is reflected back into the same port. Otherwise you need to specify <b>reflCoeffTE</b> ( <b>reflCoeffTM</b> ), and <b>reflCoupCoeffTE</b> ( <b>reflCoupCoeffTM</b> )– see below.
<b>coupCoeffL1R1TE</b>	TE coupling coefficient between the LHS port (L1) and the <i>upper</i> RHS port (R1), in the default orientation. Not used if the splitter is symmetric.
<b>coupCoeffL1R1TM</b>	as <b>coupCoeffL1R1TE</b> but for the TM polarisation
<b>insLossTE</b>	TE insertion loss – i.e. the fraction of power entering an input port that does not exit any port. Ignored if splitter is lossless.
<b>insLossTM</b>	as <b>insLossTE</b> but for the TM polarisation
<b>reflCoeffTE</b>	the TE reflection coefficient seen by the trunk waveguide (same for each port).
<b>reflCoeffTM</b>	as <b>reflCoeffTE</b> but for the TM polarisation

<b>reflCoupCoeffTE</b>	of the power entering one of the ports belonging to the two branches, this is the fraction of the reflected TE power (reflected according to <b>reflCoeffTE</b> ) that is coupled back <i>into</i> the port. Ignored if the splitter is not reflecting.
<b>reflCoupCoeffTM</b>	as <b>reflCoupCoeffTE</b> but for the TM polarisation
<b>genSMatTE</b>	if <b>isGeneral</b> = <i>True</i> , then the scattering matrices for each polarisation are specified here – see §4.10.5 for details
<b>genSMatTM</b>	as <b>genSMatTE</b> but for the TM polarisation

If we enumerate the ports of the join as follows:



where ports L1, R1 and R2 (see **coupCoeffL1R1TE/coupCoeffL1R1TM** definition above) are re-labelled as ports 1, 2 and 3, then the scattering matrix of the *power splitter* is defined as:

$$\tilde{S} = \begin{pmatrix} -rc - 2jrd\alpha\beta & acs + j\beta ds & jad s + \beta cs \\ acs + j\beta ds & cr & jdr \\ jad s + \beta cs & jrd & cr \end{pmatrix}$$

where  $c = \sqrt{T_{12}}$ ,  $d = \sqrt{1 - T_{12}}$ ,  $r = \sqrt{R}$ ,  $s = \sqrt{1 - R}$ ,  $\alpha^2 = \frac{T_{12} + R - 1}{2R - 1}$ ,  $\beta^2 = 1 - \alpha^2$

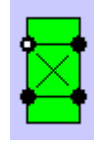
-  $T_{12}$  and  $R$  being the user-specified parameters, **coupCoeffL1R1TE/TM** and **reflCoeffTE/TM**, respectively. The output  $b_i$  of the port  $i$  is given by  $b_i = \sum_j \tilde{S}_{ij} a_j$ , where the input of

the port  $j$  is  $a_j$ . Note that if the join is non-reflecting then  $r$  is zero; that if the join is symmetric then  $\alpha = \beta$ . See §4.10.5 below for a general discussion of join scattering matrices.

#### 4.10.3 The Power Coupler Join

The *power coupler* acts like a *directional coupler* (§4.9.4), coupling a fraction of the input power in one waveguide to the other waveguide, according to a user-specified coupling coefficient that can be set separately for each polarisation. However, since it is a *join*, the coupling takes place over zero length (i.e. there is no optical time delay). In PICWAVE, the *power coupler* is defined to be symmetric, so that the coupling between any pair of ports is the same as that between the remaining pair. *By default* a *power coupler* is 50:50, i.e. half the input power in one waveguide is cross-coupled to the other waveguide. The modes traversing a *power coupler* are determined by the sections attached to it, therefore a *power coupler* must have *at least one* section attached to it. All sections attached to a *power coupler* must have the same number of modes in each polarisation; coupling can only occur between corresponding modes in each attached section (see §4.18), and the specified coupling coefficients are the same for all modes of a given polarisation.

The *power coupler* has four ports and is shown on the circuit canvas as:



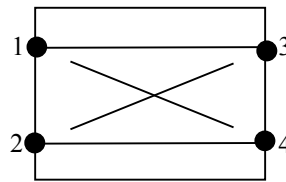
The properties of a given *power coupler* are edited in the *Power Coupler Properties* panel, they are as follows:

#### Power Coupler Properties

The properties of the *power coupler* are similar to those of the power splitter (see §4.10.2). We note here only the properties which are (or have a definition) specific to the *power coupler*:

- coupCoeffL1R2TE** coupling coefficient between the *upper* LHS port (L1) and the *lower* RHS port (R2), in the default orientation. The other cross-coupling coefficient is the same by virtue of symmetry
- coupCoeffL1R2TM** as **coupCoeffL1R2TE** but for the TM polarisation
- reflCoeffTE** the fraction of the power entering a port that is reflected back into the same port
- reflCoeffTM** as **reflCoeffTE** but for the TM polarisation

If we enumerate the ports of the join as follows:



where ports L1, L2, R1 and R2 (see **coupCoeffL1R2TE/coupCoeffL1R2TM** definition above) are re-labelled as ports 1, 2, 3 and 4, then the scattering matrix of the *power coupler* is defined as:

$$S = \begin{pmatrix} -cr & -jdr & cs & jds \\ -jdr & -cr & jds & cs \\ cs & jds & cr & jdr \\ jds & cs & jdr & cr \end{pmatrix}$$

where  $c$ ,  $d$ ,  $r$  and  $s$  are as defined in §4.10.2, with the exception that  $T_{12}$  is replaced by  $T_{13}$ , such that  $1 - T_{13}$  corresponds to the cross-coupling coefficient **coupCoeffL1R2TE/coupCoeffL1R2TM**. The output  $b_i$  of the port  $i$  is given by  $b_i = \sum_j S_{ij} a_j$ ,

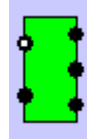
where the input of the port  $j$  is  $a_j$ . See §4.10.5 below for a general discussion of join scattering matrices.

#### 4.10.4 The General Join

The *general join* is a generalisation of the types of join discussed above: it has an arbitrary number of left and right ports, and the coupling is always specified by a general scattering matrix (see §4.18). Furthermore, unlike the other joins, the *general join* allows the sections attached to its ports to have different numbers/polarisations of modes;

the coupling between any and all of the modes of the attached sections can be specified. The use of general joins thus allows multiple modes and both polarisations to be supported throughout the *whole* circuit. As the modes traversing a given port on a *general join* are determined by the section attached to it, all ports on a *general join* must be attached to a section. This means that *optical sources* cannot be connected to the ports on a *general join*.

On the circuit canvas, a *general join* may look like:



The properties of a given *general join* are edited in the *General Join Properties* panel, accessed by right-clicking on the join and selecting */Properties*. These properties are discussed below.

#### General Join Properties

<b>numLHSPorts</b>	number of ports on the left hand side of the join
<b>numRHSPorts</b>	number of ports on the right hand side of the join
<b>genSMat</b>	here the scattering matrix can be defined (see §4.10.5 below)

#### 4.10.5 Join Scattering Matrices

Join scattering matrices are an alternative way of specifying the coupling coefficients of the *facet*, *power splitter* and *power coupler* joins, and the only way in which the coupling coefficients of a general join can be defined.

To edit the *general scattering matrix* (*S-Matrix*) of a given join:

- Double click on the **genSMat** or **genSMatTE/TM** property in the join properties panel. This will open the *General Scattering Matrix editor*, which appears as follows:

```

begin <McSMatrix(1.1)> "SMatrix"
  3 // numModePorts
  // coefficients {real imag} - one row per input mode
  1 { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 }
  2 { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 }
  3 { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 }
end
  
```

This editor works in the same way as the notes editor (see §4.7).

The format of the text specifying an *S-Matrix* should always appear as above, if the format is not correct PICWAVE will not be able to load the *S-Matrix*. Except in the case of a *general join*, PICWAVE will provide an *S-Matrix* template of the correct form and the user need only input the complex matrix elements themselves, specified in the {} brackets. The *general join* may require the user to add in extra rows or columns and specify the dimension (numModePorts) of the matrix accordingly.

The columns of an *S-Matrix* correspond to the output modes of each port of a given join, and the rows correspond to the inputs – the modes being determined by the attached components. The coupling coefficients are therefore specified by the



complex matrix elements {real, imag}. The corresponding power coupling coefficients are given by the mod-squares of the matrix elements, so that the sum of the mod-squares of the elements on a given row/column must always be less than or equal to one. The port numbering convention used to label the rows and columns *always* applies to the join as it would appear in the default orientation. Care should be taken when using rotated joins – the white dot on the upper left port in the default orientation allows the relative orientation of any rotated join to be determined i.e. the white dot always indicates Port 1.

S-Matrices are always necessarily square, as any input can also be an output. The precise meaning of the rows and columns depends on the type of join whose *S-Matrix* is being edited. There are two different cases, which are described below.

#### Facet, Power Splitter and Power coupler S-Matrices

The *facet*, *power splitter* and *power coupler* joins can couple only *corresponding* modes of attached waveguides (e.g. the second TE mode of one waveguide can be coupled only to the second TE mode of another); the coupling can be specified separately for each polarisation and is the same for *all* modes within a given polarisation. Therefore, up to two S-Matrices need to be defined: the *TE S-Matrix* and the *TM S-Matrix* (set in the **genSMatTE** and **genSMatTM** properties, respectively). The TE (TM) S-Matrix specifies the coupling between the TE (TM) modes entering each and every port of the join and the corresponding TE (TM) modes exiting each and every port – the modes themselves being determined by the components attached to the ports. Each row/column corresponds to one of the *ports* on the join. The format of the matrices is as follows:

		Output		
		Port 1	· · ·	Port <i>N</i>
Input	Port 1	{ <i>Re</i> , <i>Im</i> }	{ <i>Re</i> , <i>Im</i> }	{ <i>Re</i> , <i>Im</i> }
	⋮	{ <i>Re</i> , <i>Im</i> }	{ <i>Re</i> , <i>Im</i> }	{ <i>Re</i> , <i>Im</i> }
	Port <i>N</i>	{ <i>Re</i> , <i>Im</i> }	{ <i>Re</i> , <i>Im</i> }	{ <i>Re</i> , <i>Im</i> }

where *N* is the number of ports on the join. For details on the port numbering conventions and the definitions of each matrix element in terms of the power coupling coefficients, see the respective join sections (§4.10.1, 4.10.2 and 4.10.3).

Example: a power splitter that reflects half the TE (TM) power entering port 1 and couples half to port 2 has, as one possibility, the following TE (TM) S-Matrix:

		Output		
		Port 1	Port 2	Port 3
Input	Port 1	{ $1/\sqrt{2}$ , 0}	{ $1/\sqrt{2}$ , 0}	0
	Port 2	0	0	0
	Port 3	0	0	0

which would be written in the editor as:

```

General Scattering Matrix
begin <McSMatrix(1.1)> "SMatrix"
3 // numModePorts
// coefficients {real imag} - one row per input mode
1 { 0.707106781 0.000000000} { 0.707106781 0.000000000} { 0.000000000 0.000000000}
2 { 0.000000000 0.000000000} { 0.000000000 0.000000000} { 0.000000000 0.000000000}
3 { 0.000000000 0.000000000} { 0.000000000 0.000000000} { 0.000000000 0.000000000}
end

```

General Join S-Matrices

The *general join* can couple any and all of the modes of the attached waveguides. Therefore, only one *S-Matrix* need be defined, which gives a complete description of the coupling between each mode entering each port and each mode exiting each port - the modes themselves being determined by the components attached to the ports. Each row/column corresponds to one *mode* in one of the ports. The format of the general join *S-Matrix* is as follows:

		Output								
		Port 1			...			Port N		
		Mode 1	...	Mode n <sub>1</sub>	Mode 1	...	Mode n	Mode 1	...	Mode n <sub>N</sub>
Port 1	Mode 1	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}
	⋮	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}
	Mode n <sub>1</sub>	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}
Input	Mode 1	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}
	⋮	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}
	Mode n	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}
Port N	Mode 1	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}
	⋮	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}
	Mode n <sub>N</sub>	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}	{Re, Im}

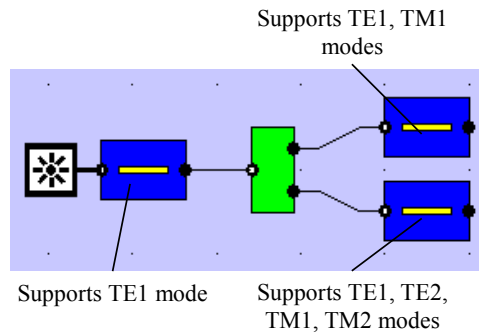
where the *i*th port has *n<sub>i</sub>* modes.

Following the convention of the other joins, ports are numbered from the top left down to the bottom left, then from the top right down to the bottom right.

Modes are listed first in order of their polarisation (TM listed after TE modes) and then in the order of decreasing effective index (see §5.6.2). For example, if a *general join* port *i* is attached to a waveguide supporting 6 TM modes, then the third mode (TM3) is simply the TM mode with the third highest effective index, as the ordered mode list is TM1, TM2, **TM3**, TM4, TM5, TM6. However, if the waveguide supports 6 modes of *both* polarisations (3 of each) then the third mode is TE3, as the ordered mode list is

TE1, TE2, **TE3**, TM1, TM2, TM3. An unused *general join* port takes up one row (column) in the matrix layout.

Example: for the following set up,



a general join *S-Matrix* which reflects half of the TE1 input power of the left waveguide and couples a third of the input power into the corresponding mode (TE1) of the upper right waveguide and a sixth to the lower could, as one possibility, be:

		Output							
		Port 1	Port 2		Port 3				
Input	Port 1	TE1	TE1	TM1	TE1	TE2	TM1	TM2	
		TE1	$\{1/\sqrt{2}, 0\}$	$\{1/\sqrt{3}, 0\}$	0	$\{1/\sqrt{6}, 0\}$	0	0	0
		TE2	0	0	0	0	0	0	0
		TM1	0	0	0	0	0	0	0
		TE1	0	0	0	0	0	0	0
		TE2	0	0	0	0	0	0	0
		TM1	0	0	0	0	0	0	0
	TM2	0	0	0	0	0	0	0	

which would be written in the editor as (trivial columns 5 – 7 not shown):

```

General Scattering Matrix
begin <McSMatrix(1.1)> "SMatrix"
7 // numModePorts
// coefficients {real imag} - one row per input mode
1 { 0.707106781 0.000000000 } { 0.577350269 0.000000000 } { 0.000000000 0.000000000 } { 0.408248290 0.000000000 } { 0.0
2 { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.0
3 { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.0
4 { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.0
5 { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.0
6 { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.0
7 { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.000000000 0.000000000 } { 0.0
end
  
```

For the general join, PICWAVE does not determine the template of the scattering matrix for the user; the user must specify an S-Matrix of the correct format by modifying the default. Also note that the number in the top left corner of the editor must be set by the user to the dimension of the matrix ( =  $\sum_{ports} \text{modes per port}$  ), which in this case is 7 (1 mode in Port 1 + 2 modes in Port 2 + 4 modes in Port 4).

#### Pasting S-Matrices

By highlighting and copying the *entire* text in the *General Scattering Matrix editor* or by copying *S-Matrix* text of the correct format to the clipboard from elsewhere (say, from a

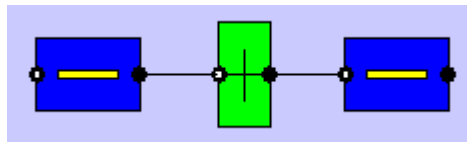
text file), you can then set the *S-Matrix* of a join by right-clicking on it and selecting */Paste S-Matrix* - for a general join - or */Paste TE S-Matrix* (*/Paste TM S-Matrix*) – for the other types of join. If the copied S-Matrix text does not have the correct signature at the top (<McSMatrix(1.1)>), the paste S-Matrix option(s) will not be enabled.

## 4.11 Connecting up a circuit

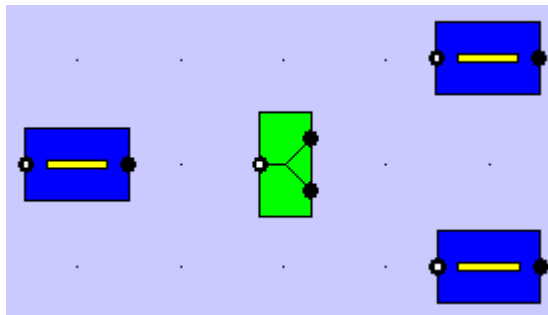
Once you have some *sections* and *joins* on your *circuit canvas* you need to connect them up. This is done with the *link* tool. Unlike previous versions, you can either connect two sections directly, where appropriate (i.e. when reflection/loss are negligible and need not be defined):



or, when you want to specify the coupling between sections, you can add a *facet* join in between the *sections* you want to connect:



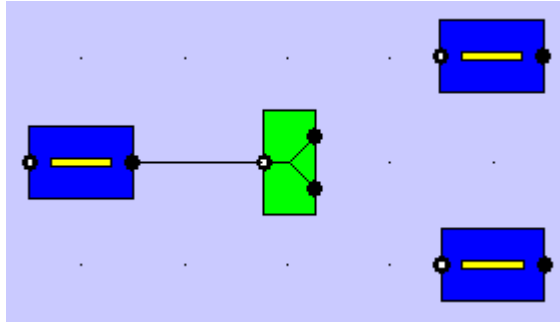
Let us assume you have a few sections and joins already on your canvas like this:



➤ Now click on the  icon to activate the *link* tool. The cursor will display:



- Move the cursor circle until it is on top of the right-hand port of the left *section*. Click your left mouse button. The cursor should now display 2 instead of 1 indicating that it is ready to move to the second port.
- Move the cursor circle until it is on top of the left-hand port of the *power splitter*. Click your left mouse button. You should now see:




Connect up the rest of your circuit in a similar fashion.

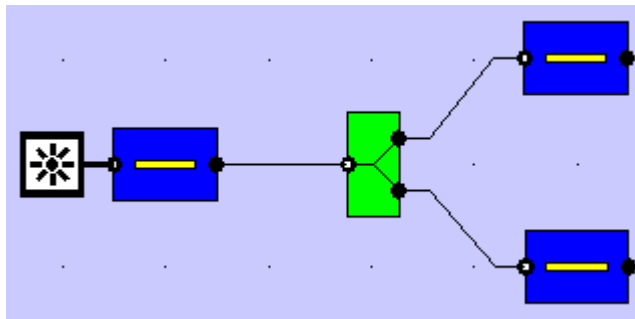
## 4.12 Adding an optical source

Unless your circuit includes a physical source such as a laser diode or LED, then you will want to add one or more *optical sources* to the circuit. An *optical source* can have an arbitrary time evolving power and arbitrary time-evolving wavelength. If you want to just find the frequency/wavelength response of your circuit then you can easily inject an optical impulse which has a finite amplitude for just one time step – and thus has a *flat* spectrum over the full free spectral range of your TDTW calculation.

To add an *optical source*:

- Click on the  icon in the tool bar. The cursor will change to a circle.
- Move your cursor until the circle is over the desired *port* and left-click to attach the source to the *port*. (note that *optical source* signals cannot be injected into *general joins*)

You should now see something like:



The *optical source* defaults to an impulse so if you want to just determine the spectral response of your circuit, you have nothing further to do. Otherwise you will want to set it up according to the parameters listed here, which are set in the *Optical Source Properties* panel, accessed by right-clicking on the source and selecting *Properties*.

### Optical Source Properties

- optoSourceName** you can enter a convenient name for the *optical source* here. The name will be used in various other places to identify the source.
- signal** here you define the power-versus-time and wavelength-versus time functions of the *optical source*. Click the property to edit. See §4.16 for details on how to define your signal in the *Signal Editor*.
- powerInstrument** this is the *Power Instrument* which measures the optical power exiting the source (see §4.15.3). Double-click the property to open the *Optical Power Instrument Properties* panel.

<b>wavelengthInstrument</b>	this is the <i>Wavelength Instrument</i> which measures the optical wavelength of the source (see §4.15.3). Double-click the property to open the <i>Optical Wavelength Instrument Properties</i> panel.
<b>modePol</b>	set this to the polarisation of the mode to be excited by the <i>optical source</i> : <i>TE-like</i> or <i>TM-like</i>
<b>ModeIndex</b>	the index of the mode (i.e. the position of the mode in the ordered list of modes of polarisation <b>modePol</b> ) to be excited by the optical source

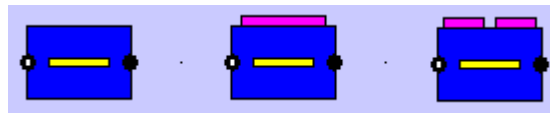
## 4.13 Adding an electrical drive

If your circuit contains a waveguide with an active layer(s) which you wish to use as an active waveguide (a gain section or an *ea* modulator), then you will need to attach an electrical drive (a *current drive* or *voltage drive*) to it. A *current drive* (*voltage drive*) can have an arbitrary time evolving current (voltage) signal.

An electrical drive is attached to a waveguide section with an electrical contact. To add an electrical contact to a waveguide:

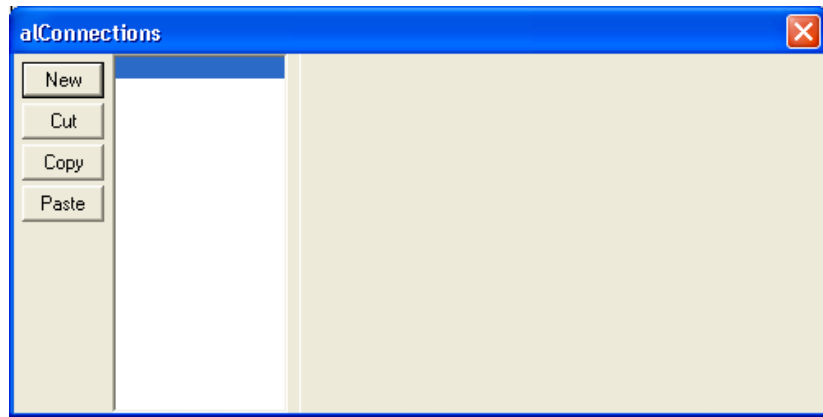
- Open *Waveguide Section Properties* panel for the particular waveguide (see §4.9.1)
- Ensure that the waveguide is active (i.e. **sectionType** is set to *gain* or *ea-modulator*). When this is done, the **numContacts** parameter will become available – this specifies the number of contacts on the waveguide
- Set **numContacts** to 1 or more, as required (only 1 is usually necessary)

On closing the *Waveguide Section Properties* panel, the specified number of contacts should appear (as pink rectangles) on the side of the waveguide. For example, waveguides with zero, one and two contacts will appear as follows:



The connection between an electrical contact and *each* active layer must also be specified. To do this:

- Double-click on the **CurrentFlowModel** property in the *Waveguide Section Properties* panel. The *CurrentFlowModel* panel will appear. The present model in PICWAVE is very simple, consisting just of a resistance between the contact and each active layer, where each active layer represents a (parallel) current path to the substrate. (For more information on the *Current Flow Model*, see §11.1.1 and 11.1.2.)
- Double-click on the **alConnections** property. The following panel will appear:



- Click **New** and set the **resistivity** of the new connection (named Element 1) to the desired value.

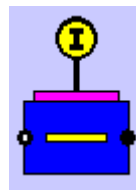
If there is more than one active layer in your waveguide, add as many connections as there are active layers. In the list of connections, Element X is the connection between the contact and active layer X, where active layers are numbered from the top down (i.e. uppermost active layer in an RWG = 1, next lowest = 2 etc)

This simple set-up is now sufficient for the attaching of an electrical drive. Further discussion of contacts and details of how the modelling of them can be improved can be found in §4.17.

To add a current or voltage drive:

- Click on either current drive **I** or voltage drive **V** icon in the tool bar. The cursor will change accordingly.
- Move your cursor over the contact. When it turns white, left-click to attach the drive.

You should now see something like:



The drive must now be set up. This is done from the *Current Drive Properties* or *Voltage Drive Properties* panel, accessed by right-clicking on the drive and selecting *IProperties*. The properties are the same for both types of drive:

#### Electrical Drive Properties

- driveName** you can enter a convenient name for the drive here. The name will be used in various other places to identify the drive.
- signal** here you define the current-versus-time or voltage-versus time functions of the drive. Click the property to edit. See §4.16 for details on how to define your signal in the *Signal Editor*.


**currInstrument** this is the *Current Instrument* which measures the optical power exiting the source - see §4.15.4. Double-click the property to open the *Current Instrument Properties* panel.

**voltInstrument** this is an *Voltage Instrument* which measures the optical wavelength of the source - see §4.15.4. Double-click the property to open the *Voltage Instrument Properties* panel.

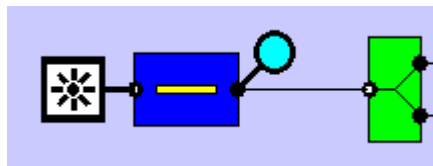
## 4.14 Monitors

You can measure the optical field at any port on your circuit by attaching a *monitor* to that point. Various instruments can be attached to the monitor point to measure the time-evolving power, time-evolving instantaneous wavelength or just record the optical field at every time-step of the TDTW simulation so that e.g. a spectrum can be generated at the end of the simulation using a Fourier transform.

To add a monitor:

- Click the monitor icon  on the toolbar. The cursor will change to a circle.
- Now move the cursor until the circle is over the port you want to attach the monitor to, and left-click.

You should now have a monitor attached to the port, as shown here:



The monitor has a number of properties as listed here, which are set in the *Monitor Properties* panel, accessed by right-clicking on the monitor and selecting *Properties*.

**monitorName** you can enter a convenient name here. The name will be used in various other places in PICWAVE to refer to the monitor, so choose a memorable name such as “MonitorIn”, “MonitorTopR” etc.

**instruments** this brings up the *Monitor Instruments Panel* from which you can add various *monitor instruments* to the monitor – see §4.15.1.

**enableSpecMeas** if you set this to *In (Out)*, then the monitor will record the complex field *entering (exiting)* the component to which the port is attached at every time step. At the end of the simulation, this data can then be used to generate a spectrum of the ingoing (outgoing) optical signal. This setting defaults to *Off* since it can use up a lot of memory for a long simulation – only enable the monitors that you need.

## 4.15 Instruments

*Instruments* are used in various places for measuring time-evolving quantities in a device when a TDTW simulation is running. The data collected by instruments can be viewed in the *Instruments* panel of the *Time Domain Results* window when the simulation is paused, stopped or complete (see §9.4 and 9.5). There are 5 main classes of instruments:



<i>Monitor Instruments</i>	associated with an optical monitor – includes <i>Facet Power Instrument</i> , <i>Facet Wavelength Instrument</i> , <i>SMSR Instrument</i> and <i>Quantum Efficiency Instrument</i> .
<i>Junction Instruments</i>	associated with an active section – these measure quantities in a diode junction.
<i>Source Instruments</i>	power and wavelength instruments measuring power and wavelength of optical sources
<i>Drive Instruments</i>	current and voltage instruments measuring the output of current and voltage sources.
<i>Contact Instruments</i>	current and voltage instruments measuring the currents through and the voltage at the contact.

All instruments have the following parameters:

<b>valueName</b>	if you set this, it will be used in graphs and reports. If you leave this blank, PICWAVE will assign one automatically.
<b>plotInMonitor</b>	if you set this to <i>True</i> then the instrument's results will be plotted in the <i>TDW Calculator's Run-time Monitor Window</i> . (Note that only the first two such enabled instruments will be displayed in the <i>Run-time Monitor Window</i> .)

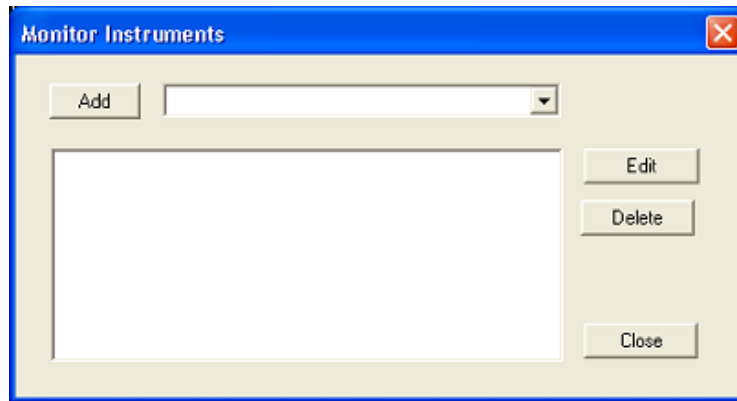
The instruments currently available, and their specific properties, are described in the following sections:

#### 4.15.1 Monitor Instruments

*Monitor Instruments* are those associated with an optical monitor. Available instruments are:

<i>Facet Power Instrument</i>	monitors the time-evolving optical power entering or exiting the port to which the monitor is attached
<i>Facet Wavelength Instrument</i>	monitors the time-evolving wavelength of the signal entering or exiting the port to which the monitor is attached.
<i>SMSR Instrument</i>	(for laser diode simulations, monitor must have <b>enableSpecMeas</b> enabled) records the time-evolving side-mode suppression ratio (ratio of strongest to next strongest peak in dB) of the signal by taking a small number of the latest time-steps, computing a Fourier transform and analysing the spectrum. (See §9.6.4 for SMSR details)
<i>Quantum Efficiency Instrument</i>	(for laser diode simulations) records the time-evolving quantum efficiency, which defined as the number of photons produced for every particle (electron or photon) entering the diode, expressed as a percentage

Monitor instruments are added, edited and deleted in the *Monitor Instruments Panel*, which is accessed from the *Monitor Properties* panel (see §4.14). The panel is shown below.



You can add the *instruments* to the monitor by selecting them from the drop-down list and clicking **Add**. You can add as many instruments as you wish, even of the same type (e.g. if you wanted measure the power in different modes etc)

The properties panel of a given monitor instrument can be edited by double-clicking on its entry in the list of added instruments, or by selecting it and clicking **Edit**.

To delete an instrument, select its entry and click **Delete**.

The properties of the monitor instruments are discussed below.

#### Facet Power Instrument Properties

The *Facet Power Instrument Properties* panel contains the following specific parameters:

- |                     |   |
|---------------------|---|
| <b>inOrOut</b>      | set to <i>In (Out)</i> to monitor the optical power entering (exiting) the port; set to <i>Off</i> to switch the instrument off   |
| <b>responseTime</b> | [ns] enter a non-zero value to smooth out the rapid fluctuations in a signal – the monitor will then filter the signal with a time constant equal to the value of this parameter.   |
| <b>dlambda</b>      | [nm] if non-zero then the instrument will measure power only at the given wavelength. The wavelength is relative to the simulation central wavelength ( <b>lambdaCentre</b> ). The instrument will have a finite bandwidth, controlled by the <b>responseTime</b> parameter – i.e. the shorter the <b>responseTime</b> the wider the bandwidth of the detector. If <b>dlambda</b> is zero, then the instrument measures the power at all wavelengths. Note that if <b>responseTime</b> is too small, you will see other wavelengths as a rapidly oscillating power. |
| <b>modePol</b>      | set this to the polarisation of the modes whose power is to be measured by the instrument: <i>TE-like</i> or <i>TM-like</i>   |
| <b>ModeIndex</b>    | the index of the mode (i.e. the position of the mode in the ordered list of modes of polarisation <b>modePol</b> ) whose power is to be measured  |

#### Facet Wavelength Properties

The *Facet Wavelength Instrument Properties* correspond directly to the *Facet Power Instrument Properties* (see above), except that there is no property corresponding to the latter's **dlambda**.

## SMSR

The *SMSR Instrument Properties* panel contains the following specific parameters:

<b>fftSize</b>	this determines the number of time-samples used to compute the Fourier transform. The larger this number the higher the wavelength resolution of the spectrum (default = 2048).
<b>filterWidth</b>	a value greater than zero applies a filter to the spectrum which allows you to reduce the effects of noise, which might otherwise give rise to subsidiary peaks. If one such noise peaks were stronger than the second strongest <i>mode</i> peak, then it would be used in calculating the SMSR – giving a spurious value. (default = 11) (See SMSR hint in §9.6.3)
<b>skipStep</b>	computing the spectrum at every time step will slow down the simulator. Set this to >1 to compute the spectrum at every nth step.
<b>modePol</b>	set this to the polarisation of the modes whose SMSR is to be calculated: <i>TE-like</i> or <i>TM-like</i>
<b>ModelIndex</b>	the index of the mode (i.e. the position of the mode in the ordered list of modes of polarisation <b>modePol</b> ) whose SMSR is to be calculated

## Quantum Efficiency

The *Facet Power Instrument Properties* panel contains the following specific parameters:

<b>responseTime</b>	[ns] enter a non-zero value to smooth out the rapid fluctuations in a signal – the monitor will then filter the signal with a time constant equal to the value of this parameter.
---------------------	---

### 4.15.2 Junction Instruments

*Junction Instruments* are those associated with associated with an active waveguide section, and measure quantities in the diode junction. In the Current Flow Model employed by PICWAVE, each active layer represents a separate (parallel) current path to the substrate. Therefore, junction quantities can be measured for a single active layer, or can be averaged over several (if the number of active layers in the RWG > 1); similarly they can be measured at a single x-coordinate or averaged over a range of lateral (x-direction) widths. Available instruments are:

<i>Junction Ne Instrument</i>	monitors the time-evolving junction carrier density for the specified active layer(s)
<i>Junction Gain Instrument</i>	measures the time-evolving gain in an active layer for the specified active layer(s)
<i>Junction Temperature Instrument</i>	measures the time-evolving temperature in an active layer for the specified active layer(s)
<i>Junction Current Instrument</i>	like the <i>Junction Current Density Instrument</i> (see below) but measures integrated current density [ $\text{mA}\cdot\mu\text{m}^{-1}$ ] (integrated in x-direction), which corresponds to the current per unit z-length

*Junction Current Density Instrument* measures the time-evolving current density [mA.cm<sup>-2</sup>] in the specified active layer(s)

*Junction Voltage Instrument* measures the time-evolving voltage across the junction, i.e. the specified active layer(s)

Junction instruments are added, edited and deleted in the *Waveguide Section Instruments* panel, which is accessed from the *Waveguide Section Properties* panel (see §4.9.1.2). The panel is identical to the Monitor Instruments panel (see §4.15.1) and is used in the same manner.

The properties of the junction instruments are discussed below.

#### Junction Ne Properties

The *Junction Ne Instrument Properties* panel contains the following specific parameters:

**zFraction** fractional z-position along the waveguide section at which the carrier density is measured.

**aLayer1/aLayer2** specifies the range of active layers [**aLayer1**–**aLayer2**] over which the carrier density is averaged, where layers are numbered from the *top down*. (e.g. for an RWG with two active layers you would set **aLayer1**=1, **aLayer2**=2, to average over both layers; or set **aLayer1**=**aLayer2**=1 to look at just the upper most active layer)

**xLateral1/xLateral2** specifies the lateral (x-direction) range [**xLateral1**– **xLateral2**] over which the carrier density is averaged, where **xLateral2** > **xLateral1** and  $0 < \mathbf{xLateral1}, \mathbf{xLateral2} \leq \text{RWG width}$

#### Junction Gain Properties

The *Junction Gain Instrument Properties* correspond directly to the *Junction Ne Instrument Properties* (see above)

#### Junction Temperature Properties

The *Junction Temperature Instrument Properties* correspond directly to *Junction Ne Instrument Properties* (see above)

#### Junction Current Density Properties

The *Junction Current Density Instrument Properties* correspond directly to the *Junction Ne Instrument Properties* (see above)

#### Junction Current Properties

The *Junction Current Instrument Properties* correspond directly to the *Junction Ne Instrument Properties* (see above)

#### Junction Voltage Properties

The *Junction Voltage Instrument Properties* correspond directly to the *Junction Ne Instrument Properties* (see above)

### 4.15.3 Source Instruments

*Source Instruments* are those associated with associated with optical sources. The two types of instrument available are:

*Power Instrument* monitors the optical power of the source  
*Wavelength Instrument* monitors the optical wavelength of the source

Source instruments are enabled from the *Optical Source Properties* panel (see §4.12). The specific properties of the source instruments are detailed below:

#### Power Instrument Properties

**enable** set this to *True* to enable the instrument.

#### Wavelength Instrument Properties

**enable** set this to *True* to enable the instrument.

**modPol** set this to the polarisation of the light whose wavelength is to be measured: *TE-like* or *TM-like*

### 4.15.4 Drive Instruments

*Drive Instruments* are those associated with associated with current and voltage drives. The same two types of instrument are available for each type of drive:

*Current Instrument* monitors the time-evolving current of the drive  
*Voltage Instrument* monitors the time-evolving voltage of the drive

Drive instruments are enabled from the drive properties panel (see §4.13). The properties of both instruments are the same, the only specific property being:

**enable** set this to *True* to enable the instrument.

### 4.15.5 Contact Instruments

*Current Instruments* are those associated with electrical contacts attached to active waveguides. Three types of instrument are available:

*Voltage Instrument* monitors the time-evolving voltage of the electrode at the specified z-position (**zFrac** –see below)

*Current Instrument* monitors the time-evolving current flowing *along* the contact electrode at the specified z-position (**zFrac**). This is only non-zero if the *Travelling Wave Electrode Model* enabled (see §4.17.2 & 11.1.3), as otherwise the electrode is assumed to be an equipotential.

*Junction Current Instrument* monitors the time-evolving current flowing from the electrode down to the line of *CoreCells* (see §11.1.3 & 11.2) below, in the *z-element* containing the specified z-position (**zFrac**)

Note that as the current flowing from the electrode down to the junction is calculated for each z-element, the measurements of the two current instruments above will be accordingly discretised given the discrete size of the z-elements (=zStep) – i.e. the currents will change in steps from one z-element to the next.

Contact instruments are enabled from the *Contact Properties* panel (see §4.17.1). The properties of all the current instruments are the same, the only specific properties being:

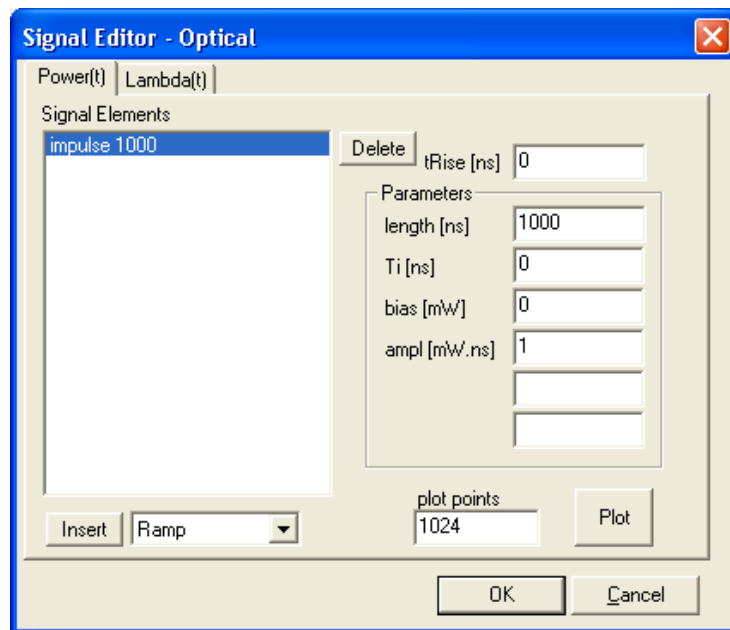
- enable** set this to *True* to enable the instrument
- zFrac** fractional z-position along the waveguide section at which the instrument measures its quantity

## 4.16 Signal Editor

The *Signal Editor* is used to define the time-dependence of a signal driving a device – the signal can be an optical source or an electrical drive.

An electrical signal has one *signal function* that defines the time-dependence of a current or voltage drive. An optical signal has two *signal functions* – one for the wavelength and one for the power of the optical signal.

The signal editor is opened from the properties panel of the optical source or electrical drive (see §4.12 and 4.13). For an optical source it appears as follows:



Every *signal function* is composed of one or more *signal elements*. In a TDTW simulation, each signal element runs for a specific length of time and then the next signal element begins to run and so on, so that the signal function consists of series of signal elements run in succession. The duration of the signal function is then simply equal to the sum of the durations of the signal elements. If the simulation time exceeds the duration of the signal function, the signal will start again from the beginning, repeating until the simulation finishes. The different types of signal elements and their use are described in §4.16 below. Further specific details on optical and electrical signals, and signal duration are given in §4.16.2 below.

### 4.16.1 Creating a Signal Function

The types of *signal element* available are as follows:

## Ramp

The *Ramp* element defines a *signal element* consisting of a linear ramp, rising from **startVal** to **endVal**. **length** is the total duration of the *signal element*. You can omit **startVal** and if you do, the ramp will start with the end-value of the previous *signal element*.

## Sine

The *Sine* element defines a *signal element* consisting of a sinusoid of amplitude **ampVal** and offset by **biasVal**. **Tp** is the period of the sine wave. The last parameter **Tp2** (ns) is optional and if given, enables you to define a chirped sine wave with initial period **Tp** and period **Tp2** at time **length** later. **length** is the total duration of the *signal-element*. Perfectly periodic signals can be created by defining just one (Sine) element, with a **length** equal to its period, **Tp**. The signal will repeat automatically as discussed above.

## NRZ

The *NRZ* element defines a *signal element* consisting of a pseudo-random NRZ bit sequence swinging between **minVal** and **maxVal**. **length** is the total duration of the *signal element*. **Tp** is the period of the NRZ sequence. **Tr** is the rise and fall time of the signal. The bit sequence is generated by pseudo-random number generator and you can ensure that you get the same sequence each time by giving a **seed** value ( $0 < \text{seed} < 215$ ). Once you have run an NRZ simulation, the program can generate an eye diagram for you (see §9.7).

## RZ

The *RZ* element is like the *NRZ* element, but the signal returns to zero between each pulse (NRZ meaning non-return-to-zero, RZ meaning return-to-zero).

## Impulse

The *Impulse* element defines a *signal element* containing an impulse. **biasVal** is the baseline amplitude and **ampVal** is the impulse amplitude. The impulse occurs at time **Ti** [ns] after the start of the signal element. **length** is the total duration of the signal element.

To add a signal element to your signal function:

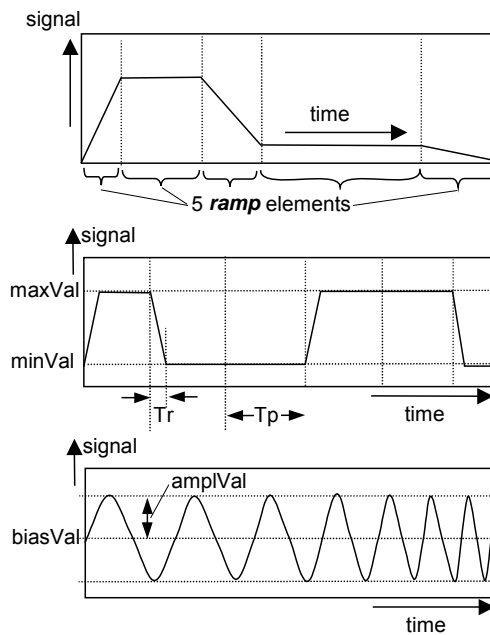
- Select the type of signal-element you wish to add from the **New element type** drop-down list
- Select the point in the list at which you want to insert the signal element, then click **Insert** to insert
- Then set the various parameters for the signal element
- Repeat the process for any further signal elements you wish to add

To delete a signal element from your signal select it from the Signal Elements and click **Delete**.

Use **plot points** to set the resolution of the graph in which the signal function is plotted.

To plot the signal function click **Plot**.

Shown in the figure below are three examples of signals, created respectively with (top) 5 ramp elements, (mid) an NRZ element, (bottom) a sine element. Each signal element is contained between two vertical dotted lines – indicating how the total signal is broken down into its constituent elements.



#### 4.16.2 Further Details On Signals

##### Electrical Signals

An *electrical signal* is required to drive a *drive node*. You can only have one *signal* per drive node. In fact there are two types of electrical signal, a *voltage drive* and a *current drive*.

Note: You cannot convert a *current drive* signal into a *voltage drive* signal or vice versa – you should delete the first and define a new signal.

##### Optical Signals

An *optical signal* must be associated with an *optical source*. The optical signal has both a time dependent wavelength and a time-dependent power. You may specify each of these separately – one may be smoothly increasing ramp, and the other oscillating sinusoidally, for example. You may inject more than one optical signal into a facet, allowing you, for example, to analyse WDM devices.

The signal power definition,  $Power(t)$ , defines the injected optical power  $P(t)$  in mW. The signal wavelength definition,  $Lambda(t)$ , defines the wavelength *offset*  $\Delta\lambda(t)$  in nm from the central simulation wavelength, which is defined elsewhere. Note that the injected power is the power incident on the facet – some of this power may be reflected by the facet.

## 4.17 Contacts

Each waveguide section may have one or more electrical *contacts*. Contacts are used to inject current into an active section such as an SOA. A *contact* runs the whole length of the section.

Beyond the simple *contact* set-up outlined in §4.13, further *contact* parameters can be set (i.e. changed from their defaults), which specify the geometry and the connection to the drive. These are described in §4.17.1 below.

In the current version of PICWAVE, the connection between a *contact* and an *active layer* is defined only by a resistivity (see §4.13), and is modelled according to the *Current Flow*

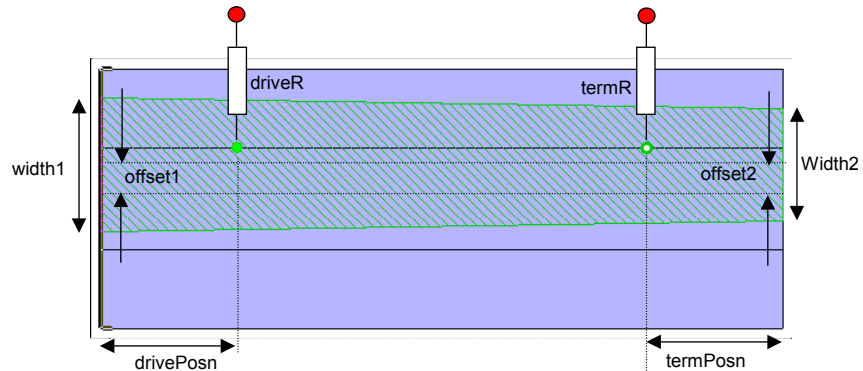


*Model* (see §11.1.2). However, the modelling of the *contacts* themselves can be improved by using the *Travelling Wave Electrode Model*, which allows for the simple modelling of the effects of capacitance across the junction and the inductance of the contacts. The details of this model are presented in §11.1.3, which should be read in conjunction with §4.17.2 below, which describes how the model is enabled and the various parameters that can be set.

Contacts also have associated *Contact Instruments* which are used to measure the contact voltage and contact currents. They are added distinguished from the corresponding drive and junction instruments by virtue of the electrical resistances between the drive and the contact (see §4.17.1 below) and the resistivities defining the connection of the contact with each active layer (see §4.13). Details on the contact instruments can be found in §4.15.5.

#### 4.17.1 Contact Properties

The *contact* properties are defined graphically in the picture below.



The properties set in the *Contact Properties* panel, accessed by right-clicking on the contact and selecting */Properties*. They are as follows:

<b>contactName</b>	you can assign a convenient name to the contact. This will be used elsewhere in the user-interface.
<b>depth</b>	[um] depth of the contact from the top (thickness)
<b>width1</b>	[um] width of contact at left side.
<b>width2</b>	[um] width of contact at right side.
<b>offset1</b>	[um] offset of centre of contact at left side from centre of waveguide.
<b>offset2</b>	[um] offset of centre of contact at right side from centre of waveguide.
<b>drivePosn</b>	[-] fractional position of drive connection onto the contact, measured from start of contact and scaled by total contact length.
<b>driveR</b>	[ohm] drive impedance (pure resistive).
<b>termPosn</b>	[-] fractional position of terminator connection onto the contact, measured from end of contact and scaled by total contact length.
<b>termR</b>	[ohm] terminator impedance.

#### 4.17.2 Travelling Wave Model Properties

(See §11.1.3 for more details about the *Travelling Wave Electrode Model* and §14.3 for details on the *Microstrip Calculator*, which is used for calculating the circuit parameters needed for the model)

The parameters for the *Travelling Wave Electrode Model* are also set in the *Contact Properties* panel (see above). The parameters are as follows:

<b>twmEnable</b>	enable or disable the <i>Travelling Wave Electrode Model</i> . If disabled, then the contact is treated as a purely resistive connection between the Drive Node and the electrical paths to the active layer(s).
<b>twModel</b>	you may either define the travelling wave characteristics by the effective index of the travelling wave mode and the microwave impedance of the contact ( $N_{eff}$ & $Z$ ); or by $L$ & $C$ ; or automatically according to the
<b>neff</b>	effective index of the microwave mode travelling along the electrode.
<b>L</b>	[H/m] inductance of electrode per unit length.
<b>C</b>	[F/m] capacitance of electrode per unit length.
<b>R</b>	[ohm/m] resistance of electrode per unit length.
<b>Z</b>	[ohm] impedance of contact at microwave frequencies.

## 4.18 Mode and Polarisation Settings : general notes

### TE and TM-like modes in TDTW simulations

In TDTW simulations, a distinction is made between TE and TM-like modes, that is, modes with a TE-fraction >50% and modes with a TE-fraction < 50%, respectively. This is reflected in the properties panels of certain circuit components and various time domain results interfaces in that modes are specified by their polarisation and an index (see Mode settings for circuit components section below). For details on TE-fraction, see §17.2.4).

### Active mode propagation in waveguide and taper sections

From the *Mode List* of a given *RWG*, only the *active modes* (see §5.5) can be propagated in a TDTW simulation if the *RWG* is used to define the cross-section of a *waveguide section* or a *taper section*. The active modes are divided into TE and TM lists, internally, when a simulation is started. The requested numbers of modes in the TE and TM lists are specified by the **numTEModes/numTMModes** parameters in the section's properties panel. The total number of modes propagated through a section may therefore be less than the number of active modes in its *RWG*'s mode list, but PicWave will warn you if there is an insufficient number of active modes to populate the TE and TM lists with the requested number of modes. For taper sections, the active modes to be propagated are those in Mode List of Lhs WG Ref; the corresponding active modes must also be present in the Mode List of the Rhs WG Ref (e.g. the  $x$ th TE mode in the Mode List of Lhs WG Ref can only be propagated through a taper section if the Mode List of Rhs WG Ref has an  $x$ th TE mode that is also active).

## Mode settings for circuit components

As far as circuit components and TDTW simulation results are concerned, a mode is specified by its polarisation and an index, where the latter is the position of the mode within list of modes of the same polarisation, ordered by decreasing effective index..

After creating a circuit, having connected up all its components, it is important to ensure that the number and polarisation of modes supported by each component have been specified consistently throughout the whole circuit. With the exception of the *general join*, all sections connected to a join must have the same number of modes in each polarisation. Also, when using an optical source or monitor instrument, the mode specified by the polarisation (**modePol**) and mode index (**ModeIndex**) settings must be one that is being propagated by the component to which the optical source or monitor instrument is attached.

PICWAVE will warn you if there are any discrepancies in the settings described above when you try to run a simulation. The simulation will not run until the necessary corrections have been made. By default all mode/polarisation settings are set to the first TE mode.

General joins allow the sections to which they are joined to have different numbers of modes in each polarisation.

# Waveguide Cross-Sections




## 5.1 The RWG Waveguide

The *RWG Waveguide* defines a “rectangular geometry waveguide”. The *RWG Waveguide* structure is composed of one or more “slices” arranged side by side. Each *slice* is uniform in the horizontal direction but may have one or more layers in the vertical direction. The *RWG Waveguide* also has a facility for simulating an etching process (described below).

A *SWG* (a 1D-slab waveguide structure) defines the vertical profile of a slice. One can use the same *SWG* for all slices or one can use different ones as is convenient. In the former case, lateral structure is achieved by varying the etch depth.

### 5.1.1 General Usage within PICWave


To add an *RWG Waveguide* to a *project*:

- Click on the  Project in the project tree, then click on the  Add RWG Waveguide button *or* right-click on the  Project and then select */Add/RWG Waveguide*

To define/edit the *RWG* structure:

- Double-click on the newly added  *RWG* waveguide in the project tree. This will open the *RWG Window* (see §5.1.2).

To use an *RWG Waveguide* to define the cross-section of a waveguide section in a PICWAVE circuit:

- Right-click on the  *RWG* waveguide in the project tree and select */Copy*
- Then right-click on the waveguide section (whose **crossSectionDef** needs to be set to *waveguide* as explained in §4.9.1.1) in the device window and select */Paste WG Ref*.

In a similar manner, two *RWG Waveguides* can be used define the cross-section at the left and right-hand ends of a *taper section*, using */Paste Lhs WG Ref* and */Paste Rhs WG Ref*

A convenient way to open the *RWG Waveguides* used to define the cross-section(s) of a waveguide section (taper section) from the device window is to

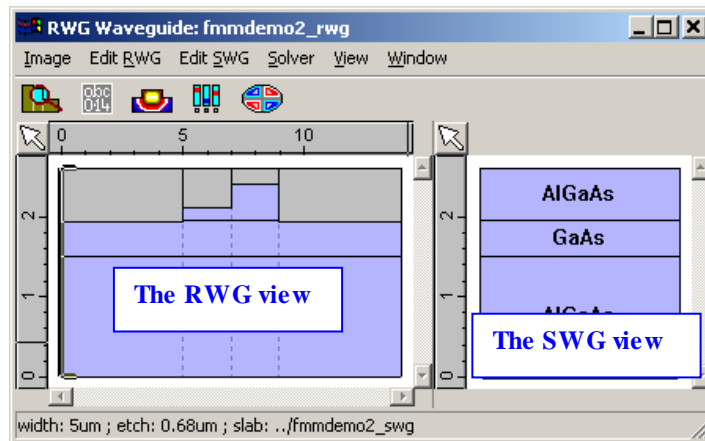
- Right click on the waveguide section (taper section) and select */Edit WG* (*/Edit Lhs WG* or */Edit Rhs WG*)

The taper section also has an */Edit WG at...* option in the right-click menu. This becomes enabled when a TDTW simulation is paused or completed; it enables one to view the taper cross-section at any position *z* along the length of the taper, thereby allowing one to view the cross-sections that have been generated for the sub-sections of the taper section (see §4.9.2).

Taper sections and waveguide sections with cross-sections defined by *RWG*s propagate a user-specified number of modes in each polarisation (see §4.9.1.4). These modes are searched for from amongst the *active modes* in the *RWG*'s *Mode List* (see §4.18).

### 5.1.2 The *RWG* Window

The *RWG Window* is the default GUI editor of the *RWG Waveguide* node. From this you can edit any part of the structure and access the mode finders. The main area of the window is taken up with a cross-section display of the waveguide (the *RWG view*). Etched regions of a slice are shown shaded. On the right is a view of the currently selected slice (the *SWG view*). Use the mouse or cursor keys to select, cut, copy, paste one or multiple slices in the *RWG view*. The same can be done with the layers in the *SWG view*.



To edit the *RWG* parameters for the waveguide, select */Edit/Waveguide*. The *RWG Editor* panel will appear (see §5.1.3).

To edit the parameters of the selected *SWG*, select */Edit SWG /properties*. The *SWG Editor* panel will appear (see §5.1.4).

You can also edit everything as text - select */Edit/As text*. The *Waveguide Text Block* will appear in the Block Editor.

#### 5.1.2.1 The menu bar

The *RWG Window* contains the following menu items:

<i>/Image/Print</i>	print waveguide window
<i>/Image/Printer Setup</i>	setup printer
<i>/Image/Export</i>	export waveguide image
<i>/Edit/Waveguide</i>	open waveguide property editor
<i>/Edit/Set Material database(.mat)</i>	associate a material database file with the waveguide
<i>/Edit/Unset Material database</i>	remove material database from waveguide.
<i>/Edit/As text</i>	edit waveguide as text.
<i>/Edit/Copy this node</i>	copy this node to the clipboard.
<i>/Edit/Insert new slice</i>	insert new slice at cursor position.
<i>/Edit/Copy slices</i>	copy selected slices.

/Edit/Paste slices	paste slices from clipboard over selected ones, or at cursor position if none selected
/Edit/Cut slices	cut selected slices to clipboard.
/Edit/Delete slices	delete slices.
/Edit SWG/properties	open waveguide property editor
/Edit SWG/Insert new layer	insert new layer in SWG window.
/Edit SWG/Copy layers	copy selected layers in SWG window.
/Edit SWG/Paste layers	paste layers in SWG window from clipboard over selected ones, or at cursor position if none selected
/Edit SWG/Cut layers	cut selected layers in SWG window to clipboard.
/Edit SWG/Delete layers	delete layers in SWG window.
/Solver/Open Solver	open default mode solver.
/Solver/Options	edit solver options.
/View/RIX preview	show RIX profile window brings up dialog asking for area to be plotted. Just press return to plot default view.
/Window/Hide this	hide this window.

### 5.1.2.2 RWG slice Popup menu

Right-clicking on a slice in the RWG view gives a popup menu containing the following:

/properties	brings up the <i>RWG</i> Editor dialog with this slice selected.
/select	sets the cursor to selection mode.
/zoom	sets the cursor to selection mode.
/scroll	sets the cursor to scroll mode.
/Copy SWG	copies the SWG of the selected slice to the clipboard.
/Paste SWG	pastes a new copy of the SWG on the clipboard into the selected slice.
/Paste SWG Ref.	sets the SWG in the selected slice to the one on the clipboard. Do this if you want to share an SWG with another slice.
/Edit SWG...	brings up the SWG editor dialog.

### 5.1.2.3 SWG slice Popup menu






Right-clicking on a layer in the SWG view gives a popup menu containing the following:

/properties	brings up the SWG editor dialog with this layer selected.
/select	sets the cursor to selection mode.

- /zoom sets the cursor to selection mode.
- /scroll sets the cursor to scroll mode.

#### 5.1.2.4 The toolbar.

The *RWG Window* contains the following toolbar items:

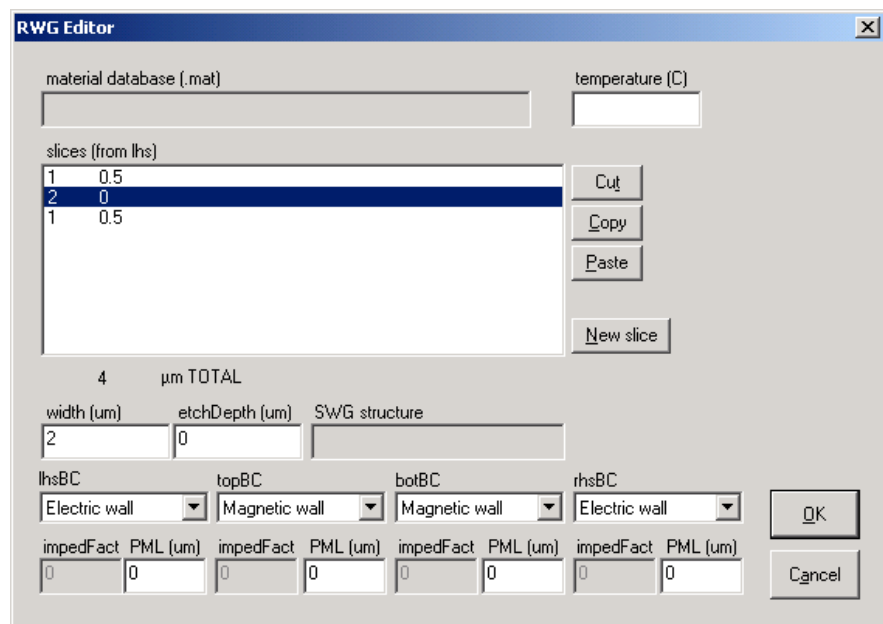
-  edit waveguide as text.
-  open waveguide editor.
-  show RIX profile window.
-  edit solver options.
-  open default mode solver.

#### 5.1.3 The RWG Editor Panel

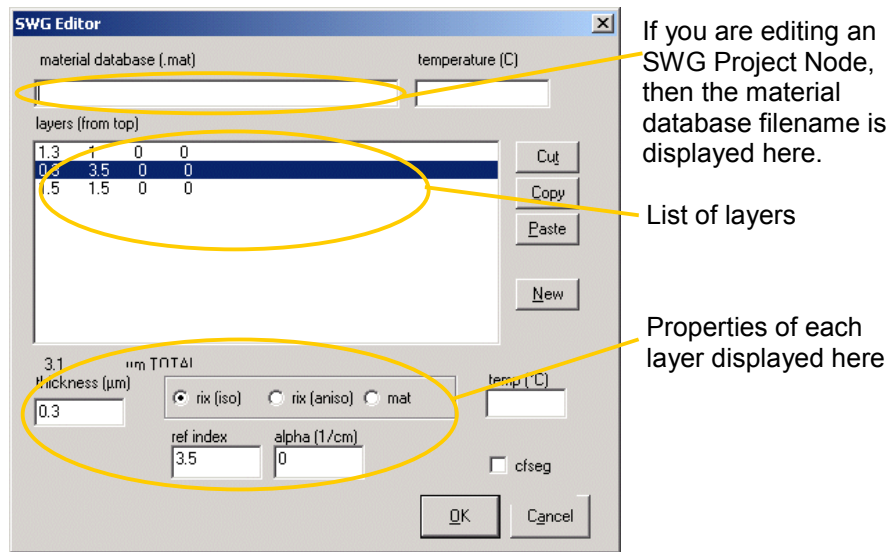
The *RWG Editor* panel, shown below, allows you to alter all the *RWG* parameters. You can add and delete slices using the **cut**, **copy**, **paste** buttons. You can add a *new* slice by setting up the parameters in the boxes at the bottom and then pressing **New slice**. You can edit the SWG associated with each slice by double clicking on the corresponding row in the slice list. To edit the parameters for a given slice, select it in the list and its parameters – slice **width**, **etchDepth**, will be displayed below the list.

Note: a pasted slice will refer to the same SWG as the copied slice.

From this panel you can also set the boundary conditions and pml parameters. See §5.1.5 for a definition of the RWG parameters.



### 5.1.4 The SWG Editor Panel



The *SWG Editor* panel, shown above, allows you to alter all the *SWG* parameters. You can add and delete layers using **Cut**, **Copy** and **Paste**, or add a completely new layer with the **New** button. You can choose to assign a material name to a layer instead of defining its refractive index and material loss coefficient. The material name must be defined in the *Material Database* file named for your waveguide.

(Hint: you can see the refractive index of a given material at the wavelength specified in */Solver/Options*: If you now click on the **rix** radio button to convert the layer to an explicit type, you will see the refractive index that the program calculated. Click the **mat** radio button to return it to material type)

The **temp** (temperature) box is a read only field – in the current version of PICWAVE it is unused.

See §5.1.9 for a definition of each parameter in this panel.

### 5.1.5 The RWG parameters.

The parameters are:

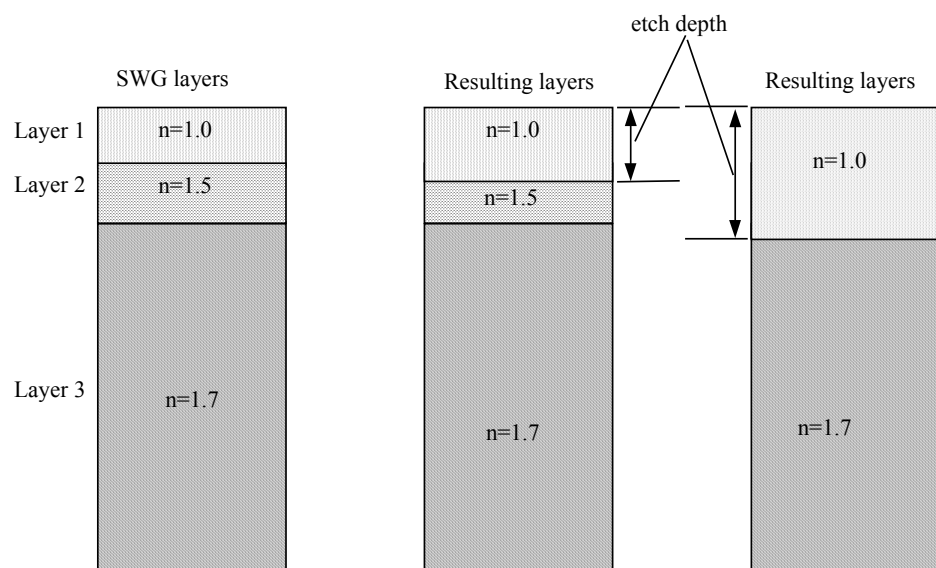
- material database** Name of a material database file (.mat).
- temperature** Waveguide temperature (in Celsius), if this is set to –1000 then defaults elsewhere are used.
- width** (um) width of the selected slice
- etchDepth** (um) etch depth of the selected slice
- lhsBC, topBC, botBC, rhsBC** These define the boundary at the boundaries of the waveguide. They refer to the left, top, bottom and right border respectively. Values can be *metal*, *magnetic*, *periodic*, *impedance*, and *Transparent* [latter two not used in current version of PICWAVE]
- ImpedFact** Impedance factor used for IMPEDANCE boundary conditions.
- PmlWidth** PML width (not used in current version of PICWAVE)



### 5.1.6 The etching mechanism

A waveguide profile may be built up either by using distinct vertical structures (*SWG*s) or by using a common *SWG* and etching the profile to varying depths with this facility. The layer structure you defined in the *SWG* will be “etched” away to the depth specified within the *RWG Editor*. The etched material is replaced by the material (refractive index etc) of the *uppermost* layer of the slice. Hence, you can define the refractive index of the etched region by adding a top layer of zero width with the etching refractive index. The results of the etch mechanism may be seen graphically in the *RWG Window* - etched regions are shown shaded.

This is illustrated in the figure below. On the left is the unetched layer structure as entered in the *SWG* layer structure. The centre and right figures show the resulting layer structure that the program generates after applying the etching, for two different etch depths.



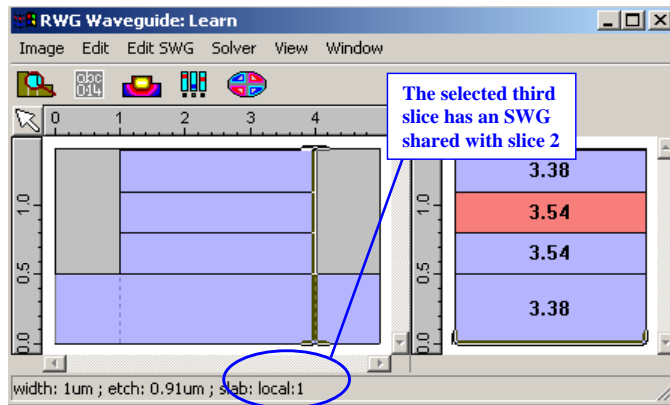
### 5.1.7 Sharing the same SWG within the RWG

An *SWG* defines the vertical layer structure of a slice of the *RWG*. An *SWG* may be stored as a local object of an *RWG Waveguide*, or in a separate node (an *SWG* node type). When creating a new slice (*/Edit /Insert New Slice*), a new *SWG* to which the slice refers is created internally.

You can then make several slices share this same *SWG* in two ways:

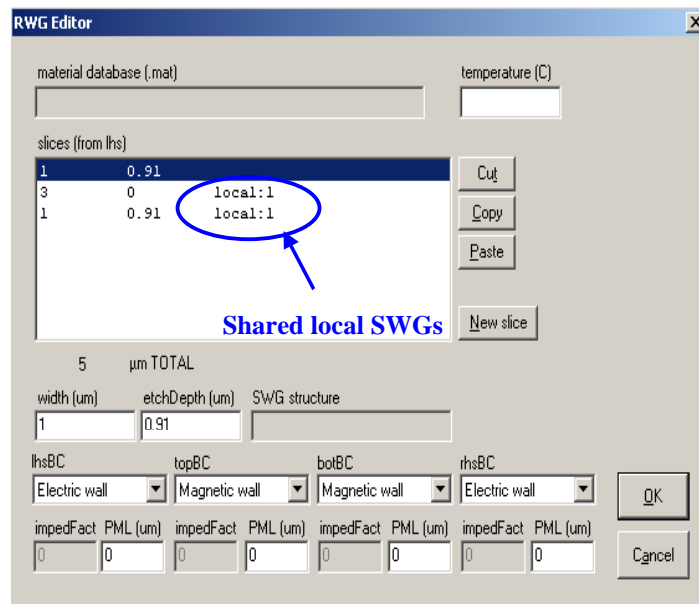
- By copying and pasting slices in the *RWG Editor* panel or *RWG Window* (for the latter, select consecutive slices by dragging the cursor over them, do */Edit /Copy Slices*, click before another slice, do */Edit /Paste Slices*).
- By copying an *SWG* and pasting it into another slice as a reference in the *RWG Window* ((right-click on the slice to be copied, select */Copy SWG*, right click on the second slice, select */Paste SWG Ref.*)

Note that when clicking on a slice sharing a local *SWG*, a string of the form “local:<number>” appears on the status bar at the bottom as shown here:



the number will be the first slice from the left referring to this SWG.

This name also appears in the *RWG Editor* panel as shown here (click on the  icon on the toolbar)




### 5.1.8 Sharing SWG's across RWG's: the SWG node

The *SWG* described above are all examples of local *SWG*s– these are stored within the *RWG Waveguides*. Although you can make slices share the same local *SWG*, you cannot share local *SWG*'s across different waveguides. To do this, you must use a *SWG Waveguide* node. Slices of two or more *RWG*'s may refer to a *SWG* node, so long as both the *RWG*'s and the *SWG* are contained within a *Project Node*.

This is best illustrated with an example. Firstly, create a *Project Node* containing two *RWG Waveguide* nodes and one *SWG Waveguide* node:

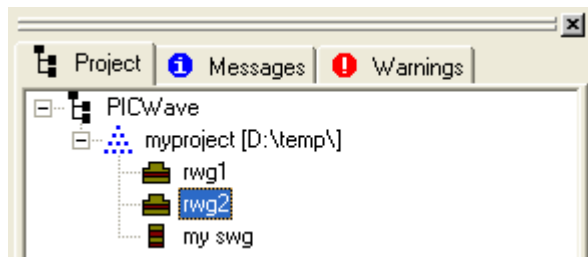
- Right-click on the “PICWave” icon in the *Project Tree* and select */Add/fimmwave Project*.
- Enter “myproject” in the dialog box that appears and press “OK”.

The *Project* icon  will appear in the *Project Tree* as shown here




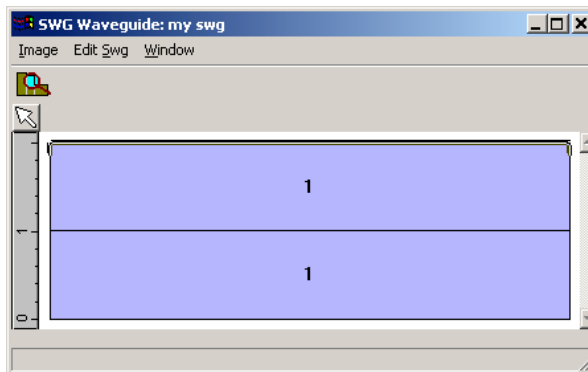
- Right-click on the “myproject” icon in the *Project Tree* and select **/Add/RWG Waveguide**. Enter the name “rwg1” in the dialog box that appears and press **OK**.
- In the same way, add a second *RWG* with the name “rwg2”
- Similarly add a *SWG node*: right-click on the “myproject” icon again and select **/Add/SWG Waveguide**. Enter the name “my swg” in the dialog box that appears and press **OK**.

The *Project Tree* should now look like this:



Next, insert some layers in “my swg”:

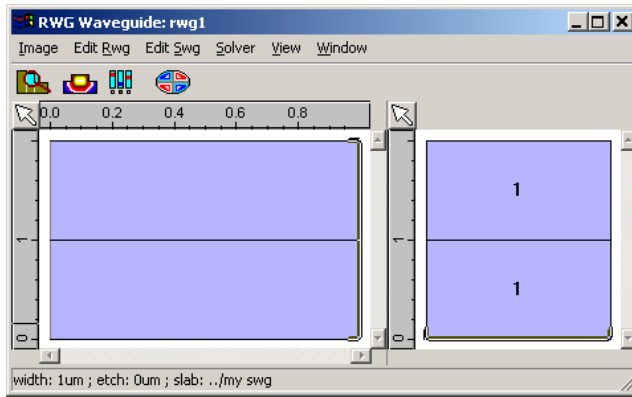
- Open the *SWG Window* of “my swg” by double-clicking on the icon  in the *Project Tree*.
- In the *SWG Window*, select **Edit Swg/Insert Layer** twice. Two layers with default width 1 and refractive index of 1 will be created as shown here.



Finally insert “my swg” into the slice of “rwg1” and “rwg2”, as follows:

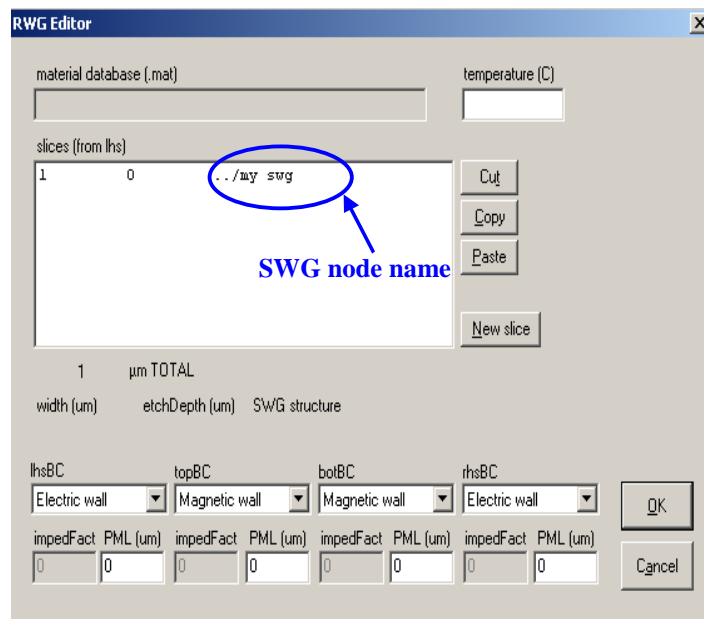
- In the *SWG Window* of “my swg”, select **Edit Swg/Copy This Node**.
- Then open “rwg1” from the project tree. In its *RWG Window*, insert a new slice (Ctrl-N)
- Right click on the slice and select **/Paste SWG Ref**.
- Do the same in the “rwg2” editor.

The slices of both *RWGs* now refer to the *SWG node* “my swg”. Both *RWGs* should look like this:



Note that the name “my swg” appears on the status bar at the bottom.

This name also appears in the *RWG Editor* panel as shown here (click on the  icon on the toolbar)



### 5.1.9 The SWG parameters

Each layer in an *SWG* has the following parameters:

**thickness** (microns) the thickness of the selected layer.

**ref index** the real part of the refractive index of the layer.

**Note** that this can also be a list of 6 numbers, which corresponds to the components of the anisotropic tensor:

$\mathbf{N}_{r_{xx}}, \mathbf{N}_{r_{yy}}, \mathbf{N}_{r_{zz}}, \mathbf{N}_{r_{xy}}, \mathbf{N}_{r_{yz}}, \mathbf{N}_{r_{xz}}$ .

**alpha** (1/cm) material absorption coefficient in 1/cm. The real solver will calculate a mode loss by doing an overlap integral with the material absorption coefficient profile. The complex solver uses this to build a complex refractive index profile. The imaginary part of the refractive index is related to **alpha** by  $\alpha = n_i \cdot 4\pi/\lambda$ . Be careful about units if you wish to convert between the two.

**Note** that as above this can also be a list of 6 numbers, which

corresponds to the components of the anisotropic tensor:

**alpha<sub>xx</sub>, alpha<sub>yy</sub>, alpha<sub>zz</sub>, alpha<sub>xy</sub>, alpha<sub>yz</sub>, alpha<sub>xz</sub>.**

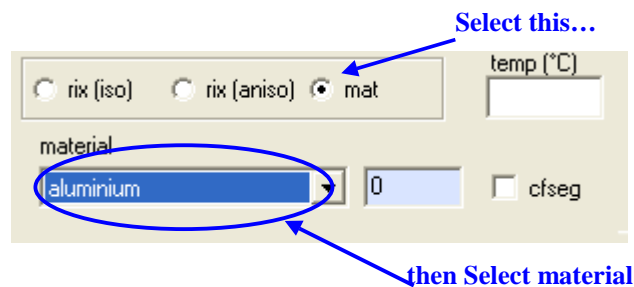
**cfseg** Indicates that this layer is to be included in the confinement factor integration, and used as an active layer in active sections

**calcTemp** (read-only) the computed temperature of the layer will be written here (NOT IN USE).

**Note:** The actual waveguide profile is affected by the etching of the structure. Etching will expand the top layer downward through the other layers. For this reason, the top layer may be zero in width.

### 5.1.10 Using a material database to define material layer

You can give a material name to a layer in an SWG. You do this by selecting **mat** radiobutton at the bottom of the SWG Editor panel, as shown here.



The refractive index and other parameters of the layer will then be retrieved from the Material Database. The format of a material layer is:

*“Matename(x)”*

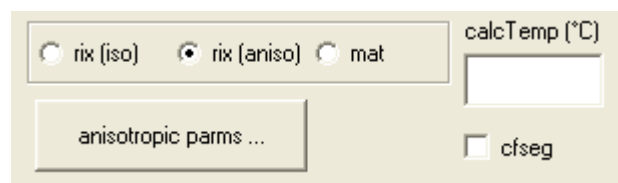
Note the composition parameter “x” - this will be passed to the material database where you can define x-dependence.

Note that to use this feature you must first make a material database available to the waveguide. You do this via **Edit /Set Material Database**, in the *RWG Window*.

### 5.1.11 Anisotropic waveguides.

PICWAVE now has some capability for dealing with anisotropic waveguides with the *FDM Solver* (NOT the *Fibre Solvers*, or the *Effective Index Solver*), where the refractive index is no longer a scalar, but a symmetric 3 by 3 tensor. For the *FDM Solver* you can set  $N_{xx}$ ,  $N_{yy}$  and  $N_{zz}$  independently.

To set an anisotropic refractive index, click on **rix (aniso)** and click **anisotropic parms...** as shown here.



This allows you to access

$N_{xx}$ ,  $\alpha_{xx}$ ,  $N_{yy}$ ,  $\alpha_{yy}$ ,  $N_{zz}$ ,  $\alpha_{zz}$ .

## 5.2 Boundary conditions.

All waveguide formats support several types of computational boundary conditions for calculating the modes of the structure. All the waveguides support perfect metal wall ( $E_{//} \rightarrow 0$ ) and perfect magnetic wall ( $H_{//} \rightarrow 0$ ) boundary conditions.

### 5.2.1 The IMPEDANCE boundary condition.

This allows you to define a boundary condition that is anywhere between a perfect metal wall ( $E_{//} \rightarrow 0$ ) and a perfect magnetic conductor ( $H_{//} \rightarrow 0$ ). This is achieved using the Leontovich boundary condition <sup>1</sup>:

$$\mathbf{n} \times \mathbf{E} = Z\mathbf{n} \times (\mathbf{n} \times \mathbf{H})$$

or equivalently:

$$\mathbf{n} \times \mathbf{E} = Z\mathbf{H}_t$$

where  $\mathbf{n}$  is the outward unit normal to the waveguide surface and  $Z$  is the impedance.

For a 1D+z (i.e. slab) waveguide varying in the x-direction, so that  $\mathbf{n}=(\pm 1,0,0)$  this gives  $E_z = -Z.H_y$  and  $E_y = Z.H_z$ . In PICWAVE, the impedance is specified by the a-dimensional real **impedance factor**  $Z_f$ :

$$Z = i\mu_0 C \frac{Z_f}{\sqrt{1 - Z_f^2}}.$$

Hence an **impedance factor** of 0 gives the metal wall, while a factor of +1.0 or -1.0 gives the magnetic wall. The impedance factor must be in the range [-1.0,1.0]. Negative values can be used to invert the relative  $\mathbf{E}$  and  $\mathbf{H}$  field directions on the waveguide surface.

Note that the boundaries are still perfectly reflecting, as the impedance is always pure imaginary. In spite of this, the impedance boundary condition can be useful to reduce the effect of the boundaries on a given mode profile – in principle you could choose a value that would allow you to bring the boundaries much closer in than would otherwise be the case.

## 5.3 Defining Waveguides - Guidelines

There are a few guidelines that you should think about when defining your structure.

### 5.3.1 Boundaries

It is important to choose a sensible position for the boundary walls defining your problem. Too far away and spatial resolution is lost. Too close to the mode and the mode will become corrupted and inaccurate. The only way to decide this matter is by trial and error unless you have some idea of the extent of the mode beforehand. Try either a) moving the boundaries or b) changing the boundary conditions and observe how much the propagation constant has changed.

---

<sup>1</sup> See IEE ELECTROMAGNETIC WAVES SERIES: “*Approximate boundary conditions in electromagnetics*” by T.B.A.Senior and J.L.Volakis – page 11.

A quick way to determine whether the boundaries are far enough away is to plot a cross-section of the mode's field profile (e.g.  $E_x$  for a TE-like mode). If the boundaries *are* far enough away you will see a nice exponential decay of the field away from the mode centre and this will be very small by the time it reaches the walls. I.e. both  $E_x(x)$  and  $dE_x/dx \rightarrow$  zero. If the walls are too close then either the field or its gradient will be non-zero, depending on what boundary condition you have chosen.

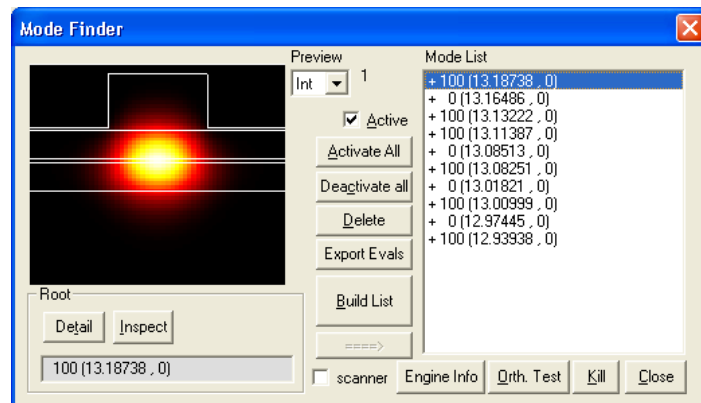
*The above comment regarding lost spatial resolution only applies to the top and bottom walls.*

## 5.4 The Mode Solvers

Once you have defined a waveguide you are ready to find the modes. It is important to understand that the mode solvers use iterative techniques. This implies two things - a) they locate modes one at a time and b) they need a starting guess. The *MOLAB* (see §5.6) provides a fully automatic algorithm for driving the iterative solvers.

## 5.5 The Mode Finder Panel

Assuming you have loaded a waveguide, then from the menu, select */Solver/Open Solver* to invoke the *Mode Finder*. If there are no solver parameters yet saved for this waveguide, PICWAVE will give an error message and ask you to specify some sensible solver parameters i.e. non-zero wavelength. The *Mode Finder* panel will then appear (there may be a short delay while the solver does some preliminary calculations).



The *Mode Finder* panel, shown above, is the main interface for all the solvers. The mode finder will automatically select a default solver if none are specified in the waveguide. For a *RWG Waveguide*, this will be the *Real FDM Solver*. You can change the solver used at any time from within the *Mode Finder* panel, as described later. The following sections assumes that you are using the *Real FDM Solver*.

To build a list of modes click **Build List**, this will open the *Build EV List Panel* (see below) which is the central Interface to the *MOLAB* (for automatically building the *Mode List*). Its buttons are described below

Build EV List Panel buttons

**Build Method**

select the *New build* to invoke the *MOLAB*, or *Polish existing list* to find the modes using the current solver settings, using the modes currently in the list as initial approximations. This is useful if you

want to find the same modes using finer for which the modes in the existing list are good initial guesses.

**Builder options**

invokes the *MOLAB Options Panel* (see §5.6.1).

**Set as default**

sets the current *MOLAB* parameters as the default ones of the waveguide. The options will be retained when the Mode Finder is closed, or will be lost otherwise.

**Start**

starts the chosen builder.

Other features of the *Mode Finder* panel are described below:

The Mode list

When you click on an entry in the *Mode List*, the entry is copied to the *Root Box*, so that when you press **Inspect**, it is this mode that you will plot, etc.

All roots are preceded by a state symbol: '+', ' ' (blank), '?', 'x'. These respectively indicate that the root is active, inactive, incompatible, numerically imprecise.

You can also inspect a mode by double-clicking on a list entry.

**Active** activates/deactivates the selected eigenmode (N.B. sections using an *RWG* can only propagate its *active modes*, see §4.18)

**Activate all**

activates all consistent eigenmodes in the list.

**Deactivate all**

deactivates all consistent eigenmodes in the list.

**Delete**

deletes the root currently highlighted in the *Mode List*.

**Export Evals**

export modes to ASCII file

**Build List**

opens the *Build EV List Panel* (see below) which is the central Interface to the *MOLAB* (for automatically building the *Mode List*).

The Preview Window

**Preview**

selects the field component to be previewed.

<right click> / Export image

exports the preview image to a file

<right click> / Edit Current Mapscheme

edit the currently selected map scheme

<right click> / Mapschemes...

choose a map scheme for the preview

Other Buttons

**Engine Info...**

Not used by any of the mode solvers - *Effective Index Solver* or (real) *FDM Solver* - in the current version of PICWAVE.

**==>**

copies the eigenmode in the *Root Box* to the *Mode List*



<b>Detail</b>	this will bring up a little panel giving you more information about the mode. The <i>TE-fraction</i> displayed may only be an estimate so do not rely on it - use the <i>Inspect Mode Panel</i> .
<b>Inspect</b>	opens the <i>Inspect Mode Panel</i> for the currently highlighted root in the <i>Mode List</i> .
<b>scanner</b>	opens the manual scanner, an alternative to MOLAB for finding modes – <i>the manual scanner is either not used, or not recommended for use, with any of the mode solvers in the current version of PICWAVE</i>

## 5.6 The MOLAB

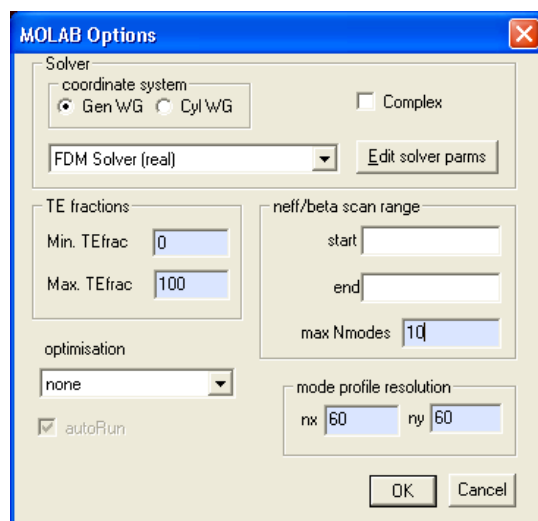
The easiest way to find the modes is to let the solver do the work automatically. This is done by invoking the *MOLAB*. In this section, we explain how to use this.

The *MOLAB* is a reliable automatic scanner that is capable of finding with almost complete certainty all eigenmodes in a given range, including those with multiple modes. Note that it is a more reliable tool than the manual scanner (described in section 5.3) for finding modes and can work with little or no user intervention, although it is somewhat slower. Use this if you require that ALL the modes within a specified eigenvalue range be found.

The *MOLAB* builds the list automatically in a variety of ways. For example, you could choose to find the first 5 modes of highest effective index, find all modes in a given effective index range, or find only modes with a TE-fraction greater than some threshold. The *MOLAB* works identically with all the mode solvers available in PICWAVE.

### 5.6.1 The MOLAB Options.

The *MOLAB* Options can be accessed via the *RWG Window's /Solver/Options* menu, or by pressing **Build List** and **Builder Options** in the *Mode Finder* dialog. These options control how and what eigenmodes are found by the *MOLAB*.



<b>Solver</b>	selects the <i>Mode Solver</i> to be used by the <i>MOLAB</i> .
<b>coordinate system</b>	select either <i>Gen WG</i> (for waveguides of general geometry, typically RWG or MWG waveguides) or <i>Cyl WG</i> (for waveguides of cylindrical geometry, typically FWG waveguides – unused in current version of PICWAVE)
<b>Complex</b>	if ticked, the <b>Solver</b> list will display the complex mode solvers, if unticked the real mode solvers – complex mode solvers not available in current version of PICWAVE
<b>TE fractions</b>	the TE-fraction (fraction polarised in Ex/Hy direction) range for valid modes. The <i>MOLAB</i> will insert as active only those modes with TE-fractions within this range (see §4.18). In Fimmwave, TE modes have E-field primarily in the x-direction and TM modes have E-field primarily in the y-direction.
<b>optimisation</b>	<i>Mode builder</i> accuracy (optimisation). You can choose between <i>fast</i> and <i>best</i> . We recommend the <i>best</i> option, unless you are sure that the field magnitude of the eigenmodes is large at the matching interface.
<b>autoRun</b>	enables the <i>MOLAB</i> to be run automatically when a TDTW simulation is started and the <i>Mode List</i> is out of date, e.g. if a waveguide dimension has changed. You may wish to disable it in order to use an eigenmode list already existent in the waveguide cross section.
<b>start, end</b>	the mode range in which to scan for eigenmodes. A negative value refers to the imaginary part of neff/beta. If you set start<end, then a default upper bound is used.
<b>max Nmodes</b>	the maximum number of modes to be found in the specified range.
<b>nx, ny</b>	specifies the resolution to be used by the <i>MOLAB</i> when calculating the mode profiles (done when evaluating TEfracs, for example). This does not influence the procedure of finding the mode or the way the structure is discretised. (See documentation for the particular mode finder for details of structure discretisation).

**Edit solver parms** brings up the *parameter editor* of the specified *Mode Solver*.

## 5.6.2 Finding and inspecting modes

Once you have found a mode or a list of modes, they are inserted into the *Mode List* in the top right of the *Mode Finder* panel. This list is saved to disk in a state file when you close the waveguide editor. The entries in this list record several pieces of information about the mode, the most important being its eigenvalue.

Once you have built your *Mode List*, you can quickly inspect the mode profile to see if the mode is a) the one you want and b) accurately generated. If the mode appears very non-physical then you may need to increase **max Nmodes** (see the next section). To bring up the *Inspect Mode* Panel, simply highlight an entry in the *Mode List* and then click the

**Inspect** button.

See §5.7 for more information about the *Inspect Mode* panel.

**NOTE:** Every time you add a new eigenmode to the list, the mode will be inserted in order of decreasing value of the real part of the refractive index.

### 5.6.3 Checking mode accuracy

Mode accuracy is determined by two factors: the accuracy of the propagation constant and the accuracy of the profile. Clearly both of these depend on the computational parameters used in the solvers. All solvers (with the exception of the *Effective Index Solver*) are accurate in the sense that, as the computational parameters are refined, the numerical solutions found will tend to the real ones. In order to determine just how accurate these modes are, PICWAVE provides mechanisms for checking both of these factors.

There is a simple generic criterion that can be used to check the accuracy of the modes: modal orthogonality. You can do this in PICWAVE via the **Orth.Test** button on the *Mode Finder* panel. This will calculate the largest overlap between different modes. Ideally this should be zero. The departure from zero of this value gives the user an idea of the mode accuracy. Generally you should aim at keeping this value below  $1e-4$ , although a set of modes for which it will be between  $1e-4$  and  $1e-3$  should be perfectly acceptable for most purposes. Be aware that if you are only testing the orthogonality of one strongly TE-like mode with one strongly TM-like mode, the test is unlikely to give you any useful information. Similarly: with two modes that have no spatial overlap.

Note: The overlaps are calculated numerically using the **nx,ny** grid resolution specified in the *Solver Parameters*, you therefore need to make sure these are large enough so that their contribution to the total error is negligible.

### 5.6.4 Polishing a list of modes

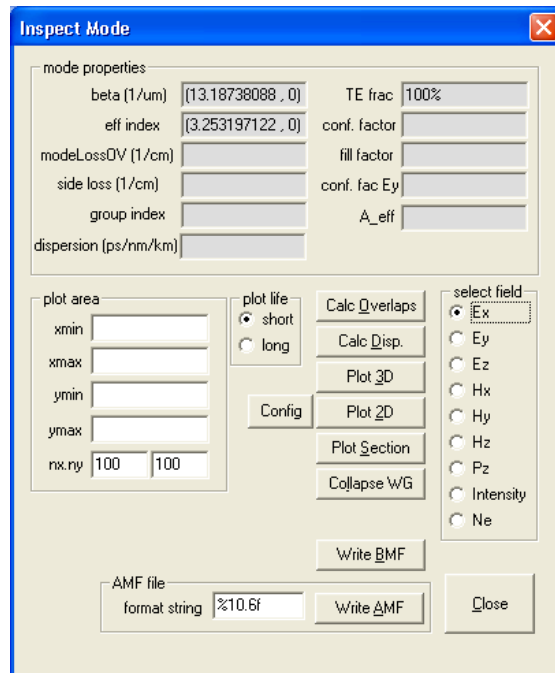
If the modes in the list are not sufficiently accurate then you may want to refine them. (e.g. by increasing the mode profile resolution)

The fastest way to do this is to use the *polish* functionality. This option allows you to change the solver parameters and then use the eigenvalues of the built modes as starting points for the iterative scheme.

To do this from the *Mode Finder* Panel, click on **Build List** and then **Builder Options** to change the relevant solver parameters. Upon clicking **ok** and returning to the *Build EV List Panel* you should check the radio box to *Polish Existing List*, and click **ok**. One option that you have here is to make permanent the changes that you made in the *MOLAB Options* by clicking **Set As Default**.

Note that if you *polish* the *Mode List* after the waveguide boundary conditions have been changed, e.g. from an Electric Wall boundary condition to Transparent, this change will also register.

## 5.7 The Inspect Mode Panel



Once you have located a mode, you can inspect the mode's 2D profile using the *Inspect Mode panel*, shown above. The top of the panel displays some basic information about the mode. Initially only the mode is shown; other parameters require the profile to be generated.

To generate a profile, the *Inspect Mode Panel* must decide a profile size and how many points to evaluate the profile. By default the profile size is defined by the waveguide's bounding rectangle. If you want to view a small part of the mode, then set the **xmin**, **xmax**, **ymin**, **ymax** values. Other parameters and features are:

**nx, ny** sets the resolution of the generated mode - a value of 60 (giving a 60x60 grid) is fine for most things.

**Config**

sets the parameters of the **Plot 2D** plot, selecting between *Contour Plot*, *Colour Map* or *Vector Plot* setting and the associated parameters.

**select field**

selects the vector field component that will be plotted. Pz is the power flux in z-direction. I is the field intensity (energy density). Ne is the refractive index profile.

**plot life**

a *short* life plot is superseded when a new plot is made. A *long* life plot will exist until you close it or exit the program.

**Calc overlaps**

calculates the mode properties, entering the results in the *Inspect Mode Panel* (see below).

**Calc Disp.**

calculates the dispersion and the group index of the mode, entering the results in the *Inspect Mode Panel* (see below).

**Plot 3D**

generates a 3D (mesh) plot of the selected component. You can alter the viewing angle of this plot from the plot's */Options/3D viewpoint* menu.

<b>Plot 2D</b>	generate a <i>Contour Plot</i> , <i>Colour Map</i> or <i>Vector Plot</i> of the selected field component, as chosen in <b>Config</b>
<b>Plot section</b>	use this to plot or export a horizontal or vertical section through the mode. You can get very high resolution using this, which enables you to zoom in.
<b>Collapse WG</b>	a utility to generate a 1D effective index profile from the 2D waveguide; see below.
<b>Write BMF</b>	generates a <i>Binary Mode File</i> (BMF) containing full vectorial information of the mode, using the current xmin, xmax, ymin, ymax, nx, ny settings. You should set its filename to a valid name first. Contact Photon Design if you require the format of this file for connection to other programs etc. We can also provide C++ source code for reading the file – see <code>examples\SourceCode\vmpb.cpp</code> .
<b>Write AMF</b>	generates a <i>ASCII Mode File</i> (AMF) containing full vectorial information of the mode. This is a direct text version of the BMF file described above. See §17.5 for more details.
<b>format</b>	enter a C-style format string to control how numbers are written to the ASCII file. See §17.5 for more details on the syntax.

#### The Mode Properties

<b>TE frac</b>	The fraction of the Poynting vector with horizontal Electric field, i.e. the fraction of the Poynting vector given by $E_x$ , $H_y$ (See also §17.2.4).
<b>conf. factor</b>	This is the confinement factor, defined over the region specified by the <b>csfeg</b> box(es) (see §5.1.9) in the <i>SWG Editor</i> . See the note <u>Integral Resolution</u> below. (see §17.2.2 for the mathematical definition).
<b>fill factor</b>	This is a measure of the fraction of the mode power flux defined over the region specified by the <b>csfeg</b> box(es) (see §5.1.9) in the <i>SWG Editor</i> . See the note <u>Integral Resolution</u> below. (see §17.2.3 for the mathematical definition).
<b>modeLossOV</b>	This is the overlap integral between the material loss profile and the mode intensity. See the note <u>Integral Resolution</u> below. (See also §17.2.1)
<b>side loss</b>	Not supported by current version of PICWAVE
<b>conf. fac Ey</b>	This is a confinement factor estimation that includes just the $E_y$ component of the field. (the quantity <b>gammaEy</b> in §17.2.2)
<b>dispersion</b>	Not supported by current version of PICWAVE
<b>group index</b>	Not supported by current version of PICWAVE
<b>A_eff</b>	(see §17.2.6 for definition)

Integral Resolution Most of these factors are currently calculated from the mode generated on the rectangular grid defined by the **nx, ny** values in the Inspect Mode Panel. There will be an error if one of the grid points falls very close to a corner in the refractive index, where the field can become very high. The integrations attempt to increase the accuracy of the integration even for coarse grids by estimating the fields between the grid points. However these estimations do not work well at a boundaries with materials of negative dielectric constant such as a metal. In all cases, increase **nx, ny** to get higher accuracy.

#### Collapse WG Routine

This routine may be used to generate a 1D effective index profile from the rectangular 2D waveguide and the 2D mode. The propagation constant and profile of the solution to the 1D mode should approximate those of the rigorous 2D solution. The success of this approximation will largely depend on the 2D waveguide - some will “collapse” more reliably than others.

The start-point, end-point and number of points of the generated profile are defined by the values of **xmin, xmax**, (or **ymin, ymax**) and **Gridsize** on the Inspect panel.

You can collapse the 2D waveguide either into a 1D horizontal line or a 1D vertical line. The 1D index profile is calculated by “weighting” the 2D waveguide index profile with the 2D mode profile and the routine also gives you the opportunity to collapse the 2D mode to a 1D mode before generating the 1D waveguide. For example, if you were generating a horizontal 1D index, the routine would first generate a 1D vertical mode profile by integrating-out the x-dependence, this vertical 1D mode profile would then be used to weight the 2D index distribution at each x-position. You will in general find that using a 1D mode profile (i.e. selecting *collapsed mode*) gives a more consistent 1D effective index distribution. But you should remember that none of these routines are mathematically rigorous in their foundation: which settings are best for your problem must be determined by experiment.

Note that while the horizontal collapse will cope accurately with thin layers in the vertical direction, the horizontal collapse may miss thin slices because it integrates horizontally along a line of equally spaced points - you may need to set a large **Gridsize** to cope with thin slices.

Here is the mathematical definition of the collapse (for the horizontal case):

$$\text{mode profile : } A(x, y) = \sqrt{P(x, y)}, \quad \text{P is mode power flux}$$

$$\text{2D index profile : } ne(x, y)$$

$$\text{generated index : } ne(x) = \frac{\int_{-\infty}^{+\infty} ne(x, y).B(x, y).dy}{\int_{-\infty}^{+\infty} B(x, y).dy}$$

$$\text{where } B(x, y) = A(x, y), \quad \text{2D mode not collapsed}$$

$$B^2(x, y) = B^2(y) = \int_{-\infty}^{+\infty} A^2(x, y).dx, \quad \text{collapsed 2D mode}$$

## 5.8 The FDM Mode Solver

The *FDM Solver* incorporates the advantages of finite-difference technique used as a mode solver into PICWAVE. In PICWAVE, this solver is implemented for RWG waveguides, in its real version (as opposed to its complex version). The *real FDM Solver* models both *real* materials (i.e. not lossy) and anisotropic dielectric tensors (diagonal tensor).

The *FDM Solver* implements a full-vectorial finite-difference technique making use of advanced techniques to improve accuracy. This allows it to accurately model waveguides with high-step refractive index profiles. It will also take into account features smaller than the grid size. Even waveguides that combine both of these features can be effectively modelled using the technique.

The real version *FDM Solver* can be used with electric/magnetic walls and impedance boundary conditions.

In addition to the previously implemented fully vectorial 3D *FDM Solver*, the semi-vectorial TE and TM options are also available. Also, as with the *Effective Index Solver*, one-dimensional options of the *FDM Solver* are provided, allowing efficient and accurate modelling of slab structures.

### 5.8.1 Options

<b>nx, ny</b>	The FDM Solver resolution is now defined by the MOLAB parameters <b>nx</b> , <b>ny</b> in the MOLAB parameter window. The horizontal grid size is given by (window-width/ <b>nx</b> ). The vertical grid size is given by (window-height/ <b>ny</b> ). The grid is uniformly spaced in each direction.
<b>HCurv</b>	Not supported by current version of PICWAVE
<b>Hsymmetry</b>	horizontal symmetry planes exploited in structure: see §5.8.2
<b>Vsymmetry</b>	vertical symmetry planes exploited in structure: see §5.8.2
<b>Dimension</b>	3D, <i>Quasi2D-XZ</i> , <i>Quasi2D-YZ</i> (See below.)

The *3D* option will attempt to solve a general waveguide.

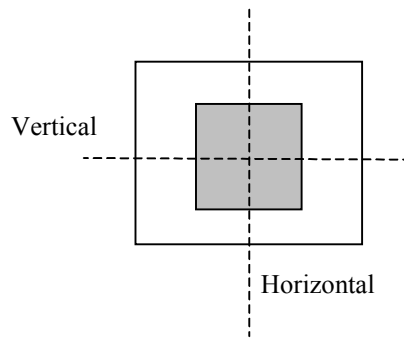
*Quasi2D-XZ* assumes that the device has no height – each slice should have only one layer and the boundary conditions on the top and bottom are ignored.

*Quasi2D-YZ* assumes the device has no width – there should be only one slice in the structure and the boundary conditions on the left and right are ignored.

### 5.8.2 Using Symmetry Planes

If the waveguide has a symmetry plane in x, and/or y, then the *FDM Solver* can take advantage of this to reduce computation time and increase the accuracy of the 2D modes. This can be set in the *FDM Solver Options*.

Vertical Symmetry and Horizontal Symmetry are defined as below:



When using the symmetry plane in the vertical direction, the same accuracy will be achieved with only half the number of 1d modes. The calculation time for finding the 2D modes is proportional to this number cubed; thus, the computation will be 8 times faster.

Using the symmetry plane in the horizontal direction reduces the calculation time by a factor of two. This is because the number of vertical slices considered in the calculations will be halved.

You should also find that when using symmetry planes, the solver will be more robust in the cases of degeneracy.

However, care must be taken when using this functionality, as no checks are performed to determine if the structure is indeed symmetric. The *FDM Solver* will solve the waveguide that results from the left-hand side, or bottom half (or bottom left corner of the structure) flipped over the relevant symmetry plane(s). This is done regardless of whether the symmetry actually exists.

The use of horizontal and vertical symmetries by the *FDM Solver* is controlled by the parameters **Hsymmetry**, **Vsymmetry** respectively. These can be set to:

- *None* - no symmetry reduction is used
- *ExSymm* - the solver is instructed to find modes with symmetric Ex field.
- *EySymm* - the solver is instructed to find modes with symmetric Ey field.
- *Both* – the solver is instructed to find modes with both types of symmetry.

### 5.8.3 Applications

The *FDM Solver* is ideally suited to the following sorts of waveguides:

- Waveguides with high-step refractive index profile

Other applications are not yet supported or pertinent to PICWAVE

### 5.8.4 Capabilities and Limitations

- The real *FDM Solver* is limited to modelling real materials (not lossy materials)
- It is able to find many modes simultaneously.
- It uses second order finite difference for accuracy.
- It can handle anisotropic materials for a diagonal tensor other terms zero.



- It uses an evenly spaced grid. This means it is not efficient for modelling structures that combine large uniform regions with smaller regions of fine features that would require a fine grid.

#### 5.8.5 Notes

- All modes are found internally before being placed into the *Mode List*, therefore when finding 40 modes, say as opposed to 20 the time taken to plot the first mode in the *Mode List* is longer.
- The grid size of the *FDM Solver* mesh is adjusted by the *MOLAB* parameters “mode profile resolution” (**nx**, **ny**) and should be chosen to sufficiently resolve the field oscillations across the computational window. It implies that higher order modes as well as modes in high-step refractive index structures may need finer resolution to achieve the same accuracy. Typically for high contrast waveguides a grid size of about 0.01 micrometers will result in a relative effective index error for the fundamental mode of about  $10^{-5}$ .
- If the waveguide has a symmetry plane in the x-, and/or y-direction, then the *FDM Solver* can take advantage of this to reduce computation time and increase the accuracy of the 2D modes. Additionally, the symmetry planes can be used for the 1D+Z *FDM Solvers* when appropriate.

#### 5.8.6 The Molab and the FDM Solver

Internally, the *FDM Solver* finds modes about a chosen *seed* effective index. So if you ask for 20 modes it will find the 20 modes nearest the seed. If both the min and max neff/beta limits of the scan range have been set on the *MOLAB Options* panel then the *FDM Solver* will set the *seed* to the mid point of this range. Thus, for example, if you are modelling an air-core waveguide, the range 1.2-0.8 (say) is more likely to find the fundamental leaky mode than the range 1.0 – 0.0. If just a maximum neff/beta has been set in the *MOLAB Options* then the *FDM Solver* will set the *seed* to this maximum value. Eigenvalues that it finds above the value will not be added to the eigenvalue list, so you will in general get fewer eigenvalues than given by the **max Nmodes** parameter.

## 5.9 The Effective Index Solver

The 3D version of the *Effective Index Solver* uses the well-known Effective Index Approximation. It is very fast and works adequately for waveguides:

- With a small *delta-n* (refractive index contrast)
- Where the field can be written in the form:  $\underline{E}(x, y) = \underline{A}(x) \cdot \underline{B}(y)$

In the current version of PICWAVE it is implemented only for real refractive index profiles.

In the 3D version it works as follows: Assuming the waveguide is divided up into M slices as in the RWG structure, with each slice invariant in the x-direction, then first we find the fundamental 1D mode of each slice which we denote  $\phi_m(y)$ . This gives us the effective indices of each 1D mode  $n_{\text{eff},m}$  in slice m. Now we construct a 1D waveguide in the x-direction, using the  $n_{\text{eff},m}$  creating a waveguide  $n_{\text{eff}}(x)$ . Finally we solve for the

fundamental mode of  $n_{\text{eff}}(x)$  which we denote  $\psi(x)$ . Then the 2D mode profile in slice  $m$  is given by:

$$\Phi_{(x,y)} = \psi(x) \cdot \phi_m(y)$$

In the 2D (or “Quasi2D”) versions this is just a slab mode solver – where the structure varies in only one direction - which case we can solve very quickly and very accurately.

### 5.9.1 Options

**Dimension**                    *3D, Quasi2D-XZ, Quasi2D-YZ* (See below.)

The *3D* option will attempt to solve a general waveguide.

*Quasi2D-XZ* assumes that the device has no height – each slice should have only one layer and the boundary conditions on the top and bottom are ignored.

*Quasi2D-YZ* assumes the device has no width – there should be only one slice in the structure and the boundary conditions on the left and right are ignored.

## Gratings

PICWAVE includes an extensive *grating* model. There are two ways of describing a *grating* in your device:

1. define the physical shape of the grating using *grating profile* in a *grating set*. The program will then use the *Kappa Calculator* to compute coupling and other coefficients for the *grating*. Together, these coefficients are referred to as **CouplingParms**. This method can only be used for sections whose cross-sections are defined by *RWGs*.
2. Enter the **CouplingParms** coefficients directly for each section.

**CouplingParms**, common to both these methods, are discussed below. Additional related details on chirped gratings and grating phase are given in §6.2 and 6.3. Finally the *grating sets* are discussed in §6.4.

## 6.1 CouplingParms

The coupling parameters, **CouplingParms**, are either defined manually for a given section (see §4.9.1.3) or calculated automatically from a physically defined grating (*grating profile*) when a simulation is run. In the current version of PICWAVE the same **CouplingParms** are used for *all* modes propagating through a section. The **CouplingParms** are as follows:

<b>gOrder</b>	the <i>grating order</i>
<b>gPeriod</b>	[ $\mu\text{m}$ ] the <i>grating period</i> – at the moment, we assume that all grating profiles in a <i>section</i> have the same period
<b>gPhase</b>	[deg] the <i>grating phase shift</i> – this can be conveniently used to change the phase of a grating. This is relative to the phase of the grating in a previous section so that if gratings in two neighboring sections have the same <b>gPhase</b> value then there is no phase jump at the join.
<b>realKappa</b>	[ $1/\mu\text{m}$ ] the real coupling coefficient
<b>lossKappa</b>	[ $1/\mu\text{m}$ ] the loss coupling coefficient.
<b>gainKappa</b>	[-] the gain coupling coefficient; gain coupling is linked to the gain of an active layer - see below.
<b>gainVolFrac</b>	the fraction of the grating layer that hasn't been etched away. The program takes this factor into account when determining how many carriers there are in an active layer with a grating.
<b>alGainKappa</b>	the active layer that is associated with the <b>gainKappa</b> .

Gain coupling is given by:

$$dA/dz = K_g * B \quad A, B \text{ are forward and backward mode amplitudes}$$

$$-dB/dz = K_g * A$$

$$K_g = \text{gainKappa}(\Gamma \cdot G(N_e) / 2),$$

where  $G(N_e)$  is the power material gain at carrier density  $N_e$ , and  $\Gamma$  is the confinement factor of the whole layer (even if most of it is etched away to form the grating).

**chirpFrac** [-] if you want to *chirp* the period of your grating, this parameter is used to control the magnitude of the period chirping for the section (see below).

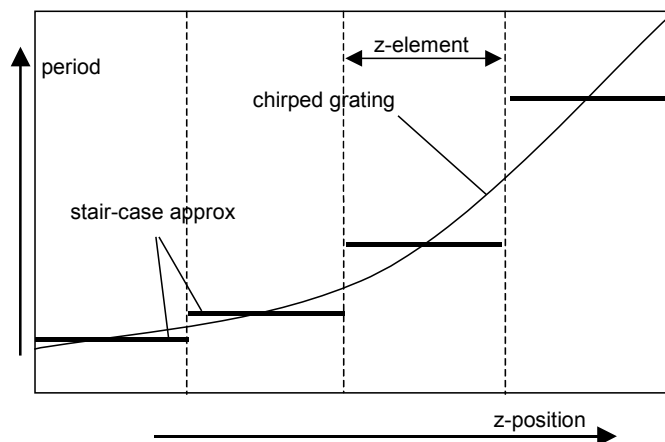
**chirpPower** [-] if you want to *chirp* the period of your grating, this parameter is used to control variation of the period with the fractional position along the section (see below).

## 6.2 Chirped Gratings

You may *chirp* the period of a grating along the length of a section using the **chirpFrac** and **chirpPower** parameters. The period may vary according to the formula:

$$\text{period} = g\text{Period} \cdot (1 + \text{chirpFrac} \cdot (z / L)^{\text{chirpPower}})$$

where  $L$  is the length of the section,  $z$  is the distance from start of the section, and other quantities are user-settable properties of the grating. The algorithm attempts to approximate the chirped grating by constant-pitch gratings within each  $z$ -element as shown in the figure below.



The algorithm attempts to ensure that the phase of the stair-case grating is the same as that of the chirped grating at the mid-point of each  $z$ -element.

## 6.3 Facet Phase and Gratings

The relative phase between a facet and a grating can have a critical influence on the performance of a DFB laser. It is therefore useful to define this relative phase exactly. This phase is defined by the **gratPhaseShift** parameter in a waveguide or taper section's properties panel.

The **gratPhaseShift** values are cumulative – if you have 3 sections, each with a **gratPhaseShift**=90° then the grating in the third section is shifted 270° from its original position.

The relative phase between the grating and the right-hand facet is dependent on the length of the cavity. Subtract the maximum number of periods from the laser length and the remaining distance will give you the phase to the right facet. I.e.




$$\text{phase} = (L - n \cdot \text{period}) / \text{period} / 2 / \pi, \quad n \text{ is largest integer s.t. } n \cdot \text{period} < L$$

[When using a *grating set* for a particular section note that, with a **gratPhaseShift** of 0, the left hand facet starts at the part of the grating shown on the left hand side of the grating profile plot in the *Grating Set Editor*.]

## 6.4 Grating Set

A *grating set* consists of a set of user-defined *grating profiles*, each of which gives a physical definition of a grating. When an *RWG* containing a layer with a reference to a *grating profile* is used in a simulation, the coupling parameters for the *grating profile* are automatically calculated.

To add an *grating set* to a *project*,

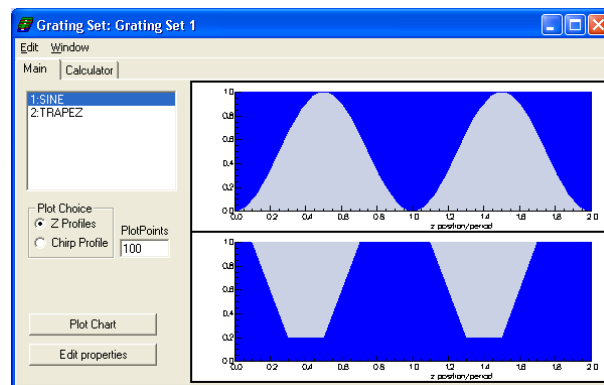
- Click on the  Project in the project tree, then click on the  Add Grating Set button *or* right-click on the  Project and then select /Add/Grating Set

To edit a *grating set*,

- Double-click on the newly added  *grating set* in the project tree. This will open the *Grating Set Editor* (see below).

### 6.4.1 Grating Set Editor

The *Grating Set Editor* is used to manage a *grating set* of one or more grating profiles. Typically you will have one *grating set* for each waveguide cross-section (*RWG*). On the left is a list of the gratings in the set. Double-click on an entry to edit its properties.



At the bottom of the window is an **Edit Properties** button – this will access the properties of the whole *grating set*, including all the *grating profiles* it contains. The *grating set* has the following properties common to all its *grating profiles*:

<b>gOrder</b>	the <i>grating order</i>
<b>gPeriod</b>	[ $\mu\text{m}$ ] the <i>grating period</i> – at the moment, we assume that all grating profiles in a <i>section</i> have the same period
<b>chirpFrac</b>	chirp parameter (see §6.3)
<b>chirpPower</b>	chirp parameter (see §6.3)
<b>allSamePeriod</b>	always true for now.

#### 6.4.2 Grating Profiles

The *grating set* may contain one or more *grating profiles*.

To add or delete a profile,

- click on the **Edit Properties** button then double-click on the **profiles** property. A list of *grating profiles* will appear, together with the properties for the selected profile. From here you can add, delete or copy a profile.

A *grating profile* may be one of:

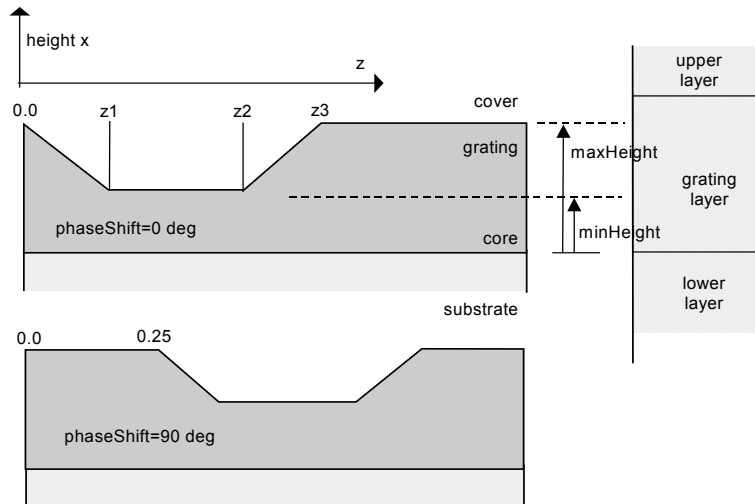
- sinusoidal
- trapezoidal. If this option is selected then the trapezoid dimensions **zg1**, **zg2**, **zg3** must be defined. These dimensions are defined as a fraction of the grating period and are described in more detail below. The trapezoid style may be used to define useful grating shapes including square, triangular and sawtooth.
- User defined. If this option is selected then the **userfile** parameter must be defined, containing the name of a file describing a user defined *grating profile*. See section §6.4.3.

Profile Properties

<b>type</b>	trapezoidal/sinusoidal/user-defined, see above
<b>phaseShift</b>	[deg] the profile may be shifted along the z-axis by altering this parameter.
<b>interfacePosn</b>	defines how the two refractive-indexes are obtained; either <i>above</i> - the refractive index below the grating surface is that of the <i>grating layer</i> and the index above the surface is that of the layer above, or <i>below</i> – the refractive index below the grating surface is that of the layer below and the index above the grating is that of the <i>grating layer</i> .
<b>minHeight</b>	[-] the distance from the bottom of the grating layer to the bottom of the <i>grating profile</i> , as a fraction of the grating layer thickness.
<b>maxHeight</b>	[-] the distance from the bottom of the grating layer to the top of the <i>grating profile</i> , as a fraction of the grating layer thickness.
<b>userfile</b>	a filename containing a user-defined <i>grating profile</i> .

**zg1, zg2, zg3** coordinates of the trapezoidal profile, as a fraction of the grating period – see figure.

The figure below shows a schematic of a grating with a trapezoidal profile, defining the **zg1, zg2, zg3** coordinates of the trapezoid and also the effect of **phaseShift**. The grating is always defined in the context of a grating layer.



Schematic showing the cross section of one period of a grating structure for a *trapezoidal* style grating shape. The definition of the trapezoid dimensions  $z_1$ ,  $z_2$ ,  $z_3$  is shown. These are all fractional values of the grating period. The grating profile is assigned to a grating layer in an *SWG*, and the grating heights are all relative to this layer thickness. The lower half shows the same profile with its phase shifted by setting **phaseShift** at 90deg.

#### 6.4.3 The user-defined grating file

This file is expected for a "user defined" grating type. The format of this file is shown below. The first line (ignoring the comments) should contain the number of grating data points defined in the file.

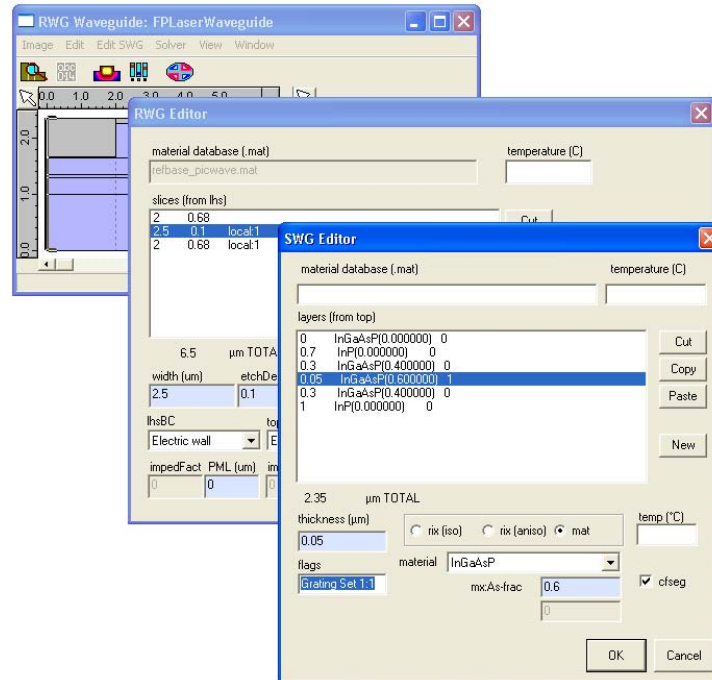
```
// user defined grating profile
// profile should be defined from z=0 to z=1, and 0<=depth<=1;
7 // number of data points
0. .0
.2 .4
.4 .2
.6 .7
.8 .9
.9 0.
1. 0.5
```

Each data point is defined by a (z,d) pair, one per line. The z co-ordinate is defined as a fraction of the grating period. The d, or depth co-ordinate is defined as a fraction of the layer thickness given in the *SWG* structure and should be in the range 0 to 1. The z values should be monotonically increasing, with a value of 0 for the first point and 1 for the last point. Vertical sides may be defined. The profile complexity is limited so that for a section in the z direction at any given depth, a maximum of 40 teeth are encountered or, in other words, the profile may not cross a horizontal line drawn from the start to the end of one grating period at any given depth more than 80 times.

#### 6.4.4 Using a grating profile to define a grating in an RWG

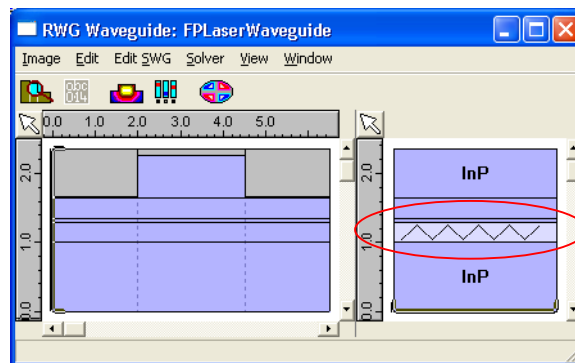
To use a *grating profile* to define a grating for a particular layer in an *RWG*,

- Open the *SWG* containing the layer, select the layer from the list, then in **flags** type the name of the *grating profile* which is to provide the physical definition of the grating, as in the example below:



The grating profile name consists of the name of grating set containing the profile following by a colon followed by the profile number as it appears in the Main tab in the *Grating Set Editor*.

Layers whose gratings have been thus defined will be indicated in the waveguide editor:



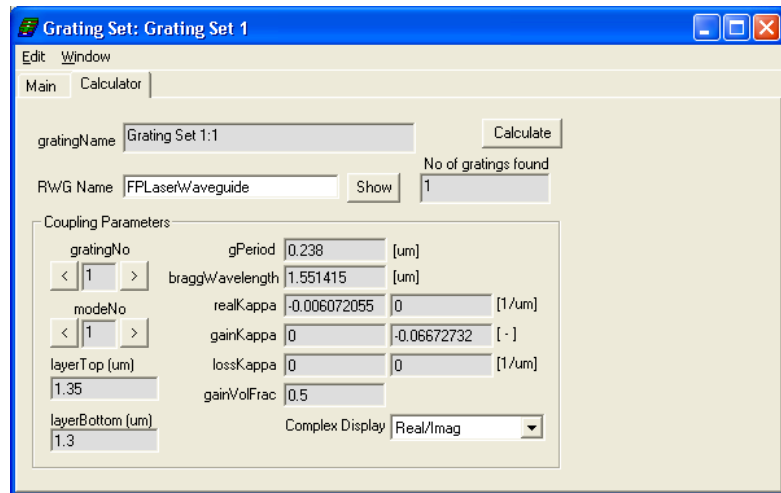
#### 6.4.5 The Kappa Calculator

This *Kappa Calculator* algorithm uses coupled mode theory to calculate the coupling parameters for a *grating profile*. The coupled mode theory is very good for small to modest refractive index steps ( $\Delta n$ ). It takes into account the polarisation of the mode (though assumes pure TE or pure TM for now, and also the lateral as well as vertical mode profile). This version of the *Kappa Calculator* does not compute radiation losses from second and higher order gratings. More detail on Kappa Calculator theory is given in §11.6.



When an *RWG* containing a grating profile reference is used in a simulation, the *Kappa Calculator* is called automatically. When the simulation is complete you will notice that the section will have an entry in its **CouplingParms** list (right-click on section and select /Edit Properties/CouplingParms). Because this **CouplingParms** entry has been added by the program, you will not be able to edit it and it will not be saved with the rest of the structure.

The *Kappa Calculator*, shown below, can be accessed directly via the calculator tab in the *Grating Set Editor* (see §6.4.5).



To use the *Kappa Calculator* proceed as follows:

1. Select a *grating profile* from the Main tab of the window.
2. Enter the name of an *RWG* waveguide in the box labelled **RWG Name**. This *RWG* should contain a layer containing a reference in **flags** to the *grating profile* you have selected, as in §6.4.4. (e.g. For the example figure above the *RWG* FPLaserWaveguide would contain the flag Grating Set 1:1)
3. Press the **Calculate** button.

The *Kappa Calculator* should then determine the coupling parameters for the *grating*, using the first mode in the *RWG*'s *Mode List*.

If more than one layer of the *RWG* contains a reference to the selected grating, then the *Kappa Calculator* will compute coefficients for each layer/grating. The *Kappa Calculator* will report the number of matching layers/gratings found in the box labelled **No of gratings found** and you can scroll between each using the **gratingNo** setting. In the current version of PICWAVE, coupling parameters are only calculated for the first active mode of the *RWG* in which the grating is used. (i.e. **modeNo** fixed at 1)

# Chapter

# 7

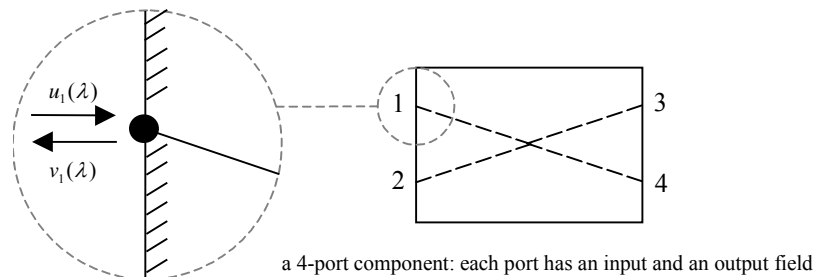
## The FIR Section

A *Finite Impulse Response Filter Section (FIR section)* is a section that can model a device with an arbitrary spectral profile. It is different from the other sections described above in that it does not have a physical size defined by the user, but instead provides a way of importing a frequency-domain model of a component from FIMMPROP or another compatible product. It will read in a wavelength dependent scattering matrix of the component and automatically generate an equivalent time-domain model for inclusion in the PICWAVE circuit simulator. This allows a wide variety of passive components to be modelled accurately in FIMMPROP and then linked into the circuit model. Bi-directionality and reflections are supported.

### 7.1 Overview and General Usage

#### 7.1.1 The Model

In a frequency-domain model of a component such as the following



the output fields  $v_j(\lambda)$  of are related to the input fields  $u_i(\lambda)$  as follows:

$$v_j(\lambda) = S_{ij}(\lambda)u_i(\lambda),$$

where  $i$  and  $j$  denote the  $i$ th and  $j$ th ports (or more accurately, specific modes within specific ports), and  $S_{ij}(\lambda)$  is the *S-Matrix spectrum*. This *S-Matrix spectrum* describes the wavelength response of the component at a number of discrete wavelengths within a specified range. It can be imported into PICWAVE using an *FIR section*, which uses the *imported S-Matrix spectrum* to generate an *FIR filter* – an equivalent time domain model of the component

In generating an *FIR filter* the *imported S-Matrix spectrum* is first sampled at  $N$  equally spaced wavelengths over the free spectral range, to produce a *sampled S-Matrix spectrum*.  $N$  is known as the *filter length* and determines the accuracy of the time domain model. An *FIR filter* computes the output fields,  $b_j[it]$ , of the component at a time step  $it$  as a function of the input fields,  $a_i[it]$ , at a *finite* number of previous time steps:


$$b_j[it] = \sum_i \sum_{k=1}^N c_{k,i,j} a_i[it - k]$$

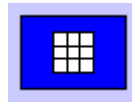
where  $i$  and  $j$  denote the  $i$ th and  $j$ th ports (or more accurately, specific modes within specific ports), and  $c_{k,i,j}$  denote the FIR coefficients describing the coupling between the  $i$ th and  $j$ th ports. The FIR coefficients are generated from a forward Fourier transform of the *sampled S-Matrix spectrum*, the number of coefficients being given by the *filter length*  $N$ .

An *S-Matrix spectrum* can be produced in FIMMPROP through the use of scripting. Instructions on how to do this are given in §7.2. Here we give a description of how an *FIR section* is set up in PICWAVE using an *imported S-Matrix spectrum* (see §7.1.4 for correct *S-Matrix spectrum* text format)

## 7.1.2 Usage

### Adding an FIR section

An *FIR section* is added to a circuit in the same manner as a waveguide section (see §4.9.1), using the  icon or *Insert/FIR section* from the menu-bar. It should appear as:



Initially there are no ports; these will appear when an *S-Matrix spectrum* has been assigned to the *FIR section*, as it is the *S-Matrix spectrum* which specifies the number of ports. The *FIR section* should be placed so that it is ready to be connected up in series with the component from which the signal to be filtered will emerge.

In the current version of PICWAVE, the *FIR section* supports only one left and one right port.

### Importing an S-Matrix spectrum

The wavelength range of the *S-Matrix spectrum* needs to be contained within, or coincide with, the **lamRange** of the TDTW simulation (see §2.1.5) so that the response of the *FIR filter* will cover all wavelengths that may be encountered in running the simulation. The simulation will not run until this requirement has been met. If you wish to use a certain **zStep** for the reference section in your device, you need to run a simulation *before* adding an *FIR section* to the device, so that the corresponding **lamRange** is calculated in the TDTW Calculator window. This will tell you the wavelength range that your *S-Matrix spectrum* must cover. A value for **zStep** cannot be defined using an *FIR section* as the reference section (see §2.1.5). Therefore a device with an *FIR section* must contain at least one other section which can be used as reference section, even if it is not connected to the *FIR section*.

Once an *S-Matrix spectrum* with a suitable wavelength range has been produced, it is assigned to the *FIR section* as follows:

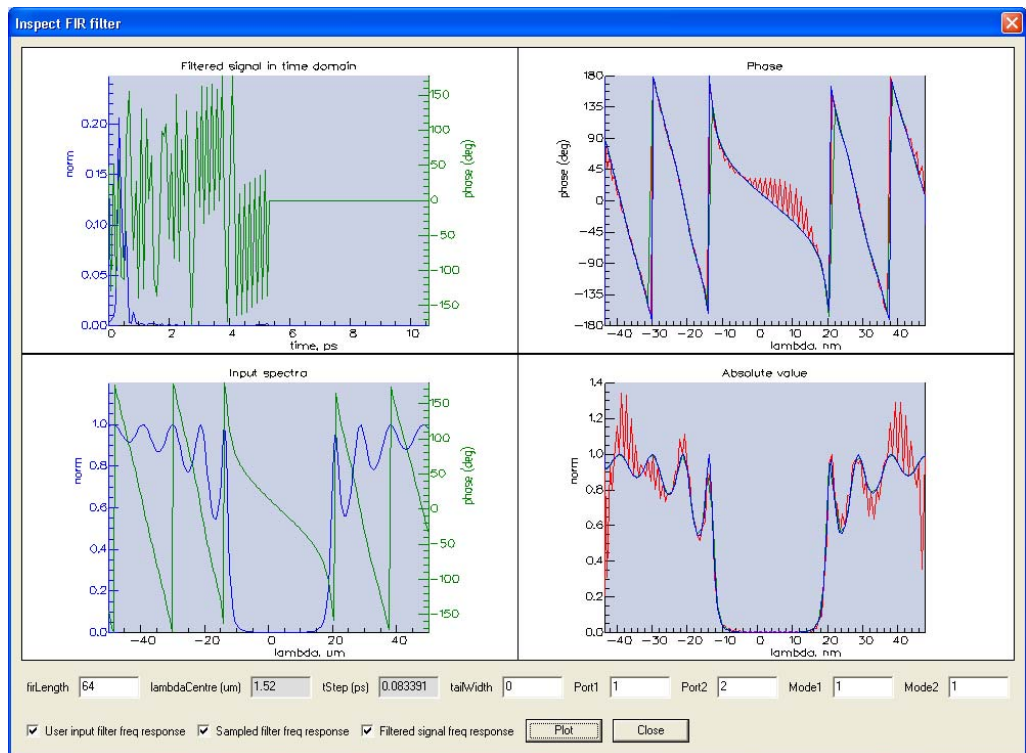
- Open the *S-Matrix spectrum* file in a text editor (e.g. notepad)
- Select all the text within the file and copy it to the clipboard
- Right-click on the *FIR section* in the PICWAVE device window and select */Paste SMatrix* from the menu

An *FIR filter* can now be generated by the *FIR section* from this *imported S-Matrix spectrum*.

## Inspecting the FIR filter

The *FIR filter* that will be generated by an *FIR section* can be inspected, before running TDTW simulation, as follows:

- In the device window, right-click on the *FIR section* and select *Inspect filters* from the menu to open the *Inspect FIR filter* window.
- Click **Plot**. (If you receive a message, your *S-Matrix spectrum* wavelength range is too narrow. As discussed above, either produce another *S-Matrix spectrum* with a sufficient wavelength range or adjust **lamRange** by adjusting **zStep** and/or **lambdaCentre** in the TDTW calculator window). The window should then display something like the following, which is an example of the FIR response of a Bragg reflector:



The element of the *S-Matrix spectrum* being plotted is determined by the port and mode indices (Port1, Port2, Mode1, Mode2) which can be set at the bottom of the window. The port and mode numbering conventions are the same those as used in specifying the element of general joint scattering matrices (see §4.10.5). In the example above, all the plots are related to the coupling between first (left-hand) and second (right-hand) ports for the first TE mode. The lower left canvas shows a direct plot of the *imported S-Matrix spectrum*, amplitude (blue) and phase (green). The canvases on the right are the amplitude and phase plots of: the *imported S-Matrix spectrum* (blue); the *sampled S-Matrix spectrum* (green); and finally the frequency response of the *FIR filter* (red) – the Fourier transform of *FIR filter* time domain response. Any of these plots can be removed by unchecking the checkboxes at the bottom of the window. The upper left canvas shows the time domain response of the *FIR filter* to an impulse (delta function) signal. i.e. a plot of the FIR coefficients as calculated from the forward Fourier transform of *sampled S-Matrix spectrum*. The plots shown here can be chosen by using the checkboxes at the bottom of the window.

Additional rapid oscillations in the filtered signal frequency spectra (red), which are not present in the *imported S-Matrix spectrum*, will be observable. These are due to the

truncation of the *imported S-Matrix spectrum* and periodic nature of the *FIR filter*. Their size can be reduced by using the **tailWidth** parameter described below.

### 7.1.3 Properties

The following properties can be edited from the *Inspect FIR filter* window or from the *FIR Section Properties* panel (right-click on the FIR section and select *Properties*):

**firLength** (*filter length*) specifies the number of FIR coefficients, or equivalently, the number of equally-spaced wavelengths at which the *imported S-Matrix spectrum* is sampled to produce the *sampled S-Matrix spectrum*. The larger the *filter length*, the better the *sampled S-Matrix spectrum* approximates the *imported S-Matrix spectrum*, but the more time-consuming the simulation. Note that it would be incorrect to make the filter length so large as to exceed the resolution of the *imported S-Matrix spectrum* itself.

**tailWidth** the *sampled S-Matrix spectrum* can be modified (distorted) to force it to be periodic over the free spectral range. This can reduce the size of additional spurious rapid oscillations in the frequency response of the FIR filter generated from the *sampled S-Matrix spectrum*. This modification is enforced within wavelength windows of identical size at each end of the free spectral range. The **tailWidth** specifies the size of these wavelength windows as fractions of the free spectral range (**lamRange**); its value must be  $<0.5$ . The remaining central part of the spectrum will be undistorted.

If the **firLength** or **tailWidth** have been changed from within *Inspect FIR filter* window, the changes can be saved or discarded when you click **Close**.

### 7.1.4 S-Matrix Spectrum Text Format

Regardless of how the *S-Matrix spectrum* is produced, its text (file) must have the following format if it is to be successfully pasted to and used with a *FIR section*:

```

begin <SMatSpectrum(1.1)> "SMatrix"
200 2 1 // nlambda s nports nspectra
1 1 1 "ID1" "descr1" // side nModesPort nTEModesPort wgID description
2 1 1 "ID2" "descr2" // side nModesPort nTEModesPort wgID description
(1, 1, 2, 1) // list matrix elements included in this file
// S-matrix spectra follow
1.47 { -0.8123068, -0.570303}
1.470391 { -0.8843665, -0.457635}
.
.
.
1.568437 { 0.9926107, -0.1195959}
1.569219 { 0.9300004, -0.3580263}
1.57 { 0.8101798, 0.5673941}
end

```

wavelengths                      complex amplitudes { Re, Im }

Signature and spectrum parameters

Line 1: signature as shown above.

Line 2: a line of 3 parameters

nlambda – number of discrete wavelength points in the spectrum

nports – number of ports

nspectra – number of scattering matrix elements included in the file

Port List

Next follows a list of ports, with a description of each port, one per line. The description of the port consists of the following parameters:

side describes what side of the component on which the port is located, 1 indicates LHS, 2 indicates RHS.

nModesPort number of waveguide modes exiting/entering the port.

nTEModesPort number of TE waveguide modes exiting/entering the port

wgID a string identifying the waveguide. The string should be enclosed in double quotes.

description a description of the waveguide The string should be enclosed in double quotes.

Element List

Next follows a list of the scattering matrix elements that are defined in this element, one element per line. Elements that are not defined are assumed to be zero. The matrix is assumed to be Hermitian so that the coupling coefficient from (port-n, mode-m) to (port-j, mode-k) is the complex conjugate of the coupling from (port-j, mode-k) to (port-n, mode-m). Each line has the following syntax:

(input-port-index, input mode-index, output-port-index, output-mode-index).

All port and mode indexes start at 1. The port and mode numbering conventions are the same those as used in specifying the element of general join scattering matrices (see §4.10.5).

Spectra data

Finally there follows the actual spectral data, one wavelength per line. Each line has the following format:

wavelength {Re(coef1),Im(coef1)} {Re(coef2),Im(coef2)} ... {Re(coefN),Im(coefN)}

where the wavelength is defined in  $\mu\text{m}$  and the scattering matrix coefficients are listed in the order they appear in the Element List.

End

The last line of the *S-Matrix spectrum* must finish with the word end on the last line.

## 7.2 PICWAVE-FIMMPROP Links: Creating S-Matrix Spectra

In FIMMWAVE, S-Matrix spectra files can be created from a FIMMPROP frequency-domain simulation of a device using the ScattMatrGenerator.py PYTHON script supplied with PICWAVE. This script accesses FIMMWAVE remotely via

TCP/IP and is fully automatic, requiring only that you specify certain *S-Matrix spectrum* parameters: usually the central wavelength, wavelength range and the number of wavelength steps.

Note the current version of this script supports only one left and one right port.

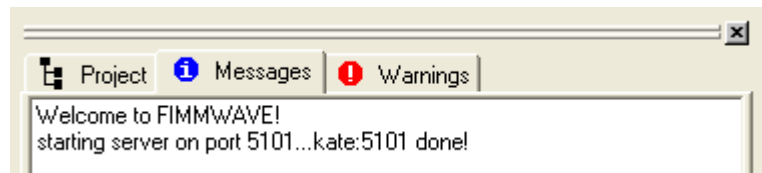
### 7.2.1 Using the Script

The first step is to open FIMMWAVE and ensure that it is able to be accessed remotely. To do this,

- Either open **Start /Run** and type in the path and filename of the FIMMWAVE executable followed by the suffix `-pt 5101` ; or if using a shortcut, modify the target (right-click **/Properties**) by adding the same suffix. In either case, the target should read `...\fimmwave.exe -pt 5101`

The suffix ensures that FIMMWAVE is serving on port 5101; the script will connect via this port. (Port 5101 is the default port number).

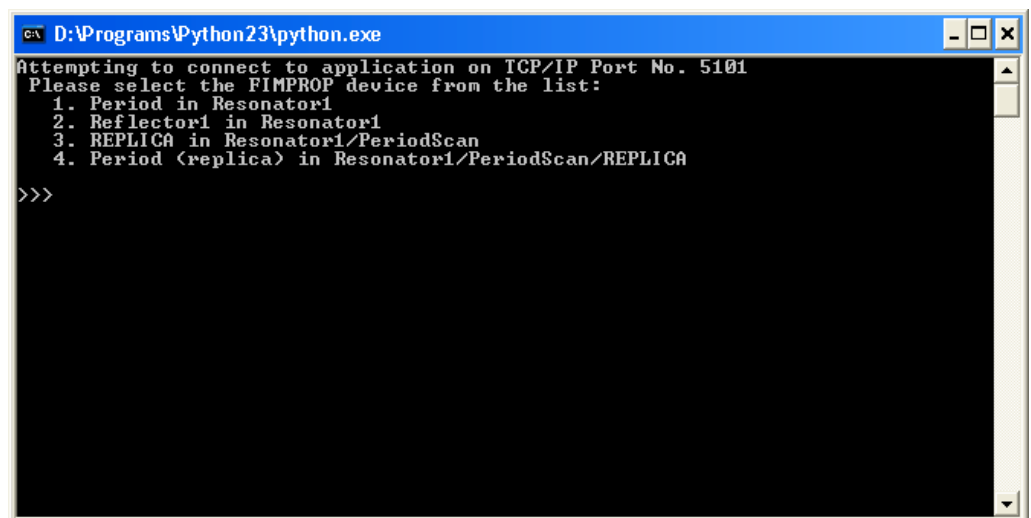
- Run FIMMWAVE using this modified path. If it has loaded successfully you will see the following message in the Messages tab of the Main Window:



- Next, open the project containing the device whose *S-Matrix spectrum* you wish to compute.

We are now in a position to run the script, so providing you have installed and set up PYTHON as detailed in §13.7.1,

- Either: select **Scripts/Run** script in the Main Window of FIMMWAVE and select and open `ScattmatrGenerator.py`; or (outside of FIMMWAVE), open the folder containing the `ScattmatrGenerator.py` script file and then double-click on the file. The script will then run via the PYTHON command line; the console window should display something like the following:

A screenshot of a Python console window titled 'D:\Programs\Python23\python.exe'. The console output shows: 'Attempting to connect to application on TCP/IP Port No. 5101', 'Please select the FIMPROP device from the list:', followed by a numbered list: '1. Period in Resonator1', '2. Reflector1 in Resonator1', '3. REPLICA in Resonator1/PeriodScan', and '4. Period (replica) in Resonator1/PeriodScan/REPLICA'. The prompt '>>>' is visible at the bottom of the console.

This menu lists all the components (in the FIMMWAVE project currently open) for which S-Matrix spectra can be computed.

- Type in the number of the device whose *S-Matrix spectrum* you wish to compute, then press Enter. This will lead to a further menu:

```

D:\Programs\Python23\python.exe
Attempting to connect to application on TCP/IP Port No. 5101
Please select the FIMPROP device from the list:
1. Period in Resonator1
2. Reflector1 in Resonator1
3. REPLICa in Resonator1/PeriodScan
4. Period (replica) in Resonator1/PeriodScan/REPLICa
>>> 2
2

Please select one of the options:
1: S(lambda0) + user supplied optical length
2: S(lambda0) + algorithm for Lopt
3: Linear extrapolation using values S(lambda0), S(lambda0 + dLambda)
4: Quadratic extrapolation using values S(lambda0), S(lambda0 + dLambda),
  S(lambda0 - dLambda)
5: Values Sj, as computed at an array of lambda_j
>>> 5

```

Option 5 will produce a rigorous *S-Matrix spectrum* for an array of wavelengths (details on the other options given here are given §7.2.2 below), so,

- Type in 5 and press Enter. You will then be asked to specify the following parameters:

- lambdaMin** [um] the minimum wavelength in the *S-Matrix spectrum*
- lambdaMax** [um] the maximum wavelength in the *S-Matrix spectrum*
- nSteps** the size of the wavelength array, i.e. the number of equally spaced discrete wavelengths in the wavelength range [**lambdaMin**, **lambdaMax**] at which the *S-Matrix spectrum* is plotted

Bear in mind that the wavelength range must meet the criteria, regarding the free spectral range of the PICWAVE simulation, outlined above in §7.1.2.

Once these have been entered (press <ENTER> after each one), the script will then run a FIMMPROP simulation which will calculate the complex transmission amplitude of the specified device for each of the wavelengths in the array. This comprises the *S-Matrix spectrum*, which will be printed in the console window.

Once the *S-Matrix spectrum* has been created and displayed, you will be asked if you want to save the data to a file. The script automatically copies the *S-Matrix spectrum* data to the clipboard so you can either: paste it manually to a file, paste it directly into PICWAVE (as described in §7.1.2) or have the script write it to a file. If the latter, type in the full path and filename as prompted (\*.txt would be an appropriate file format).

## 7.2.2 S-Matrix Spectrum Script Options

After you have selected from the list provided by the script the device whose *S-Matrix spectrum* you wish to compute, the script will present a list of five options regarding how the *S-Matrix spectrum* is to be produced. The fifth option, described in §7.2.1 above, produces an *S-Matrix spectrum* that most accurately describes the wavelength response of the device, which is calculated rigorously by FIMMWAVE at each of the discretised wavelengths within the specified range. This option will be used ordinarily when the wavelength response of the device has a high-order wavelength dependence, such as the Bragg reflector example in §7.1.2. The first four options should generally be used for devices with a wavelength response that has a second or lower order wavelength dependence. The options are as follows:



1. **"S(lambda0) + user supplied optical length"** – input parameters are: the starting wavelength **lambda0** [um], the optical length **Lopt** [um], the discrete wavelength spacing **dLambda** [um], and the number of discrete wavelengths **nSteps** (as defined above in §7.2.1). With this option FIMMWAVE calculates the transmission amplitude of the device at the starting wavelength,  $S(\lambda_0)$ , and the script computes the *S-Matrix spectrum*  $S(\lambda_j)$  according to:

$$S(\lambda_j) = S(\lambda_0) \cdot \exp(2\pi \cdot \text{Lopt} / \lambda_j),$$

where  $\lambda_j$  is an array of equally spaced wavelengths in the range **[lambda0, lambda0+nSteps\*dLambda]**, where  $j = 1, \dots, \text{nSteps}$ .

N.B. should only be used when the device has a constant or slowly varying wavelength response, such that the main overall wavelength dependence comes mostly (or exclusively) from the propagation along the optical length

2. **"S(lambda0) + algorithm for Lopt"** [NOT IMPLEMENTED]
3. **"Linear extrapolation using values S(lambda0), S(lambda0+deltaLambda)"** – input parameters are: the first wavelength **lambda0** [um]; the difference **deltaLambda** [um] between the first and second wavelengths, where the second wavelength  $\lambda_1 = \lambda_0 + \text{deltaLambda}$ ; and **dLambda** [um] and **nSteps**, both as defined above in option (1). With this option FIMMWAVE calculates the transmission amplitudes of the device at the first and second wavelengths,  $S(\lambda_0)$  and  $S(\lambda_1)$ , and the script calculates the extrapolated *S-Matrix spectrum*  $S(\lambda_j)$  according to the linear extrapolation formula:

$$S(\lambda_j) = S(\lambda_1) + [S(\lambda_0) - S(\lambda_1)] * [\lambda_j - \lambda_1] / [\lambda_0 - \lambda_1]$$

Where  $\lambda_j$  is an array of equally spaced wavelengths in the range **[lambda0, lambda0+nSteps\*dLambda]**, where  $j = 1, \dots, \text{nSteps}$ .

4. **"Quadratic extrapolation using values S(lambda0), S(lambda0+deltaLambda), S(lambda0 - deltaLambda)"** – as option (3), but this option does a quadratic fit of the transmission amplitudes of the device calculated by FIMMWAVE at three specified wavelengths: **lambda0 - deltaLambda**, **lambda0**, **lambda0+deltaLambda**. The script then uses the fitted quadratic formula to calculate an extrapolated *S-Matrix spectrum*.
5. **"Values Sj, as computed at an array of lambdaaj"** – with this option FIMMWAVE calculates the transmission amplitude rigorously at *each* of the discrete wavelengths in the specified array. This option was discussed in §7.2.1, above.

## Multiple Lorentzian Gain Fitter

The *Multiple Lorentzian Gain Fitter* is a tool which allows one to fit imported material gain data to produce a *multiple Lorentzian gain model* for use in a TDTW simulation. Provided such material gain data is available, it thus provides a straightforward means of producing a gain model without the need for manual calibration.

A material gain spectrum can be well approximated by a single Lorentzian in the region of the gain peak. This method is employed when defining a gain model in the material database system (§10.1.3) and is adequate when modelling lasers where one is concerned only with the gain peak region. When one wishes to work away from the gain peak, perhaps when working with amplifiers for example, the single-Lorentzian approximation becomes inadequate. A multiple Lorentzian model allows the gain in regions away from the gain peak to be modelled more accurately, thus enabling more accurate broadband simulations. (Single and multiple Lorentzian model are discussed in §11.4.)

### 8.1 Overview

To produce a *multiple Lorentzian gain model* for a given material, material gain data must be imported. Such data is imported from an external text file via the material database file containing the definition of the material. A gain model can then be fitted for this material in any PICWAVE circuit where it is used as an active layer material (in an RWG used to define the cross-section of an active waveguide). The imported data will consist of a set of material gain spectra for a corresponding set of carrier densities, at one or more temperatures. To produce a gain model, the material gain spectra are fitted with a number of pseudo-Lorentzians according to user-defined parameters. The gain model produced will describe the gain at any carrier density from zero to the maximum imported carrier density (at least) and at any temperature within the temperature range of the imported data. Extrapolation allows the gain model to be extended to higher carrier densities.

Of the active layer materials used in a given circuit, all those which use imported gain spectra data can have their gain models fitted in the *Multiple Lorentzian Gain Fitter* tool opened from the circuit's device window. A *multiple Lorentzian gain model* for a particular material is specific to the circuit simulation in which it is used, as the imported gain spectra are fitted only over the free spectral range, which is determined by the **tStep** and **lambdaCentre** defined in the circuit's *TDTW Calculator* (see §9). Gain models that have been fitted for a given circuit become automatically out-of-date if **tStep**, **lambdaCentre** or the imported spectra data are changed. Out-of-date gain models must be re-fitted before they can be used in a TDTW simulation.

A description of the *Multiple Lorentzian Gain Fitter* tool interface is given in the following section; a guide on how to import gain data and produce a gain model is given in §8.3.

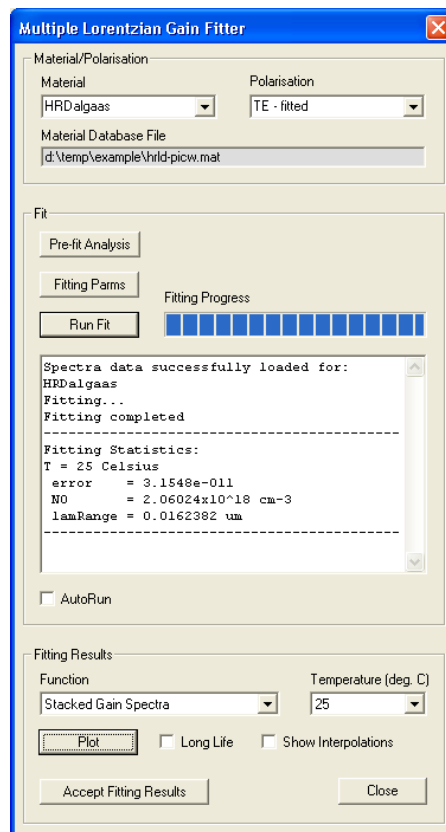
An example project using a *multiple Lorentzian gain model* can be found on the PICWAVE CD.

## 8.2 Multiple Lorentzian Gain Fitter Tool

The *Multiple Lorentzian Gain Fitter* tool is accessed from the **TOOLS** menu in a circuit's device window.

### 8.2.1 Main interface

The main interface of the *Multiple Lorentzian Gain Fitter*, shown in the figure below, is divided into three sections which correspond to the stages in the process of fitting a gain model: select material/polarisation, set-up and run fit, inspect fitting results.



#### Material/Polarisation

Here one selects the material/polarisation for which a gain model is to be fitted or for which a gain model has been fitted previously. Gain models that have been fitted previously can be viewed using the Fitting Results section (see below), provided they are not out of date.

<b>Material</b>	material for which gain model is to be fitted
<b>Polarisation</b>	polarisation of gain model <i>TE/TM</i> , each polarisation is followed by a description stating if the model <i>needs fitting</i> , <i>needs re-fitting</i> (i.e. out-of-date), or is <i>fitted</i> (up-to-date)
<b>Material Database File</b>	(read only) path and filename of material database file containing definition of <b>Material</b>

## Fit

From here one can inspect the imported spectra data (*Pre-Fit Analysis*), set the *gain fitting parameters* which control the fitting of imported spectra, and start the fitting process.

<b>Pre-Fit Analysis</b>	opens the <i>Pre-Fit Analysis</i> window, see §8.2.2
<b>Fitting Parm</b> s	open the <i>Gain Fitting Parameters Panel</i> , see §8.2.3
<b>Run-Fit</b>	starts the fitting process
<b>AutoRun</b>	if enabled, any fitted gain models which become out-of-date will be re-fitted automatically when the TDTW simulation is started

The fitting log text box gives details of the fitting progress and displays error messages should the fitting fail. Upon completion of fitting the log will display Fitting Statistics, for each of the temperatures at which gain spectra were fitted, these include:

error	sum of squared fitting errors at all sampled wavelengths for all fitted spectra at the given temperature (fitting error is the gain difference between an imported raw spectrum and a fitted spectrum at a given sample wavelength); this is a indicator of a fitting accuracy – typically of order 1e-10, the smaller the number the greater the overall fitting accuracy, positive exponents usually indicate a severe but non-fatal problem in the fitting process
N0	[cm <sup>-3</sup> ] transparency carrier density
lamRange	free spectral range expressed as wavelength range (same for all temperatures)

## Fitting Results

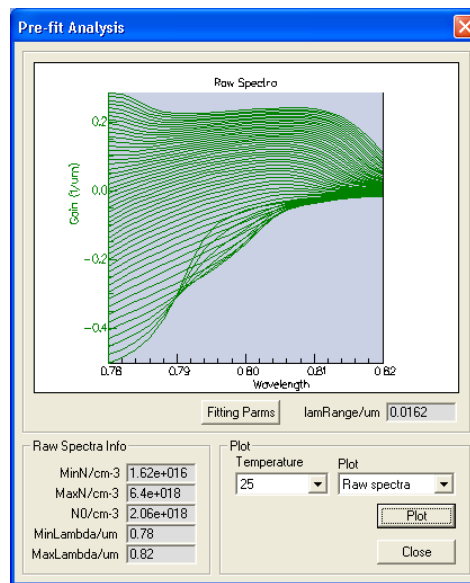
These are enabled/used once a gain model has been fitted or if the gain model has already been fitted, in which case they will be enabled once the material and polarisation have been selected.

<b>Function</b>	fitting results function to be plotted at given <b>Temperature</b> :
	<i>Gain Peak</i> - the maximum gain value in the spectrum as a function of carrier density (above transparency),
	<i>Gain Curvature</i> - gain curvature at the gain peak as a function of carrier density (above transparency),
	<i>Lambda Peak</i> - gain peak wavelength position as a function of carrier density (above transparency),
	<i>Stacked Gain Spectra</i> – gain spectra at all carrier densities plotted on a single graph: fitted spectra (blue) and extrapolated spectra (grey) are overlaid on imported spectra (green).

<b>Temperature</b>	temperature for which gain fitting results are to be plotted
<b>Plot</b>	plots the chosen <b>Function</b> at the chosen <b>Temperature</b>
<b>Show Interpolations</b>	if enabled the chosen <b>Function</b> is also plotted at interpolated carrier densities lying between imported carrier densities
<b>Accept Fitting Results</b>	accept the latest fitting results as the gain model for the chosen <b>Material</b> and <b>Polarisation</b>

### 8.2.2 Pre-fit Analysis

The *Pre-Fit Analysis* window, shown below, allows one to see the imported spectra that are to be fitted and lists some important figures relating to them so that an informed choice of *gain fitting parameters* (see below) can be made. It also allows one to view the portion of the imported spectra that can be fitted (i.e. free spectral range window)



<b>Fitting Params</b>	opens the <i>Gain Fitting Parameters Panel</i> (also accessible from Multiple Lorentzian Gain Fitter main window)
<b>lamRange</b>	[um] free spectral range expressed as a wavelength, as shown in <i>TDW Calculator</i> (calculated from the <b>tStep</b> and <b>lambdaCentre</b> )
<b>Temperature</b>	temperature for which material gain spectra are to be plotted
<b>Plot</b>	plot choice:  <i>Raw spectra</i> – plot the raw imported spectra over their full wavelength range  <i>Sampled spectra</i> – plot the portions of the spectra lying within the bounds of the free spectral range
<b>MinN</b>	[cm-3] minimum imported carrier density
<b>MaxN</b>	[cm-3] maximum carrier imported density
<b>NO</b>	[cm-3] transparency carrier density as determined from imported spectra

<b>MinLambda</b>	[um] lower wavelength bound of imported spectra data (same for all temperatures and carrier densities)
<b>MaxLambda</b>	[um] upper wavelength bound of imported spectra data (same for all temperatures and carrier densities)
<b>Plot</b>	plots spectra at all carrier densities on one graph

### 8.2.3 Gain fitting parameters

The *Gain Fitting Parameters Panel*, shown below, allows one to specify the *gain fitting parameters* which control the fitting process. These are used to determine such things as: the carrier density range of the gain model, the general fitting accuracy, and the balance between fitting accuracy at the gain peak and fitting accuracy away from the gain peak. The *gain fitting parameters* also allow one to scale/adjust the imported data and to extrapolate beyond it (to higher carrier densities).

Parameter	Value	Units
<b>TDCalculator Settings (read only)</b>		
tStep	0.00014272347	ns
lambdaCentre	0.805	um
<b>Basic Settings</b>		
EndN	5e+018	cm <sup>-3</sup>
<b>Fit Improvement</b>		
NumLoopSteps	5	
NumLambdaSamples	100	
FitFrac	0.7	
PeakFitFrac	0.2	
PeakCurvWeight	1	
GivePriorityToPeak	False	
<b>Extrapolation Settings</b>		
NumExtrapPoints	5	
ExtrapDownStartN	0	cm <sup>-3</sup>
ExtrapolateUpwards	False	
ExtrapUpStartN	5e+018	cm <sup>-3</sup>
ExtrapAndRefit	False	
<b>Scaling Options</b>		
GainScale	1	
NScale	1	
LambdaShift	0	um

The *gain fitting parameters* are as follows:

#### TDCalculator Settings (read-only)

These read-only settings correspond to **tStep** and **lambdaCentre** that are set in the *TDTW Calculator* window. They determine the free spectral range i.e. the wavelength range in which the imported spectra can be fitted.

<b>tStep</b>	[ns] time step for TDITW simulation
<b>lambdaCentre</b>	[um] central wavelength of TDITW simulation

#### Basic Settings

These settings specify a carrier density range in which to fit imported spectra. Only those imported spectra at carrier densities within this range will be fitted.

<b>EndN</b>	[cm <sup>-3</sup> ] upper bound of carrier density range
-------------	--

## Fit Improvement

These settings allow the fitting accuracy to be adjusted, either, generally, in the gain peak region, or away from the gain peak region.

- NumLoopSteps** controls the number of loops in which the fitting algorithm alternates between fitting the whole spectrum at once and fitting the gain peak region; generally, too small a number may lead to the occurrence of fitting anomalies resulting from fitting one part of spectrum at the expense of another
- NumLambdaSamples** the number of wavelengths at which an imported spectrum is sampled before it is fitted (n.b. imported spectrum data is fitted with a spline function before being sampled)
- FitFrac** central fraction of the free spectral range defining the fitting region; parts of the spectrum lying outside this region will be ignored in the fitting process
- PeakFitFrac** as **FitFrac** except for purposes of fitting the gain peak region
- PeakCurvWeight** weighting factor that determines the weight placed on the accuracy of the fitting of the gain curvature at the gain peak; a value of zero means that gain curvature at the peak is not fitted explicitly, a large value means that the algorithm will try to fit the gain peak curvature accurately even at the expense of poor fitting in other regions
- GivePriorityToPeak** if true, a given spectrum will be fitted as though its gain peak (if there is one) were at the centre of the free spectral range. This ensures that the fitting accuracy is good around the gain peak and is particularly useful when the gain peak lies away from the centre of the free spectral range

## Extrapolation Settings

These settings control extrapolation, which allows one to use a series of fitted spectra to produce a spectrum at the next carrier density and so on. Extrapolation can thus be used to produce gain spectra for carrier densities lying beyond the range of imported carrier densities. It can also be used to produce spectra at carrier densities for which imported spectra are available e.g. when a second gain peak is present which might upset the fitting process, or when one wants to use extrapolated spectra as initial starting guesses for subsequent fitting (**ExtrapAndRefit** – see below) which might give more satisfactory results in certain cases.

Upward extrapolation is used between the carrier densities **ExtrapUpStartN** (see below) and **EndN** (see Basic Settings, above). When extrapolating beyond the maximum imported carrier density (**MaxN**, given in *Pre-Fit Analysis* window), extrapolated spectra will be produced at carrier density intervals equal to that between the two highest imported carrier densities. Otherwise, imported carrier density values will be used when available.

Downward extrapolation is always enabled as the gain model must define the gain down to a carrier density of zero. It will therefore always produce an extrapolated spectrum for a carrier density of zero, and will also produce extrapolated spectra for

any imported carrier densities lying between 0 and the lowest imported carrier density lying above **ExtrapDownStartN** (see below) .

<b>NumExtrapPoints</b>	the number of adjacent carrier densities whose spectra are used to produce an extrapolated spectrum at the next adjacent carrier density; this number cannot be more than the number of spectra that have been fitted
<b>ExtrapDownStartN</b>	[cm-3] carrier density from which to begin downward extrapolation to lower carrier densities
<b>ExtrapolateUpwards</b>	enables upwards extrapolation to higher carrier densities, from <b>ExtrapBeginN</b> up to <b>EndN</b>
<b>ExtrapUpStartN</b>	[cm-3] (enabled when <b>ExtrapolateUpwards</b> is <i>true</i> ) carrier density from which to begin upward extrapolation; this cannot be greater than maximum imported carrier density ( <b>MaxN</b> ) and must be less than <b>EndN</b>
<b>ExtrapAndRefit</b>	enables the extrapolated spectra to be re-fitted to the imported spectra (if available)

Scaling options

The following options are available to scale imported data to correct for systematic error etc

<b>GainScale</b>	factor by which to scale gain values in all imported gain spectra
<b>NScale</b>	factor by which to scale all imported carrier densities
<b>LambdaShift</b>	[um] amount by which to shift all wavelength values in imported spectra

## 8.3 Fitting a gain model

### 8.3.1 Importing gain data

In order to fit a multiple Lorentzian gain model for a particular material (and polarisation), gain data must be imported into PICWAVE. This data will consist of sets of material gain spectra for one or more temperatures, where each set contains spectra for a series of carrier densities, the carrier densities being listed along with the spectra. The data must be contained in a text file with the following format:



nLambda	lambdaMin[um]	lambdaMax[um]	nT	minT[°C]	maxT[°C]	poln
nN(Tj)						
N1(T1)		N2(T1)		...		NnN(T1)
gain(N1, T1, lambda1)		gain (N2, T1, lambda1)		...		gain (NnN, T1, lambda1)
gain(N1, T1, lambda2)		gain (N2, T1, lambda2)		...		gain (NnN, T1, lambda2)
:		:		:		:
nN(T2)						
N1(T2)		N2(T2)		...		NnN(T2)
gain (N1, T2, lambda1)		gain (N2, T2, lambda1)		...		gain (NnN, T2, lambda1)
gain (N1, T2, lambda2)		gain (N2, T2, lambda2)		...		gain (NnN, T2, lambda2)
:		:		:		:

where:

nLambda	number of wavelength points in each spectrum
lambdaMin	[um] minimum wavelength in each spectrum
lambdaMax	[um] maximum wavelength in each spectrum
nT	number of temperatures for which sets of spectra are provided. Temperatures are equally spaced; if nT=1, temperature is given by average of minT and maxT (see below).
minT	[°C] minimum temperature for which a set of spectra is provided
maxT	[°C] maximum temperature for which a set of spectra is provided
poln	integer specifying polarisation for which spectra are provided (1=TE, 2=TM).
nN(Tj)	number of carrier densities at jth temperature, where j=1...nT, the temperatures being equally spaced between minT and maxT
Ni(Tj)	[cm <sup>-3</sup> ] ith carrier density value at jth temperature, where i=1...nN(Tj)
gain(Ni,Tj,lambda <sub>k</sub> )	[1/cm] value of gain for ith carrier density and jth temperature at the kth wavelength, where k=1...nLambda, the wavelengths being equally spaced between minLambda and maxLambda

An example spectra data file is shown below:

```

21 0.78 0.82 3 20 30 1 // nLambda lambdaMin[um] lambdaMax[um] nT mint[C] maxT[C] poln(1=TE,2=TM)
//T=20 [C]
52 //NN
1.498180e+016 6.746685e+017 9.536785e+017 1.175395e+018 1.365800e+018 1.535521e+018 1.690231
-4.934911e+003 -3.676893e+003 -3.166087e+003 -2.790005e+003 -2.485583e+003 -2.228602e+003 -2.005221
-4.742227e+003 -3.497078e+003 -2.963667e+003 -2.574882e+003 -2.263800e+003 -2.003239e+003 -1.777868
-4.433592e+003 -3.318502e+003 -2.768431e+003 -2.371948e+003 -2.057940e+003 -1.796273e+003 -1.571321
-3.964732e+003 -3.137687e+003 -2.577057e+003 -2.177378e+003 -1.862855e+003 -1.602177e+003 -1.379618
-3.327663e+003 -2.951457e+003 -2.387781e+003 -1.988840e+003 -1.676112e+003 -1.418611e+003 -1.199988
-2.453258e+003 -2.755632e+003 -2.199138e+003 -1.804835e+003 -1.496835e+003 -1.244517e+003 -1.031044
-1.614924e+003 -2.542964e+003 -2.008693e+003 -1.624546e+003 -1.324660e+003 -1.079391e+003 -8.721629
-1.095852e+003 -2.300138e+003 -1.812081e+003 -1.446880e+003 -1.159189e+003 -9.230512e+002 -7.232562
-7.996314e+002 -2.013023e+003 -1.601847e+003 -1.269247e+003 -9.996531e+002 -7.755594e+002 -5.847708
-6.190547e+002 -1.687098e+003 -1.372353e+003 -1.087437e+003 -8.447089e+002 -6.372564e+002 -4.57911C
-4.992179e+002 -1.307786e+003 -1.130521e+003 -9.032912e+002 -6.946431e+002 -5.094755e+002 -3.452464
-4.140128e+002 -9.187752e+002 -8.695534e+002 -7.222652e+002 -5.560709e+002 -3.974553e+002 -2.516814
-3.503248e+002 -6.470265e+002 -6.227900e+002 -5.442204e+002 -4.321960e+002 -3.82314e+002 -2.847497
-3.009413e+002 -4.847696e+002 -4.549433e+002 -4.014846e+002 -3.301404e+002 -2.452058e+002 -1.517331
-2.615444e+002 -3.835901e+002 -3.521716e+002 -3.098696e+002 -2.598357e+002 -2.031429e+002 -1.403843
-2.293787e+002 -3.151908e+002 -2.856266e+002 -2.509197e+002 -2.132225e+002 -1.730529e+002 -1.304387
-2.026060e+002 -2.656114e+002 -2.390088e+002 -2.100910e+002 -1.803197e+002 -1.499993e+002 -1.190851
-1.799610e+002 -2.277986e+002 -2.041956e+002 -1.797821e+002 -1.555036e+002 -1.315107e+002 -1.077272
-1.605506e+002 -1.978913e+002 -1.769884e+002 -1.561164e+002 -1.358511e+002 -1.162283e+002 -9.71429C
-1.437306e+002 -1.735863e+002 -1.550284e+002 -1.369795e+002 -1.197606e+002 -1.033285e+002 -8.755685
-1.290264e+002 -1.534107e+002 -1.368753e+002 -1.211137e+002 -1.062771e+002 -9.227171e+001 -7.895943

//T=25 [C]
51 //NN
1.624351e+016 7.065016e+017 9.798680e+017 1.198638e+018 1.386830e+018 1.554630e+018 1.707566
-5.014205e+003 -3.736291e+003 -3.260482e+003 -2.905696e+003 -2.616100e+003 -2.370079e+003 -2.155346
-4.864370e+003 -3.53473e+003 -3.052947e+003 -2.682957e+003 -2.384128e+003 -2.132453e+003 -1.91394C
-4.634530e+003 -3.375219e+003 -2.856203e+003 -2.476448e+003 -2.173067e+003 -1.919092e+003 -1.699662
-4.265796e+003 -3.197305e+003 -2.665313e+003 -2.280357e+003 -1.975305e+003 -1.721090e+003 -1.502865
-3.739178e+003 -3.016582e+003 -2.477752e+003 -2.091587e+003 -1.786949e+003 -1.534559e+003 -1.319261
-3.016572e+003 -2.829821e+003 -2.291999e+003 -1.908106e+003 -1.606321e+003 -1.357797e+003 -1.146758
-2.102650e+003 -2.632353e+003 -2.106278e+003 -1.728730e+003 -1.432794e+003 -1.189976e+003 -9.842985
-1.383295e+003 -2.415682e+003 -1.917558e+003 -1.552641e+003 -1.265970e+003 -1.030688e+003 -8.314491
-9.656321e+002 -2.165026e+003 -1.720584e+003 -1.378239e+003 -1.105268e+003 -8.797816e+002 -6.882605
-7.226183e+002 -1.871821e+003 -1.507136e+003 -1.202069e+003 -9.495682e+002 -7.372837e+002 -5.55322C
-5.694122e+002 -1.541128e+003 -1.276351e+003 -1.020554e+003 -7.973380e+002 -6.035224e+002 -4.34056C
-4.647578e+002 -1.156691e+003 -1.033020e+003 -8.390460e+002 -6.510335e+002 -4.806541e+002 -3.274955
-3.887568e+002 -8.046341e+002 -7.720831e+002 -6.586206e+002 -5.172218e+002 -3.752088e+002 -2.413861
-3.310704e+002 -5.777698e+002 -5.513199e+002 -4.879275e+002 -3.977086e+002 -2.928402e+002 -1.836552
-2.858141e+002 -4.415574e+002 -4.109905e+002 -3.641799e+002 -3.047408e+002 -2.349256e+002 -1.569531
-2.493685e+002 -3.543941e+002 -3.239882e+002 -2.861195e+002 -2.430079e+002 -1.953864e+002 -1.43494C
-2.193797e+002 -2.940086e+002 -2.660647e+002 -2.347176e+002 -2.014869e+002 -1.667729e+002 -1.305685
-1.942561e+002 -2.494249e+002 -2.244914e+002 -1.982009e+002 -1.715502e+002 -1.447642e+002 -1.177895
-1.728907e+002 -2.149690e+002 -1.928969e+002 -1.705709e+002 -1.485987e+002 -1.270734e+002 -1.059105
-1.544953e+002 -1.874472e+002 -1.678981e+002 -1.487117e+002 -1.302204e+002 -1.124225e+002 -9.520771

```

Spectra data for a particular material are imported from a file into PICWAVE through a material database file, as follows:

- Open the material database containing the material for which gain spectra data are to be imported
- In the material definition type one of the following lines:

for TE poln spectra data:

```

IMPORT_GAIN_SPECTRA [relative path and filename of spectra file] Or
IMPORT_GAIN_SPECTRA_TE [relative path and filename of spectra file]

```

for TM poln spectra data:

```

IMPORT_GAIN_SPECTRA_TM [relative path and filename of spectra file]

```

These flags specify the spectra file whose data will be used to fit a multiple Lorentzian gain model for the material. They also tell the material to use a multiple Lorentzian gain model in any TDIW simulation where it is used as an active layer material, instead of any other gain model that might be contained in the material definition in the material database file. In a TDIW simulation, modes with a TE fraction > 50% will look to use a TE gain model; modes with a TM fraction < 50% will look to use a TM gain model.

For example:

```

BEGIN InGaAsP(x) // material name
ANAL_INGAASP1 20 0 // loads internal InGaAsP model at 20 Celcius
GAIN_POLYN 1.5e18 7.0e-6 1 // No [1/cm3] gmat [cm3/s] polylen g_n2 g_n3... ;gain model
GAIN_LAMFUNC 10000 1.55 0.000 .000 0 0 // g2[um-2] lam0[um] lamN1 lamN2 lamT1 lamT2
GAIN_EPS 1e-15 // gaineps (non-linear gain) [cm3]
RADRECOMB 1e20 1e-10 0.0 // tsr0[ps] Btsr[cm3/s] Ctsr[cm6/s] ; rad recombination
NRADRECOMB 1e4 0.0 1e-28 // tsnr0[ps] Btsnr[cm3/s] Ctsnr[cm6/s] ; non-rad recombination
DIFFUSION 20.0 0 // ddif0[cm2/s] pdif_n[cm5/s] ; diffusion coefficients
LEF 2.5 // lef ; linewidth enhancement factor
RHO 600 // rho ; electrical resistivity
DOSCB 2.9e17 // [cm-3] dos in conduction band
DOSVB 8.4e17 // [cm-3] dos in valence band
IMPORT_GAIN_SPECTRA_TE ingaasp-gain-TE.txt //import gain spectra data
END

```

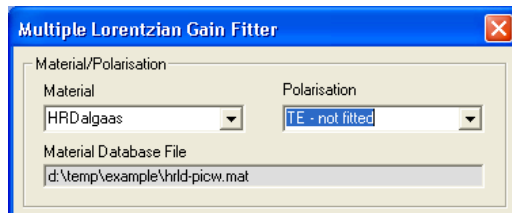
Here the line `IMPORT_GAIN_SPECTRA_TE ingaasp-gain-TE.txt` tells the material to look for a spectra data file called `ingaasp-gain-TE.txt` in the same directory as the material database file. The `IMPORT_GAIN_SPECTRA_TE` flag will also cause existing gain model definitions for this material (lines `GAIN_POLYN` and `GAIN_LAMFUNC` in this example) to be ignored.

### 8.3.2 Fitting a gain model from imported data

Once a given material has been set to import gain spectra data from a file and to use a multiple Lorentzian gain model, as detailed above, the gain model is fitted as follows:

#### 1. Selecting a material/polarisation

- Open the device window of the circuit where the material is used (i.e. containing an active waveguide or taper section using an RWG which uses the material in an active layer)
- Select `/Tools/Multiple Lorentzian Gain Fitter`
- Select the material from the list and the polarisation for which the gain model will be fitted

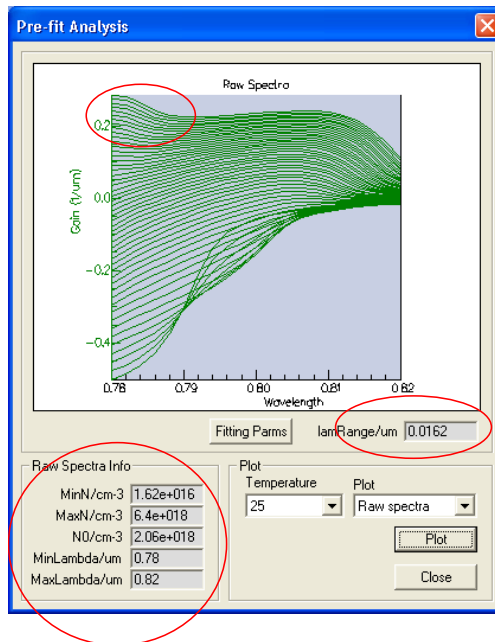


#### 2. Pre-fit Analysis

The next stage is to inspect the imported spectra data and set up the *gain fitting parameters* accordingly. It is assumed that the **tStep** and **lambdaCentre** have been defined in the TDTW Calculator, with **lambdaCentre** being ideally equal to the average gain peak wavelength in the set of imported spectra; they can be adjusted later if need be.

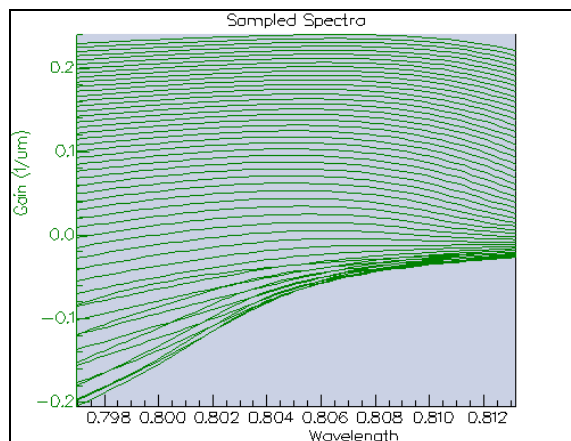
- Click on **Pre-Fit Analysis** and select a temperature and click **Plot**

At this stage, one should take careful note of the following: second gain peaks at higher carrier densities (as in the figure below); the wavelength range of the spectra relative to the **lamRange**; and the Raw Spectra Info parameters (see §8.2.2). These are illustrated in the example below.



➤ Then select *Sampled Spectra* from the **Plot** drop-down list and click **Plot**

*Sampled Spectra* plots the portion of the imported spectra lying within the free spectral range. The spectra from the previous example look as follows in an appropriately chosen free spectral range (see 8.3.3):



Note that the second gain peaks lie outside the free spectral range window and so will no longer present a problem for the fitting process (see 8.3.3).

### 3. Fitting Parameters

Having looked at the *Pre-Fit Analysis*, an informed decision about the *gain fitting parameters* can be made.

- In the *Pre-Fit Analysis* window or in the main interface, click on **Fitting Params** to open the *Gain Fitting Parameters Panel*.
- First decide a value for **EndN**, the carrier density up to which the gain model needs to operate.
- Decide if and where upward extrapolation is needed e.g. if you want to extrapolate beyond **maxN** (i.e. **EndN** > **MaxN**), or if you want to extrapolate into a carrier density region where a second gain peak emerges. To enable upward extrapolation set

**ExtrapolateUpwards** to *true* and enter a carrier density ( $\leq$ **MaxN**) at which to begin extrapolation.

- Decide whether downward extrapolation needs to start from above the minimum imported carrier density (**MinN**). If so adjust the value of **ExtrapDownStartN** accordingly.

It is difficult to determine a priori if and how the Fit Improvement settings need adjusting. They are best left at their default values until fitting has first been attempted, whereupon they can be adjusted and the spectra re-fitted if need be.

- When finished close the *Gain Fitting Parameters Panel* and return to the main window of the Multiple Lorentzian Gain Fitter

#### 4. Run Fit

With the fitting parameters set one can proceed to fit a gain model.

- Click **Run Fit** to start the fitting process. The progress bar and fitting log will indicate fitting progress.

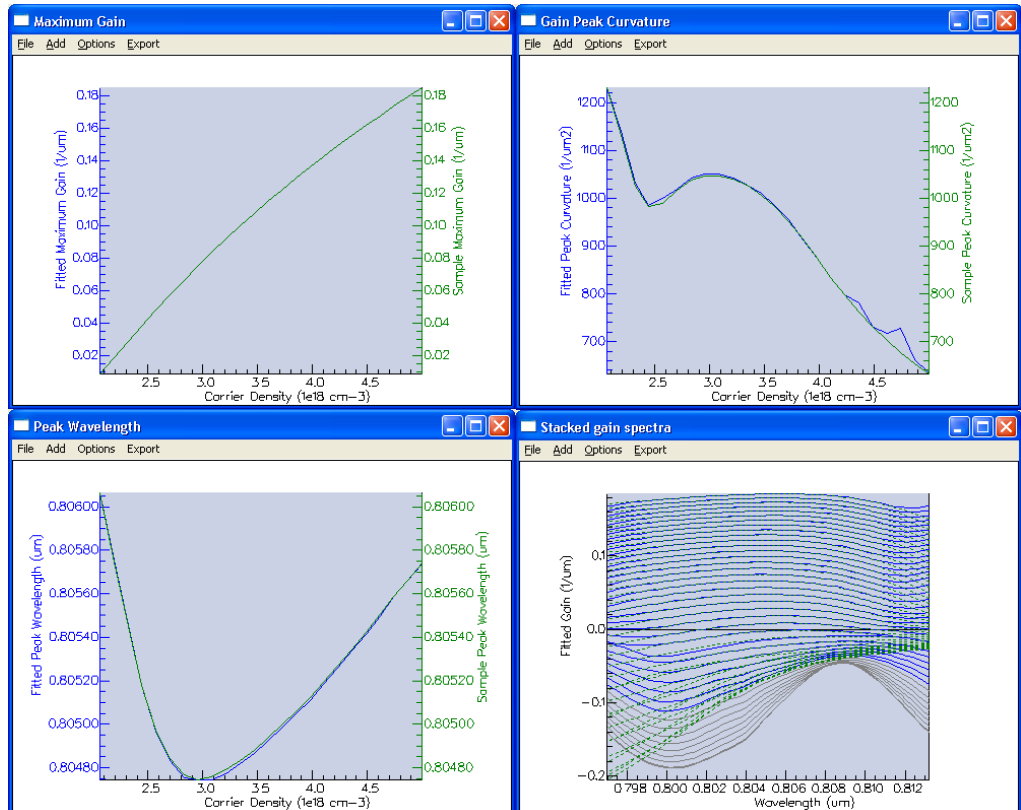
If an error occurs during fitting, the fitting process will terminate and a message will be displayed in the fitting log detailing the reason for the failure. If this happens make the appropriate changes to the *gain fitting parameters* and/or imported spectra data and restart the fitting process by clicking **Run Fit**.

#### 5. Inspect Fitting Results

When the fitting is successfully completed a fitting summary message will appear in the fitting log and the drop down menus in the Fitting Results section at the bottom of the window will become enabled.

- Select *Stacked Gain Spectra* from the **Function** drop-down list. Select a temperature from the **Temperature** drop-down list, then click **Plot**.

The stacked gain spectra plot is the best means of determining the overall success of the fitting. The other items in the **Function** drop-down list enable one to assess the accuracy of the fitting around the gain peak region. The fitting results for the previous example are as follows:



These results indicate a successful accurate fit. Note the grey extrapolated spectra in the stacked spectra plot (bottom right), where extrapolation was used for a number of imported carrier densities down to zero instead of trying to fit the imported spectra, which from previous plots can be seen to be anomalous.

If the fit is not satisfactory,

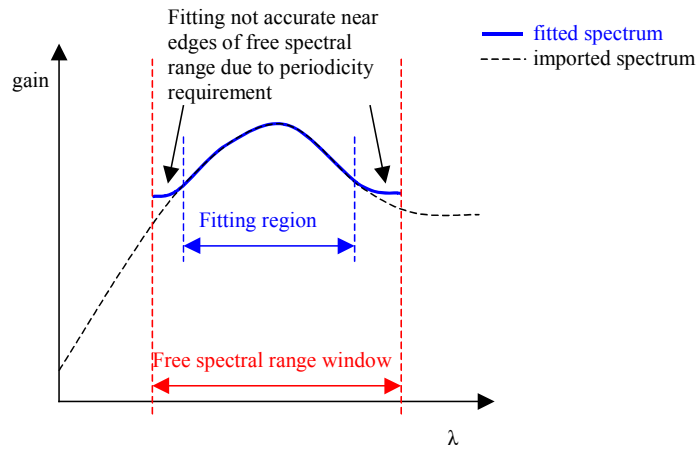
- adjust the *gain fitting parameters* accordingly and re-fit otherwise,
- click **Accept Fitting Results** to store the gain model

#### 6. Using a fitted gain model

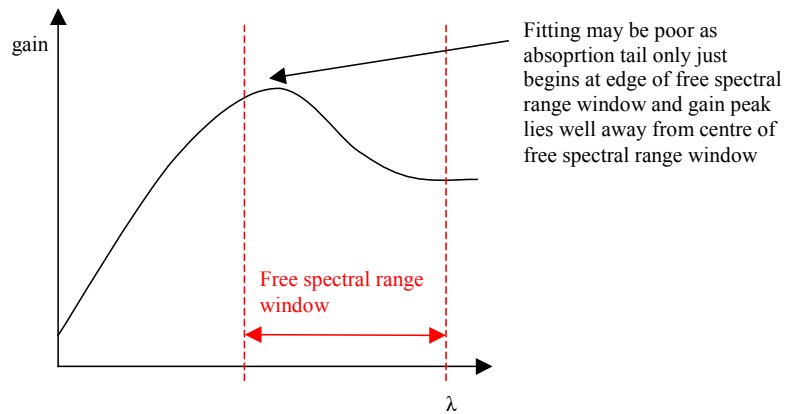
Once a gain model has been stored it can be used by the device for TDTW simulations. The model will be saved permanently when you save the project file. The fitting results of stored gain models can be viewed from the *Multiple Lorentzian Gain Fitter* main interface (see §8.2.1).

#### 8.3.3 Notes: Choice of fitting region

The *Multiple Lorentzian Gain Fitter* fits those portions of the imported spectra which lie within the bounds of the free spectral range. It fits the imported spectra with a number of pseudo-Lorentzians which are periodic over the free spectral range. This periodicity is required by the TDTW algorithm employed by PICWAVE and means that accurate fitting can only be achieved in a central portion of the free spectral range. The size of this central portion (fitting region) is defined by the user (**FitFrac** in *gain fitting parameters*, §8.2.3) and is typically  $\leq 70\%$ ; spectra data lying outside it, at the edges of the free spectral range, are ignored during the fitting process. This is illustrated in the figure below.



Within the fitting region the fitting algorithm generally expects to find an absorption tail, and at carrier densities above transparency, a single peak (preferably near the centre of the free spectral range window) and a further tail at the higher wavelength end (as in the figure above). The algorithm may give poor fitting results or even give an error if the free spectral range (and therefore the fitting region) window is poorly chosen, for example:



The periodicity requirement also means that a pseudo-Lorentzian allotted to fit one part of the spectrum will have an effect on the fitting of another part of the spectrum. In particular this means that there is a trade off between the fitting accuracy at the centre and the fitting accuracy towards the edges of the fitting region.


# Chapter

# 9

## The TDTW Calculator

The *TDTW* simulation engine is controlled by the *TDTW Calculator* window. From here you can set up the parameters controlling the engine; start, stop and pause simulations; and access simulation results on completion.

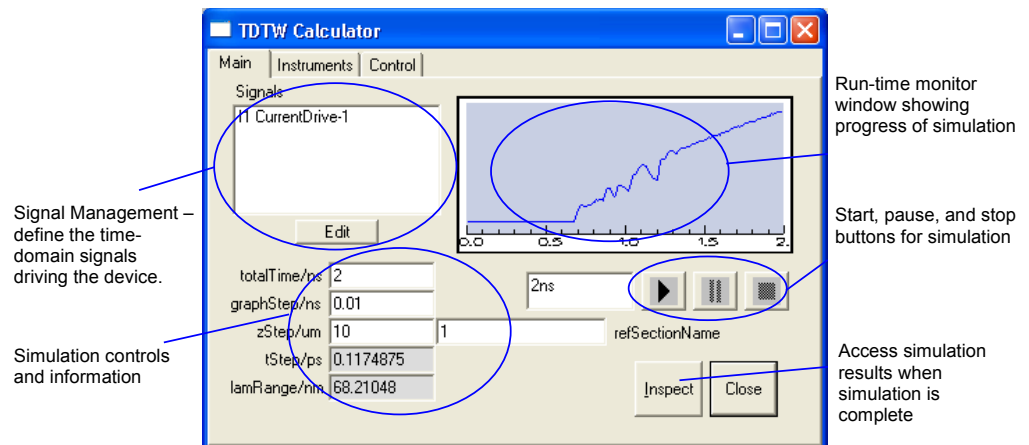
To open the *TDTW Calculator* from the device window,

- Click on the  icon in the tool-bar *or* select //Tools/TDTW Calculator from the menu bar.

There are three panels in which the simulation is configured: the Main, Instrument and Control panels (§9.1, 9.2 and 9.3 respectively). The results are viewed using the *Time Domain Results* window (§9.4), which is divided, according to the types of results, into the Instruments, Spectra, Oscilloscope and LI Fit panels (§9.5, 9.6, 9.7 and 9.8 respectively).

### 9.1 Main Panel

The Main panel of the *TDTW Calculator* window is shown below, following which is a description of its various parts.



#### 9.1.1 The Run-Time Monitor Window

The *Run-time Monitor Window* allows you to plot up to two instrument measurements as the calculation progresses in real-time. You need to choose which instruments' measurements you want to plot here – each instrument has a **plotInMonitor** switch (see §4.15). Enable this for each instrument you want to monitor in run-time.

#### 9.1.2 Simulation Control Parameters

**totalTime** [ns] the time length of the simulation



<b>graphStep</b>	[ns] the instruments will record a measurement at each graphStep interval.
<b>zStep</b>	[um] this is an important simulation parameter which determines the spatial resolution of the simulation along the propagation direction. It also determines the simulation time-step ( <b>tStep</b> ) and therefore the free spectral range - a smaller <b>zStep</b> will calculate the device response over wider wavelength or optical bandwidth. The time it takes for a simulation to run is proportional to ( <b>zStep</b> ) <sup>-2</sup> .  Note that <b>zStep</b> can be defined for only one section, the <i>reference section</i> , whose name must be specified in <b>refSectionName</b> . The zSteps of the other sections are then determined indirectly from <b>zStep</b> as detailed in §11.2. See §4.9.2 for details on using taper sections as <i>reference sections</i> .
<b>refSectionName</b>	The name of the <i>reference section</i> whose zStep is defined by the <b>zStep</b> parameter. The number of the section may also be used, sections being numbered according to the order in which they were added to the device (see <b>zStep</b> above).
<b>tStep</b>	[ps] (read-only) the engine's calculation time step; this is determined by <b>zStep</b> and group velocity of the <i>reference section</i> : <b>tStep</b> = <b>zStep</b> /group velocity (see §11.2 for more details)
<b>lamRange</b>	[nm] (read-only) the free spectral range or optical bandwidth of the calculation expressed as wavelength range; it is determined by <b>tStep</b> and the central simulation wavelength, <b>lambdaCentre</b> (see §9.3.2): <b>lamRange</b> = <b>lambdaCentre</b> <sup>2</sup> /(c. <b>tStep</b> ), where c is the speed of light in a vacuum


### 9.1.3 Signals

The panel shows a list of all the signals driving the device, optical and electrical. Double-click on a signal entry to edit or view a signal in the *Signal Editor*. See §4.16 for details of the *Signal Editor*.



### 9.1.4 Running a Simulation and Inspecting Results

Once the simulation parameters have been configured in the Main, Instrument and Control panels, the simulation is ready to be run from the Main panel.



To start a simulation,

- Click the  button. You should see the simulation time incrementing as the calculation proceeds. You should also be able to see the run-time plots of any instrument measurements you have chosen to plot in the *Run-time Monitor Window*.


To inspect the results after, or at any time during, a simulation,

- Click the  button. This will open the *Time Domain Results* window (see §9.4), where you can examine the progress/final results of the calculation in detail. If the simulation is running it will be paused, and can be resumed again once you have closed the *Time Domain Results* window, by clicking .

To stop (abort) a simulation that is running,

- Click the Run/Kill button  button in the top-right of the Main Window, or the  button on the *TDTW Calculator* panel.

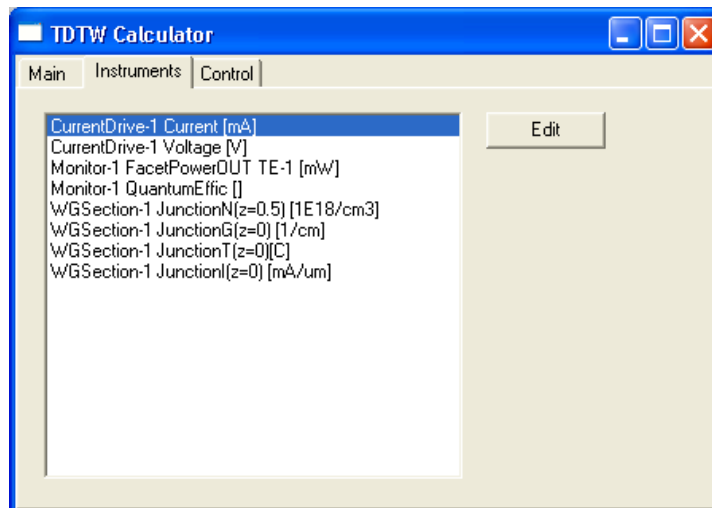
To pause a simulation,

- Click the  button, and resume by clicking it again.

## 9.2 Instruments Panel

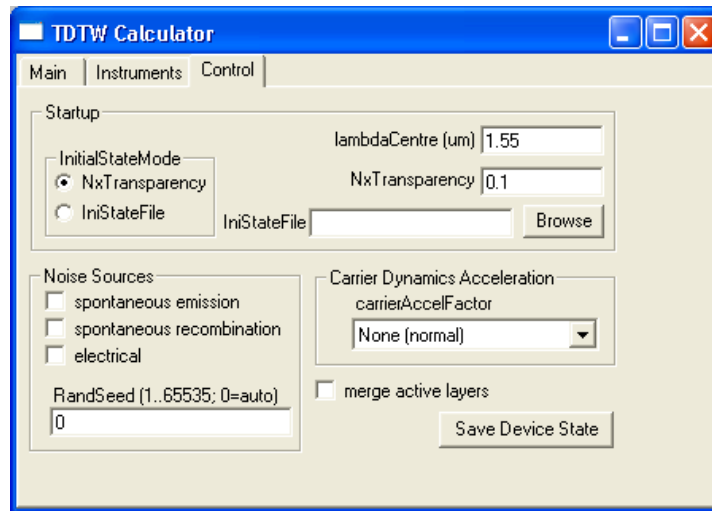
The Instruments panel, shown in the figure below, displays all the instruments that have been defined (when building the circuit in the device window) and allows you to review and modify your instruments - by selecting the instrument in the list and clicking the **Edit** button. Instruments cannot be added or removed from the Instruments panel; that can only be done from the device window.

Each instrument will perform its measurement at the interval specified by **graphStep** (see above).



## 9.3 Control Panel

The Control panel, shown in the figure below, is used to define various control parameters for the TDTW engine and also to set the central wavelength of the simulation.



The **merge active layers** setting, when enabled, causes the *active layers* of any waveguide or taper section whose active layers are *all* identical to be computationally merged into one active layer, thus reducing the time it takes for the simulation to run. Identical active layers are those with identical physical parameters (defined in the RWG) and identical resistive connections to the contact (defined in the **CurrentFlowModel** in the waveguide or taper section properties panel). As a result, active layers thus merged are no longer unique and will in effect be forced to have the same carrier densities in their *CoreCells* (see §11.2).

The rest of the Control panel is discussed in the following sections.

### 9.3.1 Startup – Device State Control

When a simulation is started, the device needs to be initialised into some state. There are two options:

1. Set carrier densities to some fraction of their transparency density. Optical intensities are set to zero. To use this option, set **InitialStateMode** to *NxTransparency* and set a carrier density value (as a fraction of the transparency density) in the **NxTransparency** box. A value of 1.5 or 2 is typically good, or you may want to start at zero.
2. Set the state to a previously saved *device state*. Enter a filename in the **IniStateFile** box and set **InitialStateMode** to *IniStateFile*. The state file will be read when the simulation starts.
  - To create a *device state* file, click on the **Save Device State** button once your simulation has completed. You can use this state file on any future simulation of a similar device (i.e. one which has the same number of sections, and also the same **zStep** value).

### 9.3.2 Startup – Control Parameters

**lambdaCentre** this is the central wavelength for your simulation. You should set this as close as possible to the centre of the range of wavelengths that will be generated by the laser. The free spectral range of the simulation is also determined by **zStep** – see §9.1 above.

**NxTransparency** (see **InitialStateMode** above) This defines a carrier density in each active region as a fraction of the transparency density of the material. Thus **NxTransparency** = 1 sets the carrier density of whole device to the transparency density.

### 9.3.3 Noise Sources

The Time Domain Module includes stochastic *noise sources* in the simulation for the following processes:

- Spontaneous emission of photons (see §17.4)
- Spontaneous recombination of carriers
- Electron injection into the active layers

Sometimes it is preferable to disable these *noise sources* to get the underlying performance of the device without long averaging. Note however that you should switch these off only with caution. For example, without spontaneous emission noise, a device with no optical sources can not start lasing. You can turn off the spontaneous emission noise if you are modelling a travelling wave amplifier or if you are starting from a *device state* (see above) that is already lasing and you are not interested in what is happening to any non-lasing modes.

**RandSeed** the noise sources are driven by a random number generator in the program. If **RandSeed** is zero, then you will get a different set of random numbers each time you run the program. If you set it to a non-zero value then you will always get exactly the same noise pattern each time you run the program with the same parameters.

#### Spontaneous Emission Noise Spectrum

By default, spontaneous emission *noise* is *white noise* (flat spectrum). *Coloured noise* is also supported, whereby active materials can be given a Lorentzian noise spectrum. This is done through the material database file in which the *noise* Lorentzian peak wavelength and half-width can be defined as functions of carrier density and temperature for a given material (see §10.1.1).

### 9.3.4 Carrier Dynamics Acceleration

Changing the **carrierAccelFactor** allows you to artificially speed up the carrier convergence to the steady state without altering the actual steady state. For example, with a given **carrierAccelFactor** *AF*, a rate equation such as the following,

$$\frac{dN}{dt} = f(N, S, I)$$

would become

$$\frac{dN}{dt} = AF \cdot f(N, S, I),$$

where *N* and *S* are the carrier and photon densities, *I* the ‘useful’ current through the active layer. The rate, *dN/dt*, is increased by a factor of *AF*, but the steady state (*dN/dt=0*) solution is unchanged.

This feature is useful if you want to find the steady state of a system quickly (i.e. without having to run the simulation for very long), particularly when you want to study how the steady state varies with a certain parameter (e.g. current).

*Hint:* Using a large **carrierAccelFactor** will also artificially increase the effects of noise. In such cases, it is advisable to switch off noise sources if you want clear measurements of quantities in the steady state. This requires, however, that you initialise the device with a previously saved *device state* (see above) that is already lasing

## 9.4 The Time Domain Results Window

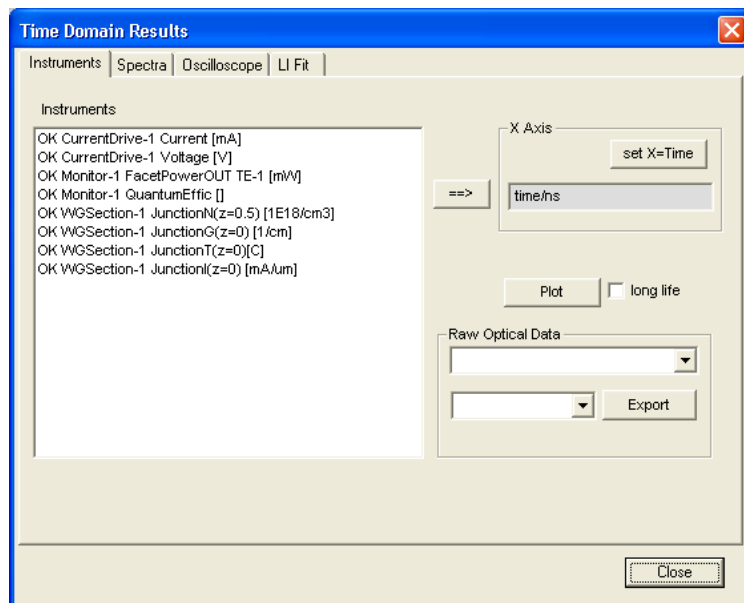
The *Time Domain Results* window allows you to view all the results that have been collected during the time-domain simulation. It is accessed by clicking the **Inspect** button in the Main panel of the TDTW Calculator window, either during or after the simulation is completed. It allows you to:

- plot the data recorded by the *instruments* (*Instrument Results* panel – see §9.5)
- plot optical and intensity (RIN) spectra (*Spectra Results* panel – see §9.6)
- view pulse trains and calculate pulse statistics (*Oscilloscope* panel – see §9.7)
- produce LI curves, fit parameters e.g. differential quantum efficiency, threshold current (*LI Fit* panel – see §9.9)

The *Instrument Results* and *Spectra Results* panels can also be accessed, when a simulation is paused or finished, by right-clicking on any monitors on your circuit in the device window and selecting */Instrument results/[instrument name]* or */Spectra results*, respectively.

## 9.5 Instrument Results

The *Instrument Results* window, shown below, will allow you to display the various data collected by the *instruments* you have set up in your simulation.



### 9.5.1 Plotting Instrument Results

On the left you will see a list of your instruments, to plot the results of one of these,

- Double click on one of the entries *or* select the line and press the **Plot** button.

If there were any errors encountered in an instrument's reading process then its OK prefix will have disappeared and you will not be able to plot its data.

By default each plot you generate will disappear when you close the *Time Domain Results* panel or plot another result. If you tick the **longLife** checkbox, the plot will remain until you explicitly close it or exit the program.

### 9.5.2 X-Axis

By default, time is plotted on the x-axis. To change the x-axis variable to one of your instrument quantities,

- Select the relevant instrument, then press the **==>** button to move it into the X-Axis box.

All instrument quantities you then plot will be plotted against this. This feature allows you to plot such things as LI (light-current) plots, for example. The x-axis variable can be reset to time with the set **X=Time** button.

### 9.5.3 Exporting Raw Optical Data

You can export the complex fields of any *monitor* using these controls. Choose the monitor from the upper drop-down list (it will only show if it has **enableSpecMeas** enabled – see §4.14), then choose from the lower drop-down list whether you want to export real/imag data or absolute/phase data then click **Export**. An ASCII file, with the specified filename, will be generated with your raw monitor data.

## 9.6 Spectra Results

PICWAVE includes extensive algorithms for obtaining various spectra from the time-domain results. The *Spectra Results* panel, shown below, can generate a wide variety of

The screenshot shows the 'Time Domain Results' dialog box with several annotations pointing to specific controls:

- Choose spectrum type:** Points to the 'Monitor' dropdown menu (set to 'Monitor-Out: TE-1') and the 'Type' dropdown menu (set to 'Optical').
- Choose which monitor result you want to study:** Points to the 'Reference' dropdown menu (set to '(none)').
- Configure the FFT:** Points to the 'FFT Size' dropdown menu (set to 'Full'), the 'interval (ns)' field (0.9997), the 'windowW (ns)' field (1.999), the 'resolution' field (500.1 MHz), the 'range' field (4257 GHz), the 'xmin' field (0.694008 nm), and the 'xmax' field (34.12 nm).
- Configure the plot:** Points to the 'X Axis' section, where the 'frequency' radio button is selected, and the 'xmin', 'xmax', 'centre', and 'filterWidth' fields are visible.
- Choose single or time-evolving spectrum:** Points to the 'Time Evolving' section, where the 'No' radio button is selected, and the 'Plot' button is visible.

Other visible controls include the 'Magnitude Axis' section (Intensity (lin), dB (logarithmic), phase (optical)), the 'dB Range' field (180), the 'Time Window' section (tStart (ns) 0, tEnd (ns) 1.9999, tRef (ns) 1.0002, tAlign Start), the 'SMSR at tRef (dB)' field (0), the 'long life' checkbox, and the 'Mode Tracker' button.

different spectra. It can do a spectral analysis on the optical field or on the optical power (the latter being useful for RIN analysis). It can generate time-evolving or time-averaged spectra. In addition it can track laser modes as they move around the spectrum. A number of configuration parameters are common to all the spectra types and have the same function and significance unless otherwise stated in the sections below.

### 9.6.1 Optical spectra

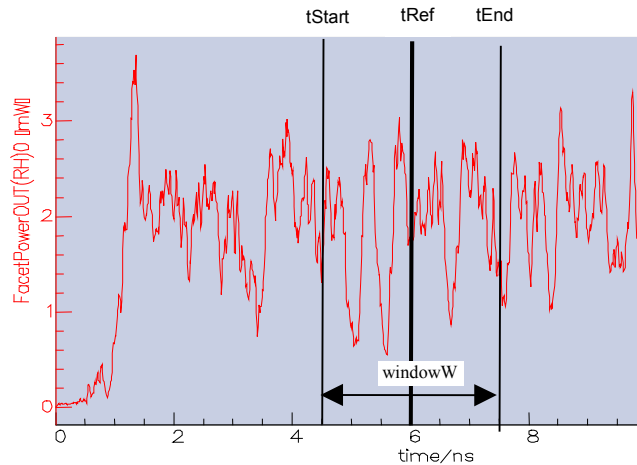
An *optical spectrum* is a spectrum of the complex *optical field*. For example it will indicate how many resonant modes a laser has, and even an estimate of the mode linewidth. There are many sorts of spectra that you might want to generate so you should understand how this module works. Firstly you should note that you can either obtain a spectrum at a given time, a *time-averaged spectrum*, or you can generate a *time-resolved spectrum*.

To create an *optical spectrum*,

- Set **Type** (on the top-right) to *Optical Spectrum*.
- From the **Monitor** drop-down list, select the monitor whose data you want to plot (it must have **enableSpecMeas** set to *True*).
- (Optional) From the **Reference** drop-down list, choose a reference monitor - this will plot a spectrum of the main monitor divided by the spectrum of the reference monitor.
- You can plot either a *time-resolved spectrum* or *time-averaged spectrum* by setting the **Time Evolving** switch to *Yes* or *No*, respectively.
- Click the  button to generate the spectrum. If you tick the **longLife** checkbox, the plot will remain until you explicitly close it or exit the program.

#### Time Averaged Spectra

With **Time Evolving** set to *No*, the  button will generate a single *time-averaged spectrum* from given *time window*. The width of the *time window* is defined by the **FFT Size** (top left of panel) – this box also shows you the width of the window, **windowW**, in ns (labelled) corresponding to the chosen **FFT Size**. By default, PICWAVE will create a spectrum for the whole time sequence. To change the length of the time window, change the **FFT Size**. To move the time window forward or back in time, set the **Time Window** parameters to the bottom right of the panel - these are illustrated below. Note that the resolution of your spectrum changes with **FFT Size**. The *free spectral range* is determined by the time step that the simulation ran with (see §2.1.5).

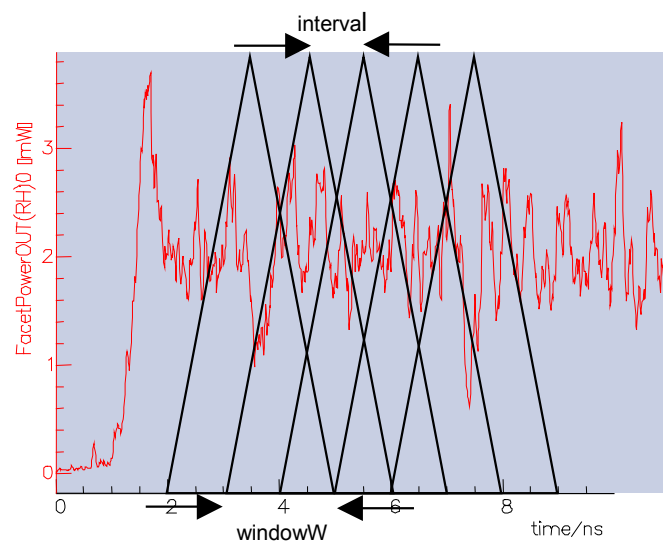


You can further customise the *time-averaged spectrum* by setting the x-axis and y-axis ranges – for example if your optical spectrum is over too wide a range then you can set the **xmin** and **xmax** values (in the **X Axis** box) to limit the range (remember that you can also alter axes after plotting by right-clicking on the plot window and selecting **/Options/Axes** etc). For the y-axis you have a choice of plotting a linear intensity, intensity in dB, or the optical phase by selecting the appropriate choice in the **Magnitude Axis** box. You can also (in the **X Axis** box) choose to plot *wavelength* or *frequency* on x-axis.

Finally, you may want to apply a *filter* to your spectrum, to reduce the amount of noise. In the **X Axis** box you will see the **filterWidth** property – increase this value to get more smoothing. The value defines the number of points (= **filterWidth** – 1) either side of a given point in the spectrum which are used in the smoothing process. Note that phase spectra are not filtered.

#### Time Resolved Optical Spectra

With **Time Evolving** set to *Yes*, the program can just as easily generate a *time-resolved spectrum*. As for the time-averaged spectrum, choose an **FFT Size** to define your sampling window width. This also defines the time intervals used to generate the *time-resolved spectrum*, using a moving triangular (Parzen) window – see figure below. It defaults to 256 steps.



If you wish, you can analyse only part of the simulation data, e.g. removing the startup transitions, by setting the **Time Window** parameters. You should note the trade off



between the optical resolution of your *time-resolved spectrum* and the time resolution **interval** (as determined by **FFT Size**).

*Hint:* Time resolved spectra produce a lot of data. SciGraph (the plotting engine) gives you many options for plotting in different ways; we recommend plotting this spectrum either as a Colour Map (2D View) or as a Slice Plot – experiment with the graph settings (right-click /**Display Options**) and set the default when you find one you like.

### 9.6.2 Intensity and RIN Spectra

The *Intensity/RIN spectrum* option allows you to plot two types of spectra: that of the power, which ignores the phase of the optical field and can be plotted as received electrical power (elec power) or optical power (AM); and that of the frequency (FM), which is the Fourier transform of the frequency chirp as calculated from the instantaneous time derivative of the optical phase. In the **type** drop-down list they are called *INT/RIN – elec power*, *INT/RIN – AM* and *INT/RIN – FM*, respectively. All three of these can be normalised by the modulation frequency spectrum of the any of the electric drive signals in the circuit.

There are two main uses for such spectra:

- a) to calculate the modulation response of the power (AM) or frequency (FM)
- b) to calculate the relative intensity noise (RIN) spectrum

RIN spectra and the modulation response of the power are essentially calculated in the same manner, except that the input signal for each should be different. Each use is described below.

#### Calculating a modulation response

To calculate the *modulation response*, you first need to run a simulation so that the laser is running stably at your desired bias current, with a reasonably strong impulse in the drive current (or drive voltage) near the beginning of the simulation. You should see a nice spike in the resulting optical output with some ringing afterwards. You will need to run the simulation for some time after the impulse to get a reasonable modulation frequency resolution.

Now choose an **FFT Size** with the desired resolution (displayed on the panel). Notice the window width (**windowW**) that is chosen and remember how this is centred around the value of **tRef**– you will want to make sure your window includes the impulse by adjusting the **Time Window** parameters if need be. (Note that tRef is always shifted automatically so that the window stays within the available data range.)

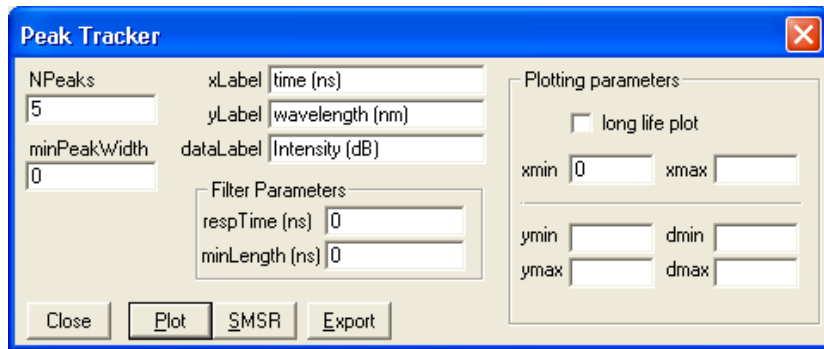
See §15.2 for further detail and instructions.

#### Calculating a RIN spectrum

See §15.3 for instructions on calculate a *RIN spectrum*

### 9.6.3 The Mode Tracker

Often it is useful to analyse the laser in terms of its cavity modes. The *Mode Tracker*, shown below, will attempt to follow the modes in time and plot both the wavelength and the intensity of all significant modes.



In a typical simulation modes may be appearing and disappearing. The *Mode Tracker* has techniques for dealing with this. On the one hand it wants to start following new modes that appear and stop trying to follow modes that disappear. On the other hand, a mode whose amplitude kept going above and below some threshold would cause lots of tracks to begin and end.

Using the Mode Tracker

- Before opening the *Mode Tracker*, first ensure that the settings of the *Spectra Results* panel give you a nice time-resolved optical spectrum clearly showing the modes of interest.
- Then click **Mode Tracker**
- Input the desired *Mode Tracker* parameters (see below) and then click **Plot**, **SMSR** or **Export** (see below)

Parameters

The *Mode Tracker* results are heavily dependent on the *FFTSize*, *filterWidth* and *TimeWindow* parameters in the *Spectra Results* panel; it also has its own parameters, which are as follows:

- NPeaks**                    the number of peaks (modes) that you would like the algorithm to track. Note that this is used only as a guideline – the *Mode Tracker* may track a few more or a few less as a result of the operation of the filters.
- minPeakWidth**            [advanced use] This controls the peak locating mechanism and defines the minimum peak width (in nm) that it will detect. Leave at zero for automatic.
- respTime**                    [advanced use] The filter prevents the algorithm trying to track a brief spike that momentarily appears in the spectrum. The magnitude of a peak first passes through a low-pass filter, and then the peak track is displayed only if it is longer than **minLength**.
- minLength**                    [advanced use] see **respTime**.

Plot

The Plot button will show the Peak Data plots window. The upper canvas traces out the different peaks that Mode tracker has found, plotting their wavelength against time. The lower canvas shows how the intensities of these same peaks evolve with time.

## SMSR

The **SMSR** button will plot the ratio of the strongest to the next-strongest peak, the *side mode suppression ratio (SMSR)*, as a function of time. This plot does not depend on the Mode Tracker parameters.

The SMSR feature in the Mode Tracker is convenient and allows for more flexibility after the simulation is completed (by adjustment of the Spectra Results parameters etc) than does the SMSR instrument but suffers from the fact that, as with time-resolved spectra, there is a trade-off between the resolution of the spectra used to calculate the SMSR and the time interval at which they are calculated. (See §9.6.4 for SMSR details)

## Export

You can export your data to an ascii file using this button. Before doing so you need to generate a track set using **Plot**. Then press **SMSR** and enter a file and a “format string”. This format string is in C programming language sprintf format. The number before the period defines the total width printed per number and the number after defines the number of significant figures.

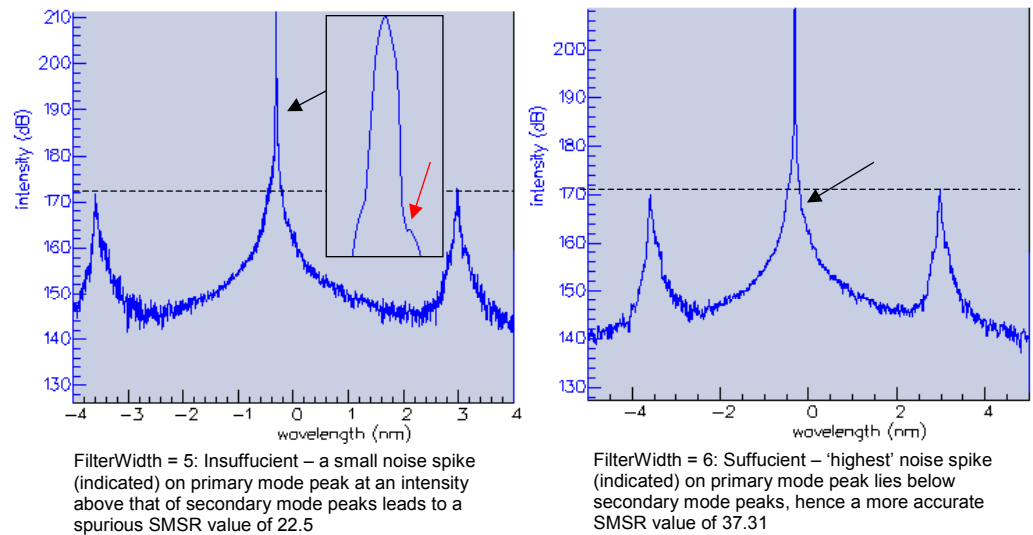
### 9.6.4 SMSR

The *side mode suppression ratio (SMSR)* can be evaluated in several ways: the *SMSR Instrument*, the SMSR feature in the *Mode Tracker* and the **SMSR at tref (dB)** box in the *Spectra Results* window.

In the *Spectra Results* window, the *SMSR* is evaluated by plotting a time-averaged spectrum and observing the *SMSR* calculated automatically in the **SMSR at tref (dB)** box.. This method is the most convenient if you not interested in plotting the time evolution of the *SMSR*, and has the added advantage in that you can visually compare the calculated value with the spectrum itself (the difference between the strongest and next strongest peaks, in dB plotting mode). The extent to which they match should allow you to determine whether your choice **filterWidth** and/or **FFT Size** were sufficient to remove the effects of noise (see below).

#### *General SMSR hints:*

- An important step in calculating the *SMSR* (for all methods) is to first remove the effects of noise (when present) by the use of a **filterWidth** number, which smooths the spectra from which the *SMSR* is calculated by convolving the spectrum with a triangular filter function of total width **filterWidth** points. Noise produces spikes on the spectrum; one such spike on the strongest *mode* peak in the spectrum may exceed the intensity of the next strongest *mode* peak and so lead to an incorrect *SMSR* value, as illustrated below. The **filterWidth** should be increased until the *SMSR* is more or less constant for all higher values. Note that increasing the **filterWidth** will cause the mode peaks to become increasingly rounded and gradually reduced in their height.



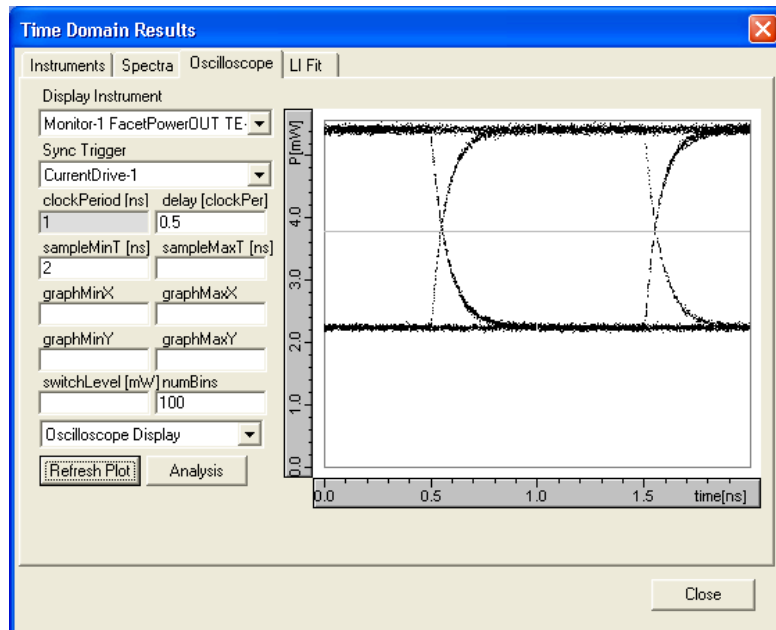
- If you wish to plot the time evolution of the *SMSR* it is worth noting that the *SMSR Instrument*, unlike the *Mode Tracker* SMSR, has a spectrum resolution (**fftSize**) and time interval (determined by **skipStep**) which are independent. Thus, for a given **fftSize**, its *SMSR* plots can have a much greater time resolution (up to the simulation time step) than the *Mode Tracker*, and are always plotted for whole time sequence.

## 9.7 The Oscilloscope

PICWAVE includes a comprehensive *Oscilloscope* (see figure below) for viewing pulse trains and calculating pulse statistics. The *Oscilloscope* is designed to display and analyse both pulses and NRZ eye diagrams.

It will calculate jitter, pulse widths, pulse energy, peak pulse power and other characteristics, and includes extensive clock sync options. Please see §15.4 for specific details on how to set up and generate an *eye diagram*.

As well as displaying an oscilloscope view of your periodic signal, the panel can also plot histograms of the measured pulse statistics.



### 9.7.1 Syncing

The *Oscilloscope* assumes that the signal is periodic or quasi-periodic and can synchronise with the signal period in one of three different ways:

- a) regenerate a clock from the optical output. This is particularly convenient for mode locking.
- b) enter a manual clock period.
- c) discover the signal period by inspection of the electrical drive. This is particularly convenient when generating *eye diagrams* from an NRZ electrical signal.

The clock regeneration algorithm will automatically set a “trigger level” based either on the mean signal level if it detects an NRZ signal or 50% of the pulse peak amplitude if otherwise.

### 9.7.2 Parameters

- Display Instrument** The *Oscilloscope* will list all FacetPower instruments here – choose one to display and analyse – typically this will be the output power of the laser.
- Sync Trigger** This defines the sync/clock source. You can choose either *Display Instrument* to regenerate a clock from the instrument signal you are analysing, or set a *Manual Period*, or alternatively extract a sync signal from one of the electrical signals driving the device. You can also turn the sync trigger off (*None (Single Scan)*) and just plot whole signal.
- clockPeriod** [ns] If you have chosen *Manual Period* in the **Sync Trigger**, then set this to the clock period of the signal you used in the simulation. If you have chosen an automatic sync, then the clock period that has been extracted by the sync algorithm is displayed here.
- delay** You can use this parameter to shift the signal along the display window. The delay is defined as a fraction of the clock period. To get a traditional eye diagram you should generally set this to 0.5.

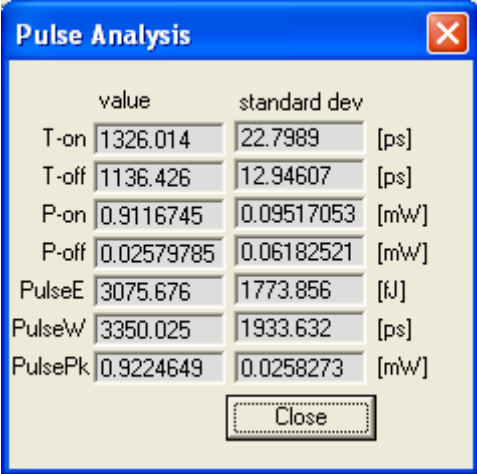
- sampleMinT** [ns] Often early parts of the signal will have transients that you want to ignore. You can set this to some non-zero value to skip earlier parts of the measured signal. Leave it blank to start displaying signal from t=0.
- sampleMaxT** [ns] Similar to above – will ignore signal parts later than this value. Leave it blank to display right to the end of the signal.
- graphMinX graphMaxX** Use these to over-ride the default graph x-axis scale.
- graphMinY graphMaxY** Use these to over-ride the default graph y-axis scale.
- switchLevel** [mW] this defines the signal level at which it is assumed a digital signal is switching from “off” state to “on” state or vice versa. Its value is indicated on the oscilloscope view by a horizontal grey line. The value is used for the *Pulse Analysis* results (see below). If you leave this parameter blank then the routine will make an intelligent guess about a suitable value.
- numBins** this defines the number of histogram bins to use for the histogram plots.

### 9.7.3 Print and Export

To print or export an image of your plot, right-click on the plot and a menu will appear with the options /Print, /Print Setup and /Export. See SciGraph documentation (§16) for details on using these routines.

### 9.7.4 Pulse Analysis

The *Oscilloscope* panel provides a comprehensive analysis of switch timings, jitter (standard dev of switch timings), pulse width etc. *The analysis results are updated each time you refresh the oscilloscope display or plot a histogram.* Note that some of the displayed quantities are useful only for NRZ signals and others only for pulse-like signals. To access the *Pulse Analysis* results panel (see figure below), click the **Analysis** button on the *Oscilloscope* panel.



	value	standard dev	
T-on	1326.014	22.7989	[ps]
T-off	1136.426	12.94607	[ps]
P-on	0.9116745	0.09517053	[mW]
P-off	0.02579785	0.06182521	[mW]
PulseE	3075.676	1773.856	[fJ]
PulseW	3350.025	1933.632	[ps]
PulsePk	0.9224649	0.0258273	[mW]

Close

The analysis will display the mean value and the standard deviation for each quantity listed below.

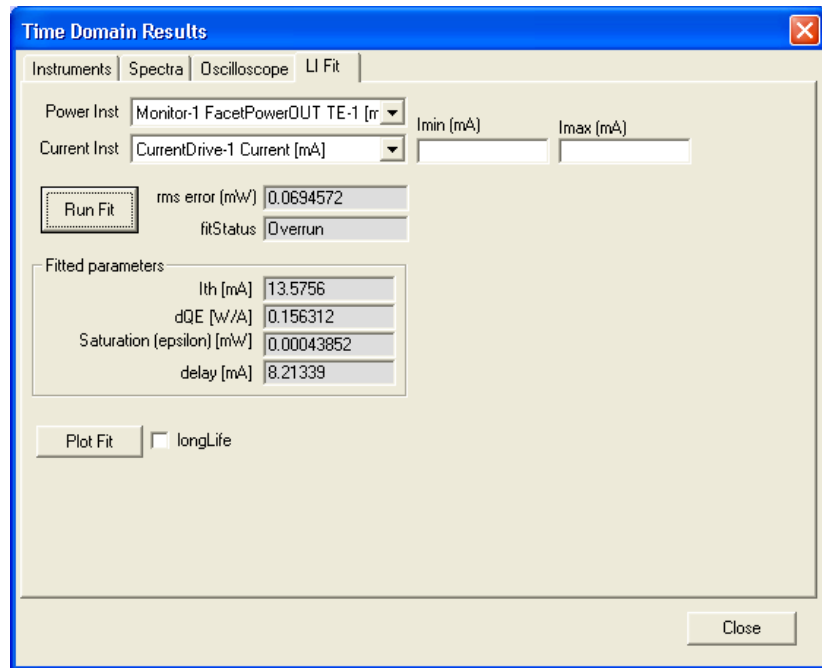
<b>T-on</b>	[ps] switch-on time – the time, as measured from the start of the clock period, at which the signal transits upwards through the <b>switchLevel</b> . (standard dev measures the jitter)
<b>T-off</b>	[ps] as <b>T-on</b> , but for transit downward through <b>switchLevel</b> .
<b>P-on</b>	[mW] the average power of samples above the <b>switchLevel</b> . The rms variation is also displayed. (Not generally meaningful if you are analysing a pulse).
<b>P-off</b>	[mW] the average power of samples below the <b>switchLevel</b> .
<b>Pulse-E</b>	[fJ] this conveniently measures the pulse energy, integrating the pulse from its peak down to 10% of its peak. (Not generally meaningful with an NRZ signal.)
<b>Pulse-W</b>	[ps] assuming you have a nominally constant single pulse in each clock cycle rather than an NRZ bit stream, this conveniently measures the pulse width – the time between the transition above <b>switchLevel</b> to transition below <b>switchLevel</b> again. If you did not set an explicit value for <b>switchLevel</b> then the routine will attempt to set <b>switchLevel</b> to 50% of the pulse peak value, so that <b>Pulse-W</b> will give you the pulse FWHM. (Not generally meaningful with an NRZ signal.)
<b>Pulse-Pk</b>	[mW] the pulse peak power.

#### 9.7.5 Histogram Plots

Using the drop-down list box at bottom-left of the panel, you can plot a histogram of any of the quantities calculated by the *Pulse Analysis*, as described above. You can control the histogram resolution through the **numBins** parameter – larger **numBins** value will give you a higher resolution. You can control the histogram range by entering values for **graphMinX** and **graphMax**.

## 9.8 The LI Fit Panel

The *LI Fit* panel (see figure below) is used to produce light-intensity curves for laser devices which can then be fitted so as measure the *threshold current* **I<sub>th</sub>**, *differential quantum efficiency* **dQE** and other parameters.



### 9.8.1 Running a Fit

Before running the fit, it is important ensure that you have run an appropriate simulation. Your simulation will need one instrument monitoring the current into your laser (i.e. the signal you are increasing to generate the LI curve) and an instrument monitoring the light output. The (ramp) current should start a little way below threshold ( $0.8 \times I_{th}$  should do) and run a fair way above threshold ( $1.5x$  to  $5x I_{th}$  or more should do). As you may not know the threshold current when you run the simulation for the first time, it is best to start off with an initial current of zero and then run an *LI Fit* to obtain an approximate value for the threshold which can then be used to determine the appropriate current limits. These limits can then be either entered directly in the **Imin (mA)** and **Imax (mA)** boxes or used when running the next simulation.

- When you have run an appropriate simulation, in the *LI Fit* panel, set the **Current Instrument** and **Power Instrument** to the relevant instruments (they should default to something sensible but you may want something different).
- Set the current range limits to be used for the fit in the **Imin (mA)** and **Imax (mA)** boxes – points outside this range will be ignored during the fit. If left blank the fit will use the current limits of the simulation.
- Then click the **Run Fit** button to run the fitting procedure.

Once the LI curve has been fitted, the fit can be plotted by clicking the **Plot Fit** button. This shows the actual LI curve of the laser device and the fitted LI curve produced by the fitting algorithm.

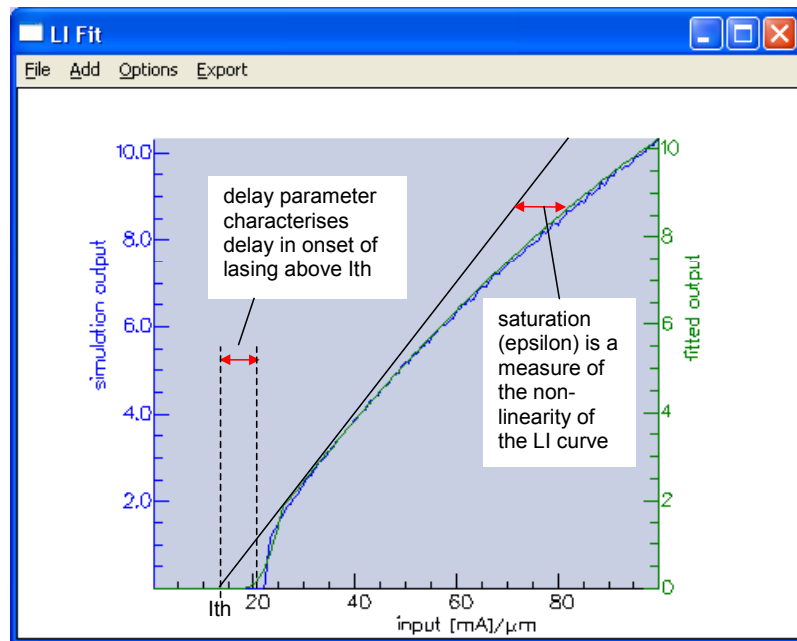
### 9.8.2 Fitted Parameters

The fit routine will give you the following results, some of which are illustrated in the figure below:

**Ith**                      the threshold current



<b>dQE</b>	the differential quantum efficiency $dP/dI$ – measured from the above-threshold slope of the linear part of the fit
<b>saturation</b>	a measure of the non-linearity of the curve, presumably due to gain saturation or heating effects.
<b>delay</b>	the fitting algorithm also includes a delay parameter to compensate for the delay often seen in running an LI simulation rapidly in the time domain.
<b>rms error</b>	this is a measure of how good the fit was – the average error between the fit and the simulation curve.
<b>fitStatus</b>	gives the convergence status of the fit, this should usually display <i>Overrun</i> .



## 9.9 Inspecting the Device State

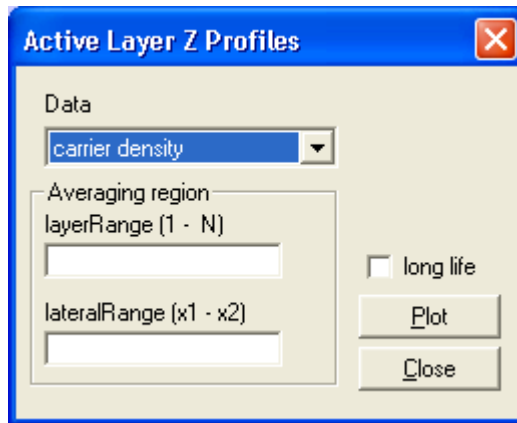
In the device window, when a simulation is paused or finished, you can inspect the various active layer quantities of the active waveguide sections in your device, both those that vary along the length of the device (*Z-profiles*) and those that vary in the lateral direction (*X Section states*), e.g. optical field intensity and the carrier density.

### 9.9.1 Z-Profiles

Active layer data

To view the *Z-profiles* of the active layer quantities of a waveguide section,

- Right click on the section and select */Z-profiles/Active layer data*. This brings up the following *Active Layer Z Profiles* box:



This box allows you to select and plot the z-dependence of the following quantities:

- carrier density in the junction
- material gain of the junction
- junction temperature
- junction current density
- junction current (see below).
- junction voltage

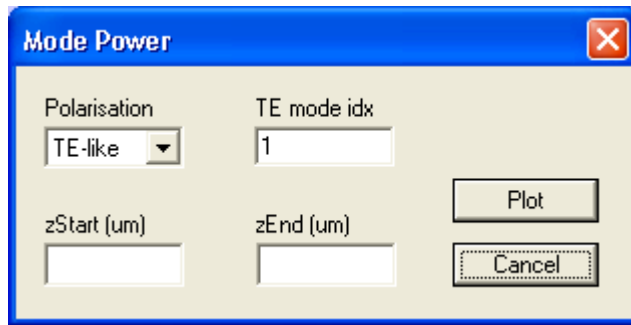
You may either measure the chosen quantity at a given point in the cross-section (given active layer and lateral position) or you can average the quantity over several layers and/or across a range of lateral positions. If you wish to measure at a point then leave the **layerRange** and **lateralRange** properties blank. If for example you enter “1 - 3” in **layerRange** (without the quotes) and “-1.5 - -1.0” in **lateralRange** then the routine will average the quantity over the range  $-1.5 < x < -1.0$  and over layers 1,2 and 3. Notice carefully the syntax – the min and max values of the range are separated by a “-” (minus) character. Note that the origin of the **lateralRange** is in the centre of the waveguide. The numbering and lateral ranges of the active layer(s) is conveniently summarised in the X-section

Also note that the junction current quantity works slightly differently from the others – if you give a lateral range then you get the integrated (or summed) current over the lateral range rather than an average. If you do not give a lateral range then you get the junction current of the *CoreCell* (see §4.9.1.2 and §11.2) at that point – you need to be aware of the dimensions of the *CoreCell* to make sense of this number. Note also that the current quantity is per unit length in the z-direction, i.e. the *CoreCell* current is divided by the length of the *z-element*.

Mode power

To view the *Z-profile* of the *Mode Power* in a waveguide section,

- Right click on the section and select */Z-profiles/Mode power*. This brings up the following *Mode Power* dialog box:



This allows you to plot the intensity of each of the optical modes as a function of z-position.

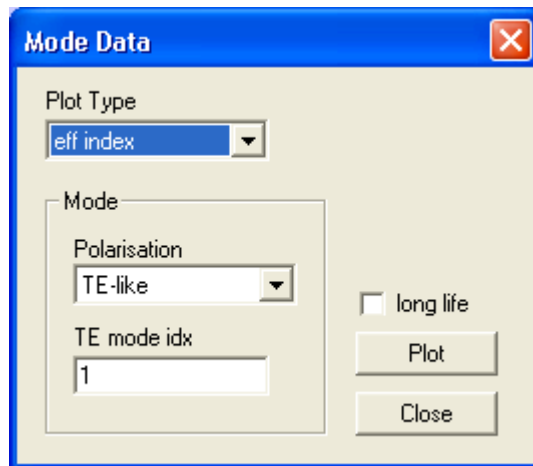
- Choose the mode you wish to plot by choosing its polarisation (**Polarisation**) and its mode index (**TE mode idx** or **TM mode idx** – the position of the mode in the list of modes of the specified polarisation, in order of decreasing effective index)
- Click **Plot**

By default the routine will plot the profile along the whole device, but you can set **zStart** and **zEnd** to limit the range. The plot contains the z-dependent mode intensity for the forward travelling power, the backward travelling power and the total (forward + backward).

Mode data

To view the *Z-profile* of the *Mode Data* in a waveguide section,

- Right click on the section and select */Z-profiles/Mode data*. This brings up the following *Mode Data* dialog box:



This allows you to plot the modal quantities (phase index, mode loss, group index, confinement factor) as a function of z-position.

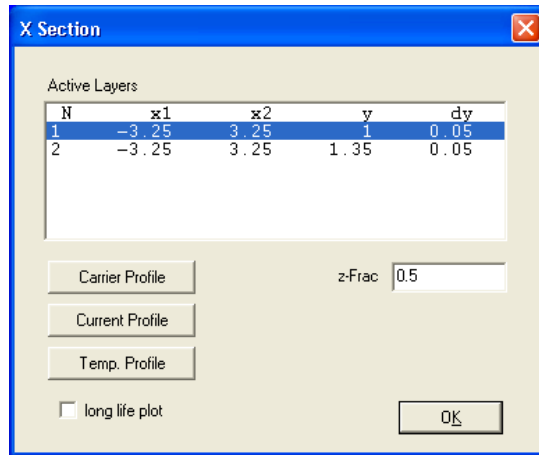
- Choose the mode you wish to plot by choosing its polarisation (**Polarisation**) and its mode index (**TE mode idx** or **TM mode idx** – the position of the mode in the list of modes of the specified polarisation, in order of decreasing effective index)
- Click **Plot**

*Mode Data* is particularly useful when using *taper sections* as it enables one to quickly check how a taper section has been divided into sub-section (see §4.9.2).

### 9.9.2 Cross-section State

To view the lateral variation of either the carrier density, current or temperature, which characterise the *cross-section state* (s) of the active layer(s) of a waveguide section,

- Right click on the section and select *View X Section State*. This brings up the *X Section* box:



- Select from the list the active layer for which the quantity profile will be plotted
- The list shows the numbers, and cross-sectional information of all the active layers in the section – this data itself is useful when producing Z-Profiles (see §9.9.1 above)
- Set the fractional position along the length (**z-Frac**) of the section at which you want the lateral profile to be measured
  - Then click on the appropriate button to plot the profile of the desired quantity

## The Material Database

The *Material Database* is loaded from the **.mat** file specified in the *RWG Window* (see §5.1.2). This file contains the full parameter description database of one or more materials and is the interface to the powerful materials description system. You may either construct one large central database containing all the materials of interest or else many smaller databases, one for each project or calculation. Although the basic parameters - e.g. refractive index - are fairly fixed and well known, you will probably find yourself defining many "special" materials or modifying basic ones, e.g. for lossy material.

The program is supplied with a file "refbase.mat". The materials defined therein are listed in Table 10-1 below. You will probably find that you quite frequently wish to make special materials, e.g. with different absorption coefficients. You are advised not to alter the refbase.mat file but use the INCLUDE instruction (see below) to include the materials into a new .mat file, from where parameters may be overwritten. Unless that is, you have better values for the reference parameters contained therein!

<i>Material name</i>	<i>x-parameter</i>	<i>Comments</i>
AlGaAs()	Al content	by default this uses an internal wavelength-dependent analytic model for AlGaAs(), after Adachi, defined at room temperature (§10.1.5)
GaAs()	must be zero	
InGaAsP()	As content	by default this uses an internal wavelength-dependent analytic model for InGaAsP() after Adachi and Weber, defined at 20°C (§10.1.6)
InGaAsPL()	<i>bandgap wavelength in um</i>	An alternative to InGaAsP()
InP()	<i>must be zero</i>	
InGaAlAs()	AllInAs fraction	(Al_0.48 In_0.52 As) <sub>x</sub> .(Ga_0.47 In_0.53 As) <sub>(1-x)</sub> by default this uses an internal wavelength-dependent analytical model after Mondry, defined at 20°C (§10.1.7)
Si()	-	loads internal model for silicon (see ANAL_SILICON1) at 20°C.
SiO2()	-	loads internal model for silica (see ANAL_SILICA1), no temperature dependence.
Aluminium()	-	loads internal model for aluminium, valid from 0.4 to 1.75um
gold()	-	loads internal model for gold, valid from 0.4 to 1.8um
Tungsten()	-	loads internal model for tungsten, valid from 0.4 to 1.8um
nickel()	-	loads internal model for nickel valid from 0.4 to 2.0um

Platinum()	-	loads internal model for platinum, valid from 0.4 to 2.0um
silver()	-	loads internal model for silver, valid from 0.4 to 2.0um
AIR()	-	ref index is 1.0, lossless
SLAB(x)	refractive index	dumb material, x is refractive index, lossless.

The **.mat** file contains a list of one or more materials. An example of a **.mat** file is given below. It must start with the signature line as shown. Each material contains a definition for the real refractive index function and the material absorption function. These functions can either be defined by simple general equations such as polynomials or you can load one of the built-in functions for the common semiconductor materials.

You can include another **.mat** file in your **.mat** file using the include instruction - see the example file below. You might, for example, include the reference database that you would not want to modify directly.

```
//-----
// Materials Parameters Database
//-----
<materbase(2.21)>      // file and version signature
INCLUDE rebase.mat    // include another .mat file
//-----
// define material for GaAs
BEGIN GaAs() // material name
RIX_POLYX      1  3.714 // N, Cn: refractive index poly ne(x)
MATLOSS_POLYX  1  0.0   // N, Cn: mat absorption poly alpha(x)
END
//-----
// AlGaAs(x) material at 0.85um, x is Al content
BEGIN AlGaAs(x) << GaAs() // material name and template
RIX_POLYX      2  3.714 -0.820 // N, Cn: refractive index poly
ne(x)
END
//-----
BEGIN AIR() // material name
RIX_POLYX      1  1.0 // N, Cn: refractive index poly rix(x)
MATLOSS_POLYX  1  0.0 // N, Cn: mat absorption poly alpha(x)
END
//-----
BEGIN air() << AIR // make lower case version too
END
```

Material parameters can generally be defined as a function of wavelength, a composition parameter "x" that you enter in the **.swg** files, and a temperature. Materials have a default temperature of 25°C, which some *Flags* can alter, or may be overridden by an explicit temperature of the device.

### 10.1.1 Material Parameter Reference Tables

The material definition

The format of the material declaration, beginning the definition of a new material, is:

```
BEGIN matname(x) [ << templatename ]
```

The name `matname` is the name of the material you will use to reference it in your structure (`.swg`) files. The name is case sensitive, i.e. GAAS and GaAs are distinct. The square brackets denote an optional parameter - the "template" material. If you specify a template material, the new material will be initialised with all the material parameters of the template material. Thus you can easily create new materials that are similar to other materials, perhaps with just one or two different parameters. You can also enter an existing material for `matname`. In this case the internal copy of the existing material will be updated by any parameter lines you enter. The latter would be useful in conjunction with an INCLUDED database.

**Parameter lines** There now follows a list of parameter lines. Each line starts with a parameter name flag from the list in Table 10-2 or Table 10-3. The flag is then followed by one or more data parameters, also shown in the tables, and finally an optional comment.

**Polarisation dependent gain** Certain gain parameters (in Table 10-3) can be polarisation dependent i.e. can be set separately for TE and TM polarisations by appending the suffixes `_TE` and `_TM` to the parameter flags. If, as in older PICWAVE versions, such suffixes are not used, identical parameters will be set for both polarisations.

**End line** The end of each material definition must be signalled by the word `END` as shown in the example.

2D-PolyData

A number of function definitions expect a *2D-Poly-Data* structure to follow. Given a 2-dimensional polynomial:

$$\text{Poly2D}(x, y) = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} C_{ij} x^{i-1} y^{j-1}$$

the format of the data structure is as follows:

```

x_min, x_max, y_min, y_max // validity range
of x and y
C11    C12    C13    ... C1Ny
C21    C22    C23    ... C2Ny
C31    C32    C33    ... C3Ny
...    ...    ...    ...
CNx1  CNx2  CNx3  ... CNxNy
ENDDDEF // end 2D polynomial parms

```

Valid Parameter Combinations

Certain combinations of material parameter flags listed in the tables below are nonsense or are not allowed. The three (real) refractive index polynomials `RIX_POLYX`, `RIX_POLYL`, `RIX_POLYT` make up **one** function internally and as such you cannot combine for example `RIX_POLYX` with `RIX_A1L` since `RIX_A1L` is a separate function.

The material loss is represented by a **second** function so you can combine any of the MATLOSS\_POLY flags with RIX\_A1L, ANAL\_ALGAAS and ANAL\_INGAASP1. The metal flags must define a complex refractive index, so define both the (real) refractive index function and the material loss function and thus cannot be used with any of the other refractive index or loss flags.

<b>Table 10-2 Material Parameter – refractive index related</b>					
<i>Flag</i>	<i>Units</i>	<i>param1</i>	<i>param2</i>	<i>param3</i>	
RIX_POLYX	[-]	N	mu0	mu1	
	Refractive index polynomial as a function of x-parameter: $\mu_p(x) = \mu_0 + \mu_1 \cdot x + \mu_2 \cdot x^2 + \dots$ N is the number of terms (<10). Note that the actual refractive index is: $\mu(x, \lambda, T) = \mu_p(x) + \mu_l(\lambda) + \mu_t(T)$				
RIX_POLYXY	[-]	nx	ny		
	{ RIX_POLYXY <b>2D-Poly-Data</b> follows on next lines }				
	Refractive index polynomial as a function of x-composition and y-composition parameters: $\mu_p(x,y) = \text{Poly2D}(x,y)$				
RIX_POLYL	[-]	N	mu0	mu1 (1/μm)	
	refractive index polynomial as a function of wavelength: $\mu_l(\lambda) = \mu_0 + \mu_1 \cdot (\lambda - \lambda_{\text{ref}}) + \mu_2 \cdot (\lambda - \lambda_{\text{ref}})^2 + \dots$ N is the number of terms (<10). Lambda is in μm. Note that the actual refractive index is: $\mu(x, \lambda, T) = \mu_p(x) + \mu_l(\lambda) + \mu_t(T)$				
LAMBDA_REF	[μm]	$\lambda_0$ (μm)			
	Reference wavelength for wavelength polynomials. (Defaults to zero).				
RIX_POLYT		N	mu0	mu1 (1/°C)	
	refractive index polynomial as a function of temperature: $\mu_t(\lambda) = \mu_0 + \mu_1 \cdot (T - T_0) + \mu_2 \cdot (T - T_0)^2 + \dots$ N is the number of terms (<10). T is in °C. Note that the actual refractive index is: $\mu(x, \lambda, T) = \mu_p(x) + \mu_l(\lambda) + \mu_t(T)$ . See also TEMP_REF.				
TEMP_REF		T <sub>0</sub> (°C)			
	reference temperature for e.g. RIX_POLYT and ALPHA_POLYT polynomials. (Defaults to 25°C if absent).				
MATLOSS_POLYX		N	a0 (1/cm)	a1 (1/cm)	
	material absorption polynomial as a function of x-parameter: $\alpha_p(x) = \alpha_0 + \alpha_1 \cdot x + \alpha_2 \cdot x^2 + \dots$ N is the number of terms (<10). Note that $\alpha = \alpha_p(x) + \alpha_l(\lambda) + \alpha_t(T)$				



MATLOSS_POLYXY	[1/cm]	nx	ny		
	Material absorption polynomial as a function of x-composition and y-composition parameters $\alpha_p(x,y)=\text{Poly2D}(x,y)$ nx, ny are the number of terms of the polynomial in x-dimension and y-dimension respectively				
MATLOSS_POLYL		N	a0 (1/cm)	a1 (1/cm/ $\mu\text{m}$ )	
	Material absorption polynomial as a function of wavelength: $\alpha_l(\lambda)=\alpha_0+\alpha_1(\lambda-\lambda_{\text{ref}}) + \alpha_2(\lambda-\lambda_{\text{ref}})^2 + \dots$ N is the number of terms (<10). Note that $\alpha = \alpha_p(x) + \alpha_l(\lambda) + \alpha_t(T)$				
MATLOSS_POLYT		N	a0 (1/cm)	a1 (1/cm/ $^{\circ}\text{C}$ )	
	Material absorption polynomial as a function of temperature: $\alpha_t(\lambda)=\alpha_0+\alpha_1(T-T_0) + \alpha_2(T-T_0)^2 + \dots$ N is the number of terms (<10). Note that $\alpha = \alpha_p(x) + \alpha_l(\lambda) + \alpha_t(T)$ . See also TEMP_REF.				
MATLOSS_EYFRAC		Frac			
	Ratio of absorption of different polarisations: $\text{Frac} = \alpha_{\text{Ey}}/\alpha_{\text{ave}}, \alpha_{\text{ave}}=(\alpha_{\text{Ex}}+\alpha_{\text{Ey}})/2$ Enter a value between 0.0 and 2.0, default is 1.0.				
RIX_AIL		c0	c1		
	Defines a wavelength dependent refractive index as: $\mu = \mu(\lambda) = \sqrt{1 + \frac{c_0 \cdot \lambda^2}{\lambda^2 - c_1}}$				
ANAL_ALGAAS		T ( $^{\circ}\text{C}$ )	type		
	(For passive materials only) Defines a material with a refractive index function $\mu(x,\lambda,T)$ and bandgap according to S. Adachi's model for AlGaAs. Note that material absorption is not defined. If type is zero then the x-parameter is Al content, otherwise the x-parameter is the bandgap wavelength. The temperature parameter is currently ignored (equations are defined for room temperature). See §10.1.5.				
ANAL_INGAASP1		T ( $^{\circ}\text{C}$ )	type		
	(For passive materials only) Defines a material with a refractive index function $\mu(x,\lambda,T)$ and bandgap according to Adachi/Weber model for InGaAsP. Note that material absorption is not defined. If type is zero then the x-parameter is As content, otherwise the x-parameter is the bandgap wavelength (in microns). The model is loaded at the temperature specified. NB. The temperature is not used for any other purpose. See §10.1.6.				

ANAL_INGAALAS1					
	<p>(For passive materials only)</p> <p>Defines a material with a refractive index function <math>\mu(x,\lambda,T)</math> according to the model by Mondry et al<sup>2</sup> for InGaAlAs lattice matched to InP. Note that material absorption is not defined. The composition formula is: <math>(Al_{0.48}In_{0.52}As)_x.(Ga_{0.47}In_{0.53}As)_{(1-x)}</math> and x is the “x-parameter”. <math>\mu(x,\lambda)</math> is accurate to about 5E-3 in the range (x:0..1), (<math>\lambda</math>:1.0<math>\mu</math>m...2.0<math>\mu</math>m). Mondry does not specify the measurement temperature.</p> <p>Also defines bandgap according to Olega<sup>3</sup> eqn. 1.</p>				
ANAL_SILICON1		T (°C)			
	<p>Defines a material with a refractive index function <math>\mu(\lambda)</math> of silicon according to Li<sup>4</sup>. Note that material absorption is not defined. The model is loaded at the temperature specified. NB. The temperature is not used for any other purpose. Valid approximately in the range <math>\lambda = 1.2\mu</math>m to 14<math>\mu</math>m, T=-50°C to 100°C</p>				
ANAL_SILICA1					
	<p>defines a material with a refractive index function <math>\mu(\lambda)</math> of silica according to Agrawal<sup>5</sup>. Note that material absorption is not defined. The model does not include a temperature dependence.</p>				
ANAL_ALUMINIUM					
	<p>defines a material with a complex refractive index function <math>\mu(\lambda)</math> for aluminium. Valid from 0.4 to 1.75<math>\mu</math>m</p>				
ANAL_ALUMINIUM2					
	<p>defines an alternative aluminium data set for a complex refractive index function <math>\mu(\lambda)</math>. Valid from 0.25 to 6.2<math>\mu</math>m.</p>				
ANAL_GOLD					
	<p>Ditto for gold. Valid from 0.4 to 1.8<math>\mu</math>m</p>				
ANAL_NICKEL					
	<p>Ditto for nickel. Valid from 0.4 to 2.0<math>\mu</math>m</p>				
ANAL_PLATINUM					
	<p>Ditto for platinum. Valid from 0.4 to 2.0<math>\mu</math>m</p>				
ANAL_SILVER					
	<p>Ditto for silver. Valid from 0.4 to 2.0<math>\mu</math>m</p>				
ANAL_TUNGSTEN					
	<p>Ditto for tungsten. Valid from 0.4 to 1.8<math>\mu</math>m</p>				

<sup>2</sup> M.J. Mondry et al, IEEE Phot. Tech. Lett. vol. 4, pp. 627-630, 1992.

<sup>3</sup> D. Olega, T.Y Chang, E. Silbert et al, APL v41 (5), pp. 476-8, 1982.

<sup>4</sup> H.H. Li, J. Phys, J. Phys. Chem. Ref. Data, vol. 9, pp. 561, 1980.

<sup>5</sup> G.P. Agrawal, 'Nonlinear Fiber Optics', eqn. (1.2.6).

**Table 10-3 Active Material Parameters – active layer related: gain, carrier properties, noise etc**

Flag	Unit	Param 1	Param 2	Param 3	Param 4	Param 5
GAIN_POLYN_TE	1/s	No [1/cm <sup>3</sup> ]	G [cm <sup>3</sup> /s]	m	c2	
		Polynomial gain function for the TE polarisation $g_0(N) = GN_0 \cdot (-1 + n + c_2 \cdot n^2 + c_3 \cdot n^3 \dots c_m n^m), m \leq 10, n = N / N_0$ This takes a varying number of extra parameters; if m=2 then it requires c2, if m=3 then c2, c3 etc.				
GAIN_POLYN_TM	1/s	No [1/cm <sup>3</sup> ]	G [cm <sup>3</sup> /s]	m	c2	
		As GAIN_POLYN_TE but for the TM polarisation				
GAINL_POLYN_TE	[1/cm]	No [1/cm <sup>3</sup> ]	G [cm <sup>2</sup> ]	m	c2	
		As GAIN_POLYN_TE but defines the material gain per unit length rather than per unit time.				
GAINL_POLYN_TM	[1/cm]	No [1/cm <sup>3</sup> ]	G [cm <sup>2</sup> ]	m	c2	
		As GAINL_POLYN_TE but for the TM polarisation				
GAIN_LOGN_TE	[1/s]	No [1/cm <sup>3</sup> ]	G [cm <sup>3</sup> /s]			
		Logarithmic gain function for the TE polarisation: $g_0(N) = \Gamma_v G \cdot N_0 \cdot \ln(N / N_0), \quad N > N_0 \quad g_0(N) = \Gamma_v G \cdot N_0 \cdot (N / N_0 - 1), \quad N < N_0$				
GAIN_LOGN_TM	[1/s]	No [1/cm <sup>3</sup> ]	G [cm <sup>3</sup> /s]			
		As GAIN_LOGN_TE but for the TM polarisation				
GAINL_LOGN_TE	[1/cm]	No [1/cm <sup>3</sup> ]	G [cm <sup>2</sup> ]			
		As GAIN_LOGN_TE but defines the material gain per unit length rather than per unit time.				
GAINL_LOGN_TM	[1/cm]	No [1/cm <sup>3</sup> ]	G [cm <sup>2</sup> ]			
		As GAINL_LOGN_TE but for the TM polarisation				
GAINL_FUNCNT	[1/s]	No [1/cm <sup>3</sup> ]	G [cm <sup>3</sup> /s]	POLY2D or SPLINE2D flag	nT	nN
		Followed by <b>2D-Poly-Data</b> or 2D-Spine-Data (latter used may be used in gain models written by HAROLD)				
		Polynomial gain function $g_0(N, T) = \Gamma_v GN_0 \cdot poly2D(T - T_{ref}, N / N_0)$ nT, nN are the number of temperatures and carrier densities i.e. dimensions of polynomial/spline to follow on next line.				

GAINL_FUNCNT	[1/cm]	No [1/cm <sup>3</sup> ]	Go [cm <sup>2</sup> ]	POLY2D SPLINE2D flag	or	nT	nN
		As GAIN_POLYNT but defines the material gain per unit length rather than per unit time.					
GAIN_LAMFUNC		g2[1/um <sup>2</sup> ]	λ <sub>0</sub> [um]	lamN1, lamN2 [um]		lamT1 [um/°C], lamT2 [um/°C <sup>2</sup> ]	
		Gain curvature, describing a quadratic dependence of the gain with wavelength, with a constant curvature. See also §10.1.3 $g_2(\lambda, N, T) = G_0 \cdot N_0 \cdot g_2 \cdot (\lambda - \lambda_{pk}(N, T))^2$ $\lambda_{pk}(N, T) = \lambda_0 + \Delta n \cdot (\text{lamN1} + \Delta n \cdot \text{lamN2}) + \Delta T \cdot (\text{lamT1} + \Delta T \cdot \text{lamT2})$ $\Delta n = N/N_0 - 1, \Delta T = T - T_{ref} \text{ (see TEMP\_REF)}$ Go, No are obtained from the gain peak definition, e.g. GAIN_POLYN.					
GAIN_LAMNT	[um]	POLY2D or SPLINE2D flag	nT	nN			
		Followed by <b>2D-Poly-Data</b> or 2D-Spine-Data (latter used may be used in gain models written by HAROLD)					
		Wavelength of gain peak $\lambda_{pk} = \text{Poly2D}(T - T_{ref}, N/N_0 - 1)$ nT, nN are the number of temperatures and carrier densities i.e. dimensions of polynomial/spline to follow on next line.					
GAIN_CURVNT		POLY2D or SPLINE2D flag	nT	nN			
		Followed by <b>2D-Poly-Data</b> or 2D-Spine-Data (latter used may be used in gain models written by HAROLD)					
		Defines gain curvature parameter g <sub>2</sub> dependence on carrier density and temperature. $g_2(T - T_{ref}, N/N_0 - 1) = G_0 \cdot N_0 \cdot \text{Poly2D}(T - T_{ref}, N/N_0 - 1)$ Go, No are obtained from the gain peak definition, e.g. GAIN_POLYNT. nT, nN are the number of temperatures and carrier densities i.e. dimensions of polynomial/spline to follow on next line.					
IMPORT_GAIN_SPECTRA_TE		relative path and file name					
		Imports TE gain spectra from specified file (path relative to material database file) and sets material to use fitted multiple Lorentzian gain model for TE poln (modes with TE-fraction > 50%) (see §8.3.1)					
IMPORT_GAIN_SPECTRA_TM		relative path and file name					
		As IMPORT_GAIN_SPECTRA_TE but for TM polarisation					

GAIN_EPS	[cm <sup>3</sup> ]	$\epsilon_{\text{sat}}$ [cm <sup>3</sup> ]				
		Gain saturation parameter, reduces $g_{\text{pk}}$ as follows: $g_{\text{pk}}' = g_{\text{pk}} / (1 + \epsilon_{\text{sat}} \cdot P)$ , where P is the local photon density where $g_{\text{pk}}$ is given by GAIN_POLYN, GAIN_LOGN GAIN_POLYNT etc				
T_EPS	[ps]	$t_{\text{eps}}$ [ps]				
		Time constant for the quantity $\epsilon_{\text{sat}} \cdot P$ (see GAIN_EPS above). If T_EPS is undefined (default), $\epsilon_{\text{sat}} \cdot P$ will change instantaneously i.e. $\lim t_{\text{eps}} \rightarrow 0$ .				
RADRECOMB	[1/ps]	$\tau_0$ [ps]	$B_r$ [cm <sup>3</sup> /s]	$C_r$ [cm <sup>6</sup> /s]		
		Defines radiative recombination rate: $dN / dt = -N / \tau_0 - B_r \cdot N^2 - 10^{12} C_r \cdot N^3$ , N in 1/um <sup>3</sup>				
NRADRECOMB	[1/ps]	$\tau_0$ [ps]	$B_n$ [cm <sup>3</sup> /s]	$C_n$ [cm <sup>6</sup> /s]		
		Defines non-radiative recombination rate: $dN / dt = -N / \tau_0 - B_n \cdot N^2 - 10^{12} C_n \cdot N^3$ , N in 1/um <sup>3</sup>				
DIFFUSION		$D_0$ [cm <sup>2</sup> /s]	$D_N$ [cm <sup>5</sup> /s]			
		Diffusion coefficient, with linear carrier dependence only: $D(N) = D_0 + D_N(N)$				
DIFFUSE_POLYXY			$n_x$	$n_y$		
		{ DIFFUSE_POLYXY <b>2D-Poly-Data</b> follows on next lines }				
		Composition dependence of diffusion coefficient: $D_{xy}(x,y) = \text{Poly2D}(x,y)$ $n_x, n_y$ denote the size of the 2D polynomial following. Together with the DIFFUSE_POLYT and DIFFUSE_POLYN data, this defines the diffusion coefficient as: $D(x,y,N,T) = D_{xy}(x,y) + D_T(T-T_{\text{ref}}) + D_{Nc}(N/N_0)$				
DIFFUSE_POLYT		M	$c_1$	$c_2$	... $c_m$	
		Temperature dependence of diffusion coefficient. $D_T(\Delta T) = \text{Poly1D}(\Delta T) = c_1 + c_2 \cdot \Delta T + \dots + c_m \Delta T^{(m-1)}$ , $\Delta T = T - T_{\text{ref}}$ Together with the DIFFUSE_POLYXY and DIFFUSE_POLYN data, this defines the diffusion coefficient as: $D(x,y,N,T) = D_{xy}(x,y) + D_T(T-T_{\text{ref}}) + D_{Nc}(N/N_0)$ .				
DIFFUSE_POLYN		M	$c_1$	$c_2$	... $c_m$	
		Carrier dependence of diffusion coefficient $D_N(n) = \text{Poly1D}(n) = c_1 + c_2 \cdot n + \dots + c_m n^{(m-1)}$ , $n = N/N_0$ See also DIFFUSE_POLYT and DIFFUSE_POLYXY				

LEF	[-]	$\alpha_{lef}$				
		Linewidth enhancement factor				
RIXNESHIFT_POLYN	[-]	Nref [1/cm <sup>3</sup> ]	polyLen	c <sub>1</sub> [cm <sup>3</sup> ]	c <sub>2</sub> [cm <sup>9</sup> ]	...
		Change of real index with carrier density $\Delta n_r(N_e) = c_1 \cdot \Delta N + c_2 \cdot \Delta N^2 + \dots + c_{polyLen} \cdot \Delta N^{polyLen}$ $\Delta N = N_e - N_{ref}$				
EGAP	[eV]					
		Define a constant bandgap energy				
EGAP_POLYXY	[eV]	n1	n2			
		{ EGAP_POLYXY <b>2D-Poly-Data</b> follows on next lines }				
		Defines the dependence of the bandgap on the material composition parameters x and y. $E_{g_{xy}}(x,y) = Poly2D(x,y)$ Together with the EGAP_POLYT and EGAP_POLYN data, this defines the bandgap: $E_g(x,y,N,T) = E_{g_{xy}}(x,y) + E_{gT}(T-T_{ref}) + E_{gN_e}(N/N_o)$				
EGAP_POLYT	[eV]	m	c <sub>1</sub>	C <sub>2</sub>	...C <sub>m</sub>	
		Defines the dependence of the bandgap on temperature: $E_{gT} = Poly1D_T(T-T_{ref}) = c_1 + c_2 \cdot \Delta T + \dots + c_m \Delta T^{(m-1)}$ Use with EGAP_POLYXY – see above.				
EGAP_POLYN	[eV]	M	c <sub>1</sub>	c <sub>2</sub>	...C <sub>m</sub>	
		Defines the carrier dependence of the bandgap: $E_{gN_e} = Poly1D(n) = c_1 + c_2 \cdot n + \dots + c_m n^{(m-1)}$ , $n = N/N_o$ Use with EGAP_POLYXY – see above.				
DOSVB	[1/cm <sup>3</sup> ]	N <sub>vb</sub> [1/cm <sup>3</sup> ]				
		Effective density of states in valence band at 300K – used for <i>junction voltage</i> function. See Notes below.				
DOSCB	[1/cm <sup>3</sup> ]	N <sub>cb</sub> [1/cm <sup>3</sup> ]				
		Effective density of states in conduction band at 300K - used for <i>junction voltage</i> function. See Notes below.				

NOISE_LAMFUNC	[um]	noiseLambda0 [um]	noiseLamN1, noiseLamN2 [um]	noiseLamT1 [um/°C], noiseLamT2 [um/°C²]		
		<p>Defines quadratic dependence of peak wavelength of Lorentzian noise spectrum on carrier density and temperature. See also §10.1.3</p> $\text{noiseLambda}(N,T) = \text{noiseLambda0} + \Delta n \cdot (\text{noiseLamN1} + \Delta n \cdot \text{noiseLamN2}) + \Delta T \cdot (\text{noiseLamT1} + \Delta T \cdot \text{noiseLamT2})$ $\Delta n = N/N_0 - 1, \Delta T = T - T_{\text{ref}} \quad (\text{see TEMP\_REF})$ <p>Not obtained from the gain peak definition, e.g. GAIN_POLYN.</p>				
NOISE_WIDTHFUNC	[um]	halfWidth0 [um]	halfWidthN1, halfWidthN2 [um]	halfWidth T1 [um/°C], halfWidth T2 [um/°C²]		
		As NOISE_LAMFUNC except defines half-width of Lorentzian noise spectrum				
NOISELAMPK_FUNCNT	[um]	n1	n2			
		<p>Peak wavelength of Lorentzian noise spectrum</p> $\text{noiseLambda} = \text{Poly2D}(T - T_{\text{ref}}, N/N_0 - 1)$				
NOISEWIDTHL_FUNCNT	[um]	n1	n2			
		<p>Half-width of Lorentzian noise spectrum</p> $\text{noiseWidth} = \text{Poly2D}(T - T_{\text{ref}}, N/N_0 - 1)$				

Table 10-4 Material Parameters – miscellaneous quantities					
Flag	param 1	param 2	Param 3	Param 4	Description
THCON	sigma				Thermal conductivity in W/°C/m
RHO	$\rho(\text{Ohm}, \mu\text{m})$				Resistivity (ignored)
MXPARAM	ON   OFF	defaultVal	[min:max]	“label”	
	<p>Turns on/off the X material composition parameter. Enter either the word ON or OFF in param1, then a unit for the parameter in [] brackets, followed by a label in double quotes. For example:</p> <p>MXPARAM ON 0.0 [0:1] “x: Al fraction”</p> <p>MXPARAM ON 1.0 [0.8:1.27] “x: bandgap energy[eV]”</p> <p><b>defaultVal</b> is used if you do not supply a x-material-composition parameter when you specify this material.</p> <p><b>min</b> and <b>max</b> allow you to provide limits to prevent you accidentally using invalid x-composition parameters.</p> <p>The label that you specify is used to display in the user interface where an x-parameter is entered and for no other purpose. I.e. any unit you specify needs to be consistent with other functions defined elsewhere in the material file such as MATLOSS_POLYX, ANAL_ALGAAS (with type=1) that use the x-parameter.</p>				
MYPARAM	ON   OFF	[unit]	“label”		
	Turns on/off the Y material composition parameter. See MXPARAM for details.				

### 10.1.2 Refractive Index Functions

The real part of the refractive index is computed as:

$$\mu(x,y,\lambda,T) = \mu_p(x,y) + \mu_l(\lambda) + \mu_t(T)$$

where  $\mu_p$ ,  $\mu_l$ ,  $\mu_t$  are defined as described in Table 10-2 above.

The imaginary part of the refractive index is defined via the material absorption coefficient “alpha”:

$$\alpha(x,y,\lambda,T) = \alpha_p(x,y) + \alpha_l(\lambda) + \alpha_t(T)$$

which of course is related to the imaginary part of the refractive index via:

$$\alpha = 4.\pi.\text{Im}(\mu)/\lambda$$

In addition, the change of the real part of the refractive index with carrier density can be defined via the `RIXNESHIPT_POLYN` parameter data or alternatively by supplying a linewidth enhancement factor via the `LEF` parameter data.

### 10.1.3 The Gain Model

When using a *single Lorentzian gain model*, the material gain is calculated according to:

$$g(N, \lambda, T) = \frac{g_{pk}(N, T)}{1 + \varepsilon_{sat}(N)P} - G_0 N_0 g_2(N, T) \cdot (\lambda - \lambda_{pk}(N, T))^2$$

where  $g_2$  describes the curvature of the gain at the gain peak,  $G_0$ ,  $N_0$  are derived from the  $g_{pk}$  definition data (and may be polarisation dependent if specified, see §10.1.1),  $\varepsilon_{sat}$  describes the gain saturation and  $P$  is the optical power density. This function can be defined in units of 1/s or 1/cm – the program can use either and convert between the two as needed by reference to the group velocity. For 1/s units, the quantities  $g_{pk}$ ,  $g_2$ ,  $\lambda_{pk}$  and  $\varepsilon_{sat}$  are defined in the `.mat` file as follows:

```
gpk:    GAIN_POLYN or GAIN_LOGN or GAIN_POLYNT
λpk:    GAIN_LAMFUNC or GAIN_LAMNT
g2:     GAIN_LAMFUNC or GAIN_CURVNT
```

Alternatively for 1/cm units  $g_{pk}$  is defined using:

```
gpk:    GAINL_POLYN or GAINL_LOGN or GAINL_POLYNT
```

The gain saturation function is defined in the `mat` file by:

```
εsat:    GAIN_EPS
```

The mode gain is given by:

$$g_m(N, \lambda, T) = \Gamma_m g(N, \lambda, T)$$

where the confinement factor  $\Gamma_m$  is calculated internally by the program by computation of the waveguide mode.

Internally in the `PICWAVE` time-domain model, the effective gain has a near-Lorentzian shape, i.e. of the form:

$$g(\lambda) = g_p / [(1 + \tau^2 (\lambda^{-1} - \lambda_{pk}^{-1})^2)^{1/2}].$$

The gain curvature  $g_2$  and  $\tau$  are related by:

$$g_2 = 0.5\tau^2 / \lambda_{pk}^4.$$

### 10.1.4 Density Of States

$N_{cb}$ ,  $N_{vb}$  (**effective density of states**). For a simple system (one conduction band and LH, HH bands) these are given approximately by:

$$N_{cb} = 2 \cdot (2.\pi.m_e^*.kT/h^2)^{3/2}$$

$$N_{vb} = 2 \cdot (m_{lh}^{3/2} + m_{hh}^{3/2}) \cdot (2.\pi.kT/h^2)^{3/2}$$



where  $m_e^*$ ,  $m_{lh}^*$ ,  $m_{hh}^*$  are the effective masses of conduction, LH and HH bands respectively. The program expects effective densities at 300K and will add a  $T^{3/2}$  dependence internally.

#### 10.1.5 Built-in AlGaAs model

**(Note:** this function is useful for passive materials only, since it will not calculate refractive indices for energies above the band gap.)

<b>Material flag</b>	ANAL_ ALGAAS
<b>Material system</b>	$Al_xGa_{1-x}As$
<b>Composition Parameter mx</b>	“x” in material system shown above
<b>Composition Parameter my</b>	Unused
<b>Functions provided</b>	<ul style="list-style-type: none"> <li>• Real part of refractive index <math>-n_e(\lambda, T)</math></li> <li>• Bandgap - <math>E_g(\lambda, T)</math></li> </ul>
<b>Wavelength dependence</b>	Yes
<b>Temperature dependence</b>	Yes

The functions are taken from S. Adachi<sup>6</sup> are as follows:

$$n^2 = A_0 \left\{ f(X) + \frac{f(X_{so})}{2} \left( \frac{E_0}{E_0 + \Delta_0} \right)^{3/2} \right\} + B_0 + \frac{1}{\epsilon_0} \frac{\partial \epsilon_\infty}{\partial T} (T - 300) \quad \text{eqn.3.1}$$

$$\text{with } f(X) = X^{-2} [2 - \sqrt{1+X} - \sqrt{1-X}]$$

$$X = \hbar\omega / E_0$$

$$X_{so} = \hbar\omega / (E_0 + \Delta_0)$$

the constants  $A_0$  and  $B_0$  are given by

$$A_0(x) = 6.3 + 19.0x$$

$$B_0(x) = 9.4 - 10.2x$$

and the bandgap details by:

$$E_0 = q_e (1.424 + 1.594 + x(1-x)(0.127 - 1.310x))$$

$$\Delta_0 = q_e (1.765 + 1.115x + 0.370x^2) - E_0$$

The program will complain if you specify a photon energy above the bandgap, i.e. that gives  $X > 1.0$ . The temperature dependent term in  $n(X)$  is currently undefined (zero) for the AlGaAs equation.

#### 10.1.6 Built-in InGaAsP model

**(Note:** this function is useful for passive materials only, since it will not calculate refractive indices for energies above the band gap.)

<sup>6</sup> S. Adachi, J. Appl. Phys. Vol. 58, pp. R1-R29, 1985.

<b>Material flag</b>	ANAL_ INGAASP1
<b>Material system</b>	In <sub>1-x</sub> Ga <sub>x</sub> As <sub>y</sub> P <sub>1-y</sub> lattice matched to InP
<b>Composition Parameter mx</b>	“x” in material system shown above
<b>Composition Parameter my</b>	Unused
<b>Functions provided</b>	<ul style="list-style-type: none"> <li>• Real part of refractive index -n<sub>e</sub>(λ)</li> <li>• Bandgap - E<sub>g</sub>(λ)</li> </ul>
<b>Wavelength dependence</b>	Yes
<b>Temperature dependence</b>	Yes

The functions follow essentially from the Adachi formula above (eqn. 3.1), with parameters by Weber<sup>7</sup> as follows:

$$A_0 = 8.616 - 3.886y$$

$$B_0 = 6.621 + 3.461y$$

$$\frac{\partial \varepsilon_\infty}{\partial T} = 5.16 \times 10^{-4} \varepsilon_0 [K^{-1}] \text{ for InP}$$

$$E_0 = q_e (1.35 - 0.72y + 0.12y^2) + (T - 300) \frac{\partial \varepsilon_0}{\partial T} [J]$$

with T in Kelvin (see <sup>8</sup>), and its temperature dependence given by

$$\frac{\partial \varepsilon_0}{\partial T} = -q_e (3.18 - 0.41y + 0.61y^2) \times 10^{-4} [J / K]$$

the split off valence band gap at 300K is

$$\Delta_0 = q_e (0.118 + 0.225y) [J]$$

#### 10.1.7 Built-in InGaAlAs model

**(Note:** this function is useful for passive materials only, since it will not calculate refractive indices for energies above the band gap.)

The model follows the work of Mondry et al<sup>9</sup>. The model is summarised by the table below.

<sup>7</sup> J.P. Weber, IEEE J. Quant. Electron. Vol. 30, pp. 1801-1815, 1994, See appendix therein.

<sup>8</sup> G.P. Agrawal, N.K. Dutta, “Long Wavelength Semiconductor Lasers”, 1986, eqn. 4.7.1.

<sup>9</sup> M.J. Mondry et al, PTLett, vol 4, pp627-9, 1992.

<b>Material flag</b>	ANAL_INGAALAS1
<b>Material system</b>	(Al <sub>0.48</sub> In <sub>0.52</sub> As) <sub>x</sub> .(Ga <sub>0.47</sub> In <sub>0.53</sub> As) <sub>(1-x)</sub> lattice matched to InP
<b>Composition Parameter mx</b>	“x” in material system shown above
<b>Composition Parameter my</b>	Unused
<b>Functions provided</b>	<ul style="list-style-type: none"> <li>• Real part of refractive index -<math>n_e(\lambda)</math></li> <li>• Bandgap - <math>E_g(\lambda)</math></li> </ul>
<b>Wavelength dependence</b>	Yes
<b>Temperature dependence</b>	No – assumed room temperature

### 10.1.8 Metals

The material system includes built-in definitions for the complex refractive index of several metals as shown in Table 10-1. These functions are taken from the Handbook of Optical Constants of Solids (Academic Press, 1985) and are precise wavelength dependent fits to the data therein, using a cubic spline algorithm. The algorithm will give an error if you attempt to calculate a refractive index outside the specified wavelength range. Note that the Handbook’s tables contains data from several different sources and sometimes (e.g. in the case of silver) the sources do not agree. In this case the fit used here is a compromise function that is a well-behaved combination of the different data sets.

Complex refractive indexes for other metals must be defined by the user. Note that PICWAVE defines a complex refractive index by its real part plus the material absorption coefficient. The absorption coefficient and the imaginary refractive index are related by:

$$\text{Im}(\mu) = \frac{\alpha\lambda}{4\pi}$$

Remember that the material loss  $\alpha$  is usually defined in units of 1/cm and the wavelength in units of  $\mu\text{m}$ .

### 10.1.9 Inspecting Material Databases

Besides editing *Material Database* files directly, you can also plot material properties as defined by such files by using the *Materials Inspector*, details of which can be found in §14.2.

## The Theoretical Model

## 11.1 The Electrical Model

PICWAVE now features a travelling wave electrode model. The device may have one or more electrodes, and there may be more than one electrode in each section – so that they lie side by side.

The following diagram describes the electrical model of PICWAVE.

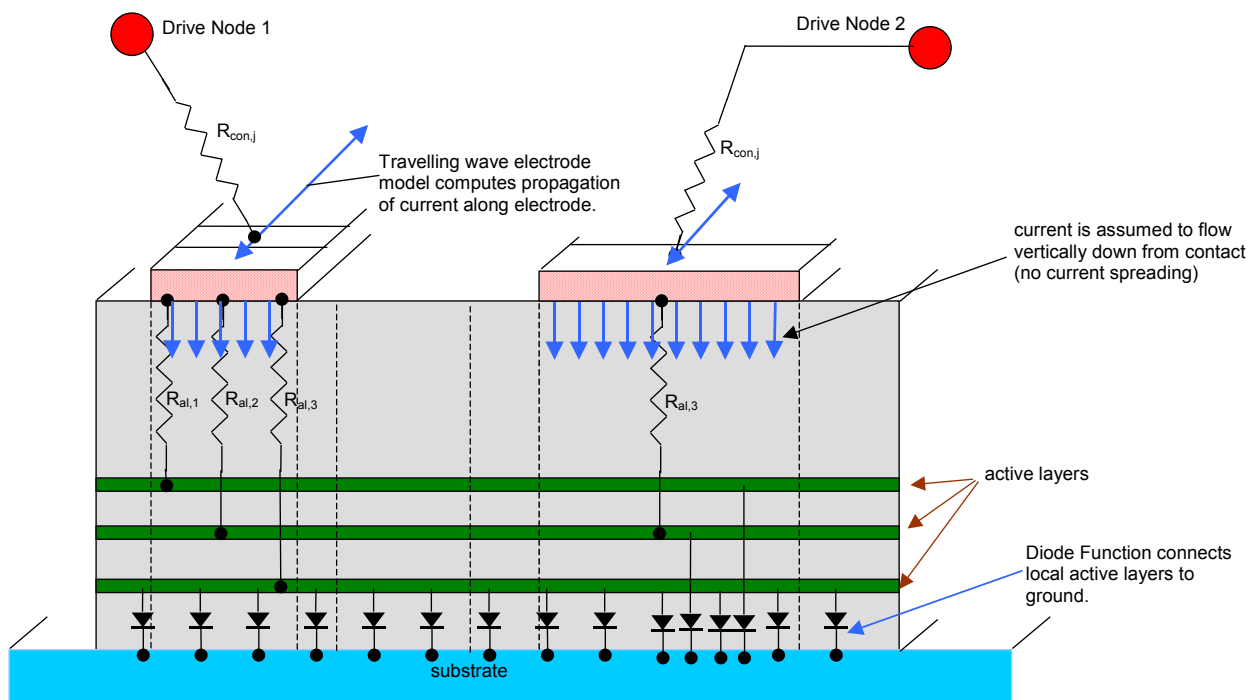


Figure 11-1 The Cross-Section, showing electrical model

## 11.1.1 The Connection Model

Each *contact* may be connected to a *current drive* or *voltage drive* via a resistor. You can connect only one *current drive/voltage drive* to each contact. Future versions may allow more complex connection models including capacitors and inductors. The contacts are assumed to be perfect conductors, i.e. equipotentials, unless you enable the *Travelling Wave Electrode Model*.

### 11.1.2 The Current Flow Model

The connection from the *contacts* to the *active layers* is defined by the *Current Flow Model*. In this version of the product, the model is a very simple one where current is assumed to travel directly down from the contact to an active layer, i.e. with no lateral or longitudinal spreading outside the active layer. A resistance  $R_{al}$  defines the connection from the contact to each active layer – a resistance per unit area - units of ohm.um<sup>2</sup>.

The *Current Flow Model* parameters may be edited via the **CurrentFlowModel** property of the *waveguide section*. The list of resistances  $R_{al}$  is found in the **alConnections** property of the *CurrentFlowModel*, with one connection for each active layer present in the RWG. The resistance **Ral** is define in units of ohm.um<sup>2</sup> and the actual resistance between the contact and each *CoreCell* is given by  $R_{al}/dx/dz$ , where dx and dz are the width and z-length of the cell respectively.

#### The Junction

Each active layer represents a current path to the substrate. The voltage across the active layer is dependent on the carrier density and is defined by a *junction voltage* function:

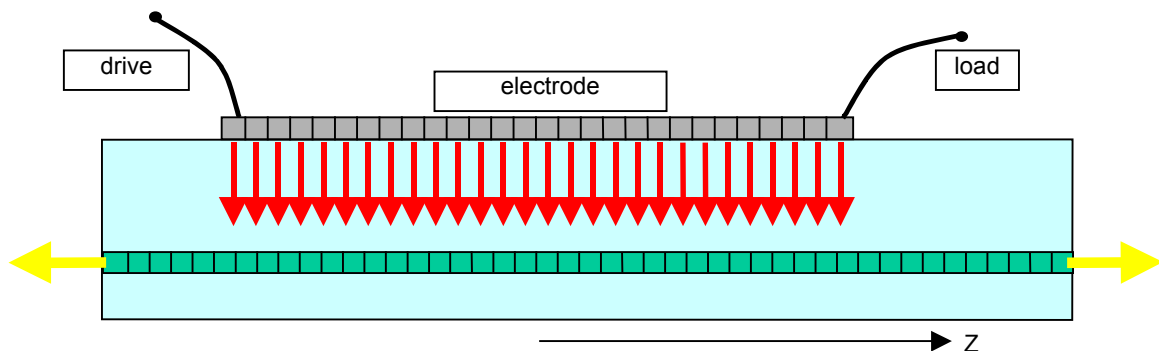
$$V(N_e) = \frac{E_g}{e} + \frac{kT}{e} \cdot \left( 2 \cdot \log\left(N_e / \sqrt{N_{cb} \cdot N_{vb}}\right) + c_1 \cdot N_e + c_2 \cdot N_e^2 \right)$$

where  $N_{cb}$  and  $N_{vb}$  are the effective density of states in the conduction and valence bands and  $c_1$ ,  $c_2$  are also functions of  $N_{cb}$  and  $N_{vb}$ . Each active layer forms a parallel circuit to the substrate as shown in the diagram above.

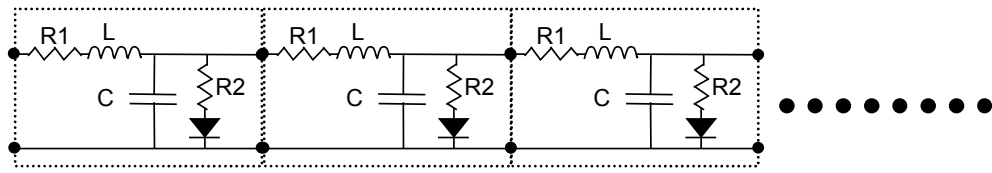
$N_{cb}$  and  $N_{vb}$  are defined by the quantities DOSCB and DOSVB in the material database.  $E_g$  is the bandgap and is defined either by EGAP or by the built-in material models (ANAL\_ALGAAS, ANAL\_INGAASP etc.)

### 11.1.3 The Travelling Wave Electrode Model

Contacts may be modelled by a more sophisticated *Travelling Wave Electrode Model*. In this case the contacts are approximated using a circuit model. This is shown schematically in the figure below. The drive signal may be applied at any point on the electrode and an optional terminating load at one end. In essence, the electrode is divided longitudinally into small elements and each element is approximated by an electrical circuit of resistors, capacitors and inductors.



The circuit is shown schematically in the figure below:



The characteristics of the electrode may be defined either by specifying

- a) the electrode resistance per unit length  $R_1$ ,  
and either
- b) the effective index of the travelling wave mode ( $n_{eff}$ ) and the microwave impedance of the electrode ( $Z$ ); or
- c) the values of  $L$  and  $C$ .

The junction voltage  $V_j$  across the diode and resistor  $R_2$  are calculated by the programme, thus you can't set the value manually. But the resistor  $R_1$ , the inductor  $L$ , and the capacitor  $C$  can be set manually (or left at the programme default values).

The values of  $R_1$ ,  $L$  and  $C$  are determined as follows:

$$R_1 = R'_1 \times dz, \quad L = L' \times dz, \quad \text{and} \quad C = C' \times dz,$$

where  $R'_1$ ,  $L'$ ,  $C'$  are defined in the *Contact Properties* editor (see §4.17.2).

So,  $R'_1$  is the conductor loss per unit length,  $L'$  is the inductance per unit length, and  $C'$  is the capacitance per unit length.

A parallel-plate capacitor model gives the values of  $R'_1$ ,  $L'$  and  $C'$ :

$$R'_1 = \frac{\rho}{Wt}, \quad C' = \frac{\epsilon_0 \epsilon_r W}{d}, \quad \text{and} \quad L' = \frac{\mu_0 \mu_r d}{W}$$

where  $\rho$  is the resistivity of the electrode,  $W$  is the electrode width,  $t$  is the electrode thickness,  $\epsilon_r$  is the relative dielectric constant, and  $d$  is the dielectric thickness between the signal electrode and the ground electrode.

Note: The impedance and effective index of the electrode and the associated inductance and resistance of the circuit models are frequency dependent to some extent. To model the full frequency dependence is a major simulation in itself. However this circuit model should give you a good approximation of your travelling wave electrode.

The Microstrip Calculator

You will find a calculator to compute appropriate values of  $R'_1$ ,  $L'$ ,  $C'$ ,  $n_{eff}$  and  $Z$  under the Tools menu: the *Microstrip Calculator* (see §14.3 for more details).

## 11.2 The Optical Model

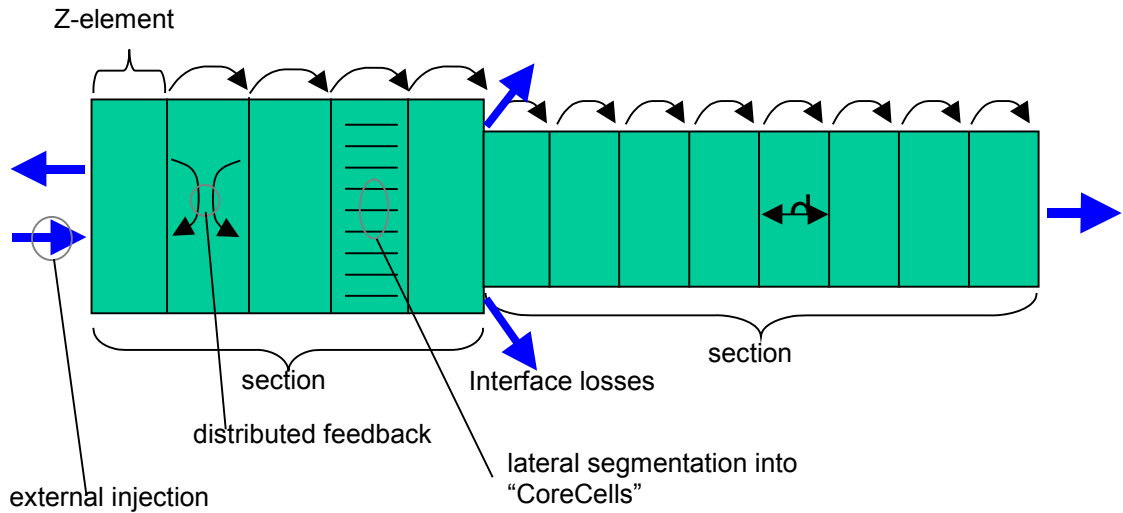
The optical model of PICWAVE is built round a 1D *time domain travelling wave (TDTW)* model. This 1D TDTW algorithm computes how the optical power of the waveguide mode travels up and down the laser or amplifier.

The figure below shows a schematic of the optical model. A *circuit* is composed of one or more *sections*. Each section is divided into one or more *z-elements* as shown. The *z-*

*element* is the smallest discretisation in the *z* direction. The division of the sections into *z-elements* is automatic and is controlled by **zStep** from the *TDTW Calculator* panel in such a way that all the *z-elements* of a section are of equal length.

At each time step *dt* a photon travels the length of one *z-element* as shown in the diagram. This implies that the length of each *z-element* is related to the time step by the group velocity

$$dz = v_g dt .$$



**Figure 11-2 Schematic of optical propagation model**

The time step *dt* must be the same for each section. This implies that **zStep** can be chosen for only one section, the *reference section*, – the algorithm decides an appropriate *dz* for other sections, taking into account the group velocity of each section. This also implies that the program will not in general be able to match the group velocity of every section perfectly – since the program must satisfy:

$$N \cdot dz = \text{section-length}, \quad N = \text{integer}$$

In general, the smaller you make *zStep*, the better able the program is to match the group velocity of each section. However a 10% error in the group velocity will not in general significantly affect your results.

CoreCells (active waveguides)

Each *z-element* may have one or more *active layers* as shown in Figure 11-1. Each active-layer is divided into one or more *CoreCells*. The *CoreCell* is the smallest unit of the model. Each *CoreCell* has a unique carrier-density, temperature, voltage etc. The model will take account of diffusion between the *CoreCells*, i.e. transverse diffusion. You can control the number of *CoreCells* by setting the **ncellW** parameter in the *waveguide section*. The *CoreCells* and lateral diffusion are shown schematically in the figure below.

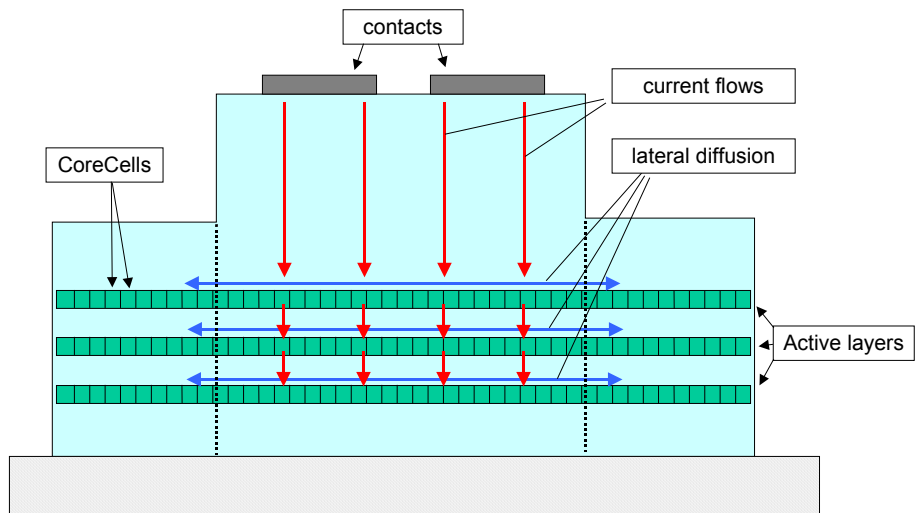


Figure 11-3 CoreCells and lateral diffusion

## 11.3 Algorithm Flowchart

The figure below shows a simplified algorithm flowchart of the TDTW engine. As you can see, the optical mode finder and Kappa Calculator are run once at the beginning of a simulation – i.e. they do not respond to changing conditions during the simulation.

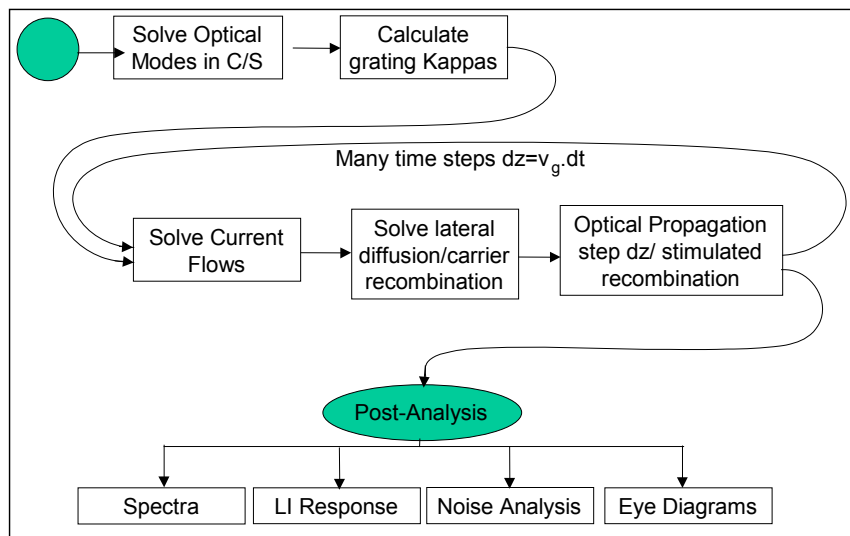


Figure 11-4 Algorithm Flowchart

## 11.4 The Gain Model

### 11.4.1 Single Lorentzian Model

PICWAVE's *single Lorentzian gain model*, as defined for a material in a material database file, uses an expression of the form:



$$g = g_{pk} (w/2)^2 / ((\omega - \omega_0)^2 + (w/2)^2)$$

where  $\omega_0$  is the optical frequency of the gain peak, and  $w$  is the FWHM. When you define a material in the material database, you define a gain peak  $g_{pk}$ , the position of the gain peak, and the gain curvature  $d^2g/d\omega^2$  at the gain peak. The program then fits a Lorentzian to these parameters. You should be aware of the difference between this and the gain in a real semiconductor. This is illustrated in the figure below.

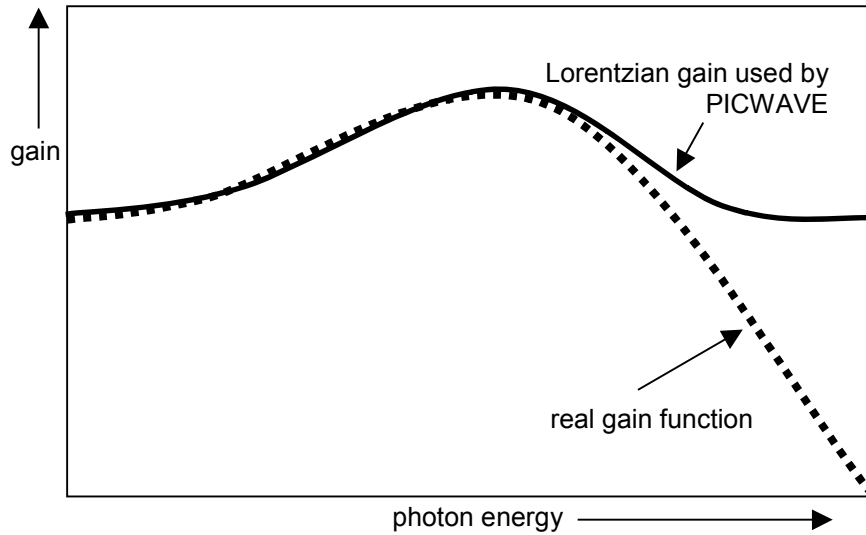


Figure showing form of gain function used in PICWAVE

Note that as the carrier density tends to transparency then it is preferable to use a gain that is flat rather than maintain the Lorentzian peak. Thus the program uses a constant linewidth at gains well above transparency but increases the linewidth gradually as the gain approaches transparency. This is illustrated in the figure below:

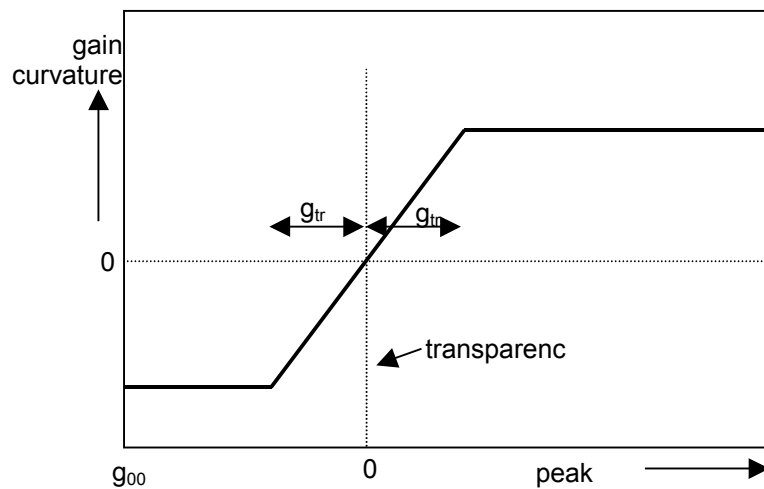


Figure showing form of gain peak curvature variation around transparency

The transition span  $g_{tr}$  is currently defined as a fraction of the (negative) gain  $g_{00}$  at a carrier density of zero. The fraction  $g_{tr}/g_{00}$  is currently fixed at 0.4.

This gain model will give a reasonable approximation of the real gain function around the gain peak wavelength and also at energies below the gain peak. But you should be aware that the model will not represent the absorption at high energies so well.

#### 11.4.2 Multiple Lorentzian Model

PICWAVE'S *multiple Lorentzian gain model*, employed by the *Multiple Lorentzian Gain Fitter* (see §8), uses several Lorentzians and so, compared with the single Lorentzian model, allows an accurate fit over a greater portion of the free spectral range. The figure below is an example of the type of fitting one would get when using multiple Lorentzians, which should be compared with the figure above for a single Lorentzian fit.

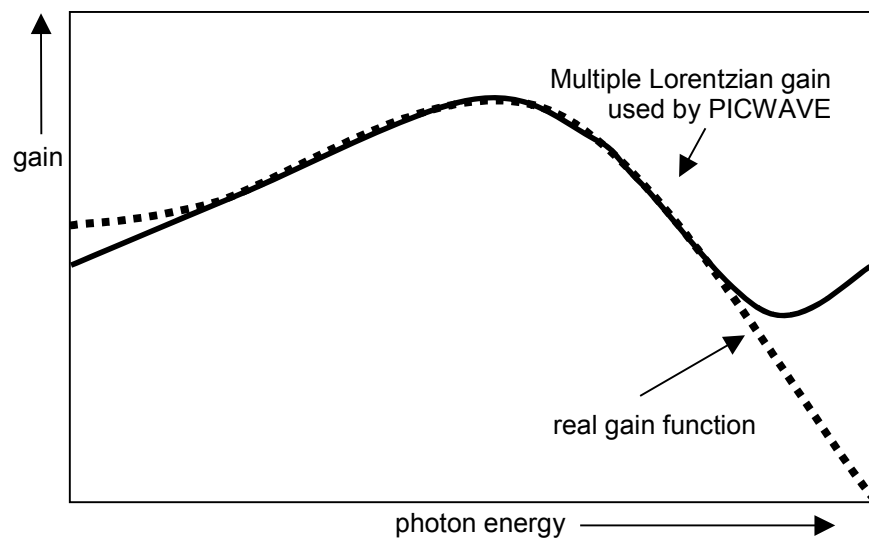


Figure showing form of multiple Lorentzian gain function used in PICWAVE

As with the single Lorentzian model, a fitted multiple Lorentzian gain spectrum must be periodic over the free spectral range, as required by PICWAVE'S TDTW algorithm. This means that the fitting at the edges of free spectral range cannot in general be accurate.

Unlike the single Lorentzian model, the gain peak is maintained all the way down to transparency.

## 11.5 The Electro-Absorption Modulator Model

Like the Gain Model, the EA-Modulator model is based on a Lorentzian response, but here we concern ourselves with absorption instead of gain.

The absorption  $\alpha$  is calculated from the Lorentzian response:

$$\alpha = \alpha_{pk} \frac{(w/2)^2}{(\omega - \omega_0)^2 + (w/2)^2}$$

where  $\omega$  is the optical frequency,  $\alpha_{pk}$  is the peak absorption,  $\omega_{pk}$  is the frequency of the peak and  $w$  is the FWHM width of the peak.

In addition to the usual dependence of the Lorentzian peak (position, strength and width) on carrier density, the EA-Modulator model introduces also a voltage dependence of both the absorption peak strength and position. This voltage dependence can be define via the user-interface as either a cubic spline or polynomial function.

## 11.6 Kappa Calculator – theory

The Kappa Calculator built into PICWAVE is based on coupled mode theory<sup>10,11</sup> with a few improvements of our own.

First we define some terms:

A - forward guided wave amplitude.

**AA\*** - guided power flux in +z direction.

B - backward guided wave amplitude.

**BB\*** - guided power flux in -z direction.

DFB coupling coefficient definition: 
$$+\frac{dA}{dz} = \kappa B$$

### DFB Coupling Coefficients

TE 
$$\kappa = \frac{k_0^2}{2i\beta_m} \int \mathcal{E}_m^* A_s \mathcal{E}_m dx$$

TM 
$$\kappa = \frac{1}{2i\beta_m} \left[ -(\beta_m + s)\beta_m \int \mathcal{H}_m^* \frac{A_s(x)}{\varepsilon(x)} \mathcal{H}_m dx + \int \frac{d\mathcal{H}_m^*}{dx} \frac{A_s(x)}{\varepsilon^2(x)} \frac{d\mathcal{H}_m^*}{dx} dx \right]$$

$\beta_m$  - propagation constant of forward guided wave.

E and H are the electric and magnetic field profiles of the modes, normalised:

$$1 = \int \mathcal{E}_m^* \mathcal{E}_m dx \quad 1 = \int (\mathcal{H}_m^* \mathcal{H}_m / \varepsilon) \cdot dx$$

These are solved as solutions of the TE or TM wave equation to the unperturbed dielectric profile  $\bar{\varepsilon}(x)$  in eqn. (12) below.

The grating is represented by the Fourier series:

$$\varepsilon(x, z) = \bar{\varepsilon}(x) + \sum_s e^{isz} A_s(x)$$

Alternatively the reciprocal may be expanded:

$$v(x, z) = \frac{1}{\varepsilon(x, z)} = \bar{v}(x) + \sum_s e^{isz} B_s(x)$$

and  $A_s$  in eqns. (2) to (3) replaced by  $B_s \bar{\varepsilon}^{-2}$

<sup>10</sup> Yamamoto et al. JQE vol. 14, p. 245, 1978

<sup>11</sup> W. Streifer, JQE vol. 12, p. 74, 1976

## Variables and Expressions

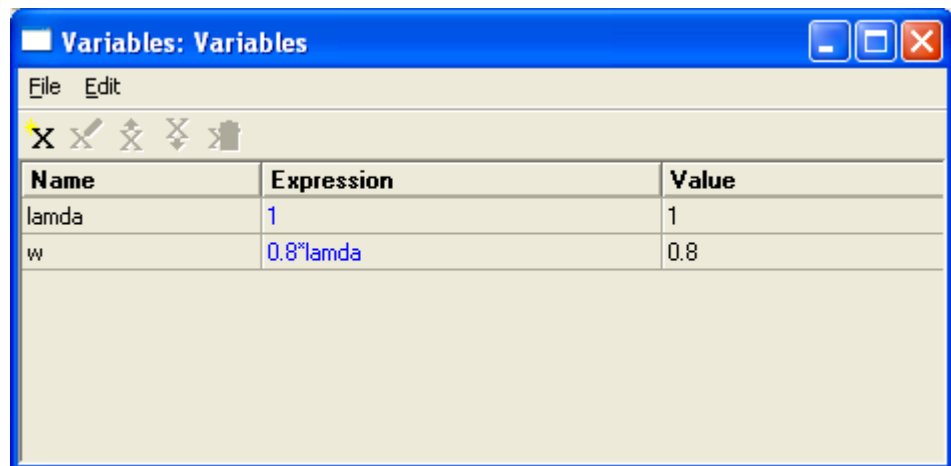
*Variables* are a powerful way of specifying values for parameters in PICWAVE.

Normally when you specify a parameter, such as the **Width** of a slice, you set it to a *value*, such as **1.5**. However, you can also specify a parameter by setting it to an expression, such as **5/2**. PICWAVE evaluates the expression **5/2 = 2.5**.

*Expressions* can contain various mathematical constants, operators and functions. For example, if you set the **Width** of the slice to **2\*cos(\_PI/6)**, PICWAVE evaluates the expression **2\*cos(\_PI/6) = 1.732**, and the slice will appear with a width of 1.732 $\mu\text{m}$ .

*Expressions* can also contain *variables*. If you set the **Width** of the slice to *w*, and set the *variable w* to the value **1.5**, then the slice again appears with width 1.5 $\mu\text{m}$

This may seem unnecessarily direct, but it can be extremely powerful. Suppose you want the width of the slice be exactly 0.8 times the wavelength. You have also four other slices in your structure and you want their width to be 0.8 times the wavelength. You can create a variable **lambda**, and set *w* to be **0.8\*lambda**. Then you can set the widths of the slices to *w*, so that they will automatically be updated when you change **lambda**.



## 12.1 Expressions

Any *parameter* in PICWAVE that would normally be set to a numerical value can instead be set to an expression. Such *parameters* are documented throughout this manual. There are far too many to list, but here are a few examples:

- the **length** of a waveguide section
- the **refractive index** of a layer
- the **Temperature** of a device
- the **Wavelength** .

In general any white text box can accept a variable or an expression.

*Expressions* can contain numbers, mathematical constants, operators and functions, and *variables*. Here are a few examples of expressions:

- **5/2**
- **2\*cos(\_PI/6)**
- **0.8\*z**
- **w\*sin(alpha).**

When you set a *parameter* to an expression in a property dialog, PICWAVE evaluates the expression using the current values of any variables to which it refers.

For example, suppose you are setting the variable **xsize** (referring to the x-size of a shape) of a shape to the expression **w\*sin(alpha)** in the *Device Properties dialog*. As you type the expression, the parameter will appear as follows:

xsize	w*sin(alpha)	0
-------	--------------	---

However, when you press **Enter**, or click on another property, PICWAVE evaluates the expression and the value appears after an equals sign after the expression. For example, if **w** is set to **12** and **alpha** is set to **\_PI/6**, the parameter will appear as follows:

xsize	w*sin(alpha)	6
-------	--------------	---

### 12.1.1 Numbers in Expressions

Numbers in *expressions* can be in either decimal or scientific notation:

- -5
- 1.732
- 3.9979e8
- 8.854e-12

Imaginary numbers can be used in *expressions* using *j* to represent the square root of -1:

- 1.732j
- 4 - 3j
- -1.38e6 + 9.05e-5j

### 12.1.2 Mathematical Constants in Expressions

You may use the following mathematical constant in *expressions*:

**\_PI**       $\pi$ , i.e. 3.1415926

### 12.1.3 Mathematical Operators in Expressions

You may use the following mathematical operators in *expressions*:

- +      add, e.g.  $1 + 3 = 4$
- subtract, e.g.  $7 - 5 = 2$
- \*      multiply, e.g.  $3 * 5 = 15$
- /      divide, e.g.  $12 / 3 = 4$
- \*\*     power, e.g.  $2 ** 8 = 2^8 = 256$
- (...)   brackets, e.g.  $(3 * 2) + 7 = 13$  but  $3 * (2 + 7) = 27$

## 12.1.4 Mathematical Functions in Expressions

You may use the following mathematical functions in *expressions*:

<i>sqrt</i>	square root of a number, e.g. $\text{sqrt}(3) = 1.732$
<i>pow</i>	one number to the power of another, e.g. $\text{pow}(2, 8) = 2^8 = 256$
<i>exp</i>	e to the power of the number, e.g. $\text{exp}(1) = e^1 = 2.7182818$
<i>ln</i>	natural logarithm of a number, e.g. $\text{ln}(2.7182818) = 1$
<i>sin</i>	sine of an angle in radians, e.g. $\text{sin}(1.5707963) = 1$
<i>cos</i>	cosine of an angle in radians, e.g. $\text{cos}(1.5707963) = 0$
<i>tan</i>	tangent of an angle in radians, e.g. $\text{tan}(0.78539816) = 1$
<i>asin</i>	inverse sine in radians, e.g. $\text{asin}(1) = 1.5707963$
<i>acos</i>	inverse cosine in radians, e.g. $\text{acos}(0) = 1.5707963$
<i>atan</i>	inverse tangent in radians, e.g. $\text{atan}(1) = 0.78539816$
<i>tanh</i>	hyperbolic sine, e.g. $\text{sinh}(1) = 1.1752012$
<i>cosh</i>	hyperbolic cosine, e.g. $\text{cosh}(1) = 1.5430806$
<i>tanh</i>	hyperbolic tangent, e.g. $\text{tanh}(1) = 0.76159416$
<i>sign</i>	sign of a real number, i.e. (-1 if number < 0) or (0 if number = 0) or (1 if number > 0), e.g. $\text{sign}(-4) = -1$
<i>abs</i>	absolute value of a number, e.g. $\text{abs}(-4) = 4$
<i>real</i>	real part of a complex number, e.g. $\text{real}(4 - 3j) = 4$
<i>imag</i>	imaginary part of a complex number, e.g. $\text{imag}(4 - 3j) = -3$
<i>arg</i>	argument in radians of a complex number, e.g. $\text{arg}(7j) = 1.5707963$
<i>max</i>	maximum of the specified values, e.g. $\text{max}(-7, 2, 5) = 5$
<i>min</i>	minimum of the specified values, e.g. $\text{min}(3.01e6, 7.48e5) = 7.48e5$
<i>lininterp</i>	$\text{lininterp}(x, a1, b1, a2, b2, a3, b3\dots)$ returns the value at x of a piecewise linear interpolation through the points (a1, b1), (a2, b2), (a3, b3)...., e.g. $\text{lininterp}(3, 2, 5, 4, 9) = 7$ , because the straight line through (2, 5) and (4, 9) crosses $x = 3$ at $y = 7$
<i>spline</i>	$\text{spline}(x, a1, b1, a2, b2, a3, b3\dots)$ returns the value at x of a spline through the points (a1, b1), (a2, b2), (a3, b3)...., e.g. $\text{spline}(3, 2, 5, 4, 9, 6, 1) = 7$ , because the spline through (2, 5) and (4, 9) crosses $x = 3$ at $y = 7$

## 12.1.5 Variables in Expressions

*Variables* should be defined before being included in *expressions*.

For example, if a parameter is set to the expression **x\*sin(alpha)**, and the variable **x** is set to 1.5 but the variable **alpha** is undefined, the following message appears in the Warnings window:

**Expression "x\*sin(alpha)" refers to undefined variable alpha**

For the purposes of evaluating the expression, the undefined variable **alpha** is taken to be zero, so the expression **x\*sin(alpha)** evaluates to zero.

## 12.2 Variables Nodes

*Variables* are defined in *variables nodes*.



You may create any number of *variables nodes* in a *project*.


Note that if the *Project Tree* is not visible, you can show it by clicking on the *Project tab* in the Main Window:



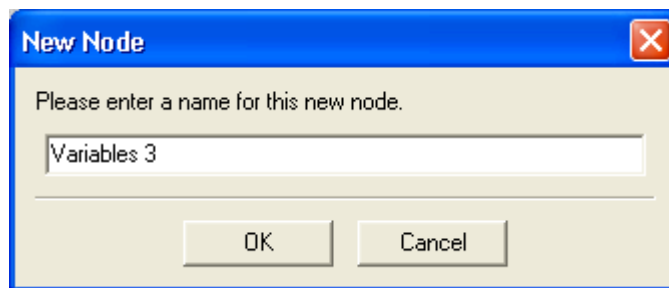
### 12.2.1 Creating a Variables Node

- To create a new *variables node* in a *project*:


click on the  *project* in the *Project Tree*, then click on the  Add Variables button

or right-click on the  *project* in the *Project Tree*, then select **Add/Variables**.

The *New Node dialog* will appear for you to enter the name of the new variables node:



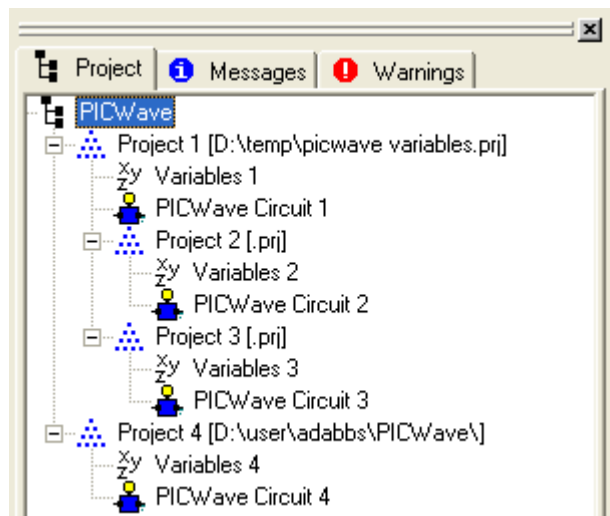
- Type a name for the *variables node*, then click .

The new *variables node* will be shown in the *Project Tree* with the variables symbol .

It should be noted that a *variables node* can be a child of a project node only. We can't have a variable node as a child of a *RWG Waveguide* for example.

## 12.3 Variables Scope

The *variables* defined in a *variables node* can be used in any *device* (or *RWG* etc) in the same *root project*. For example, consider the following project tree:



There are two *root projects* in this tree, Project 1 and Project 4.

The members (devices, RWGs, variable nodes) of a *root project* and all its child projects can use any of the *variables* defined in any of *variables nodes* in the *root project* or its child projects e.g. PICWave Circuit 1 could use a variable defined in the Variables 3 node, and PICWave Circuit 3 could a variable defined in the Variables 2 node, and so on.

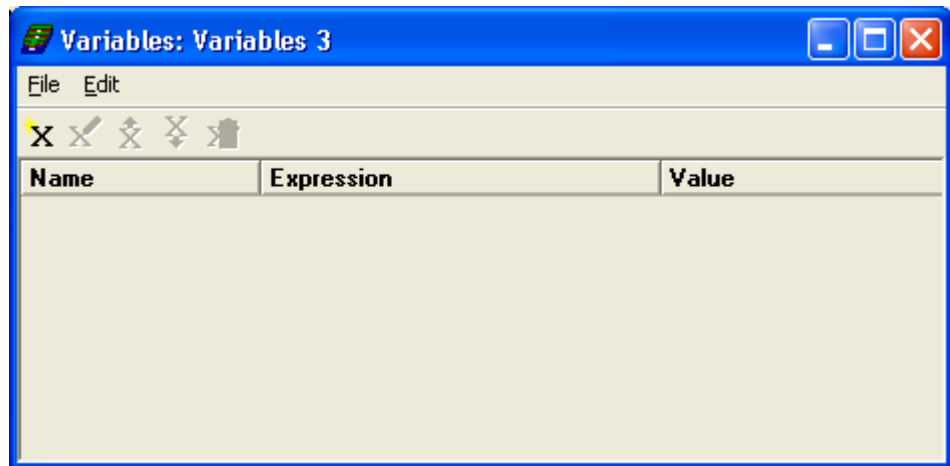
However, the members of a *root project* and its child projects cannot use *variables* defined in *variables nodes* belonging to *another root project* or its child projects e.g. PICWave Circuit 4 could not use a variable defined in the Variables 2 node.

In a similar sense, *variables* belonging to nodes belonging to the same *root project* cannot have the same name e.g. a variable in the Variables 1 node cannot have the same name as variable in the Variables 3 node; whereas variables belonging to different *root project* can have the same name.


When you copy a *device* (or any other type of member) from one *root project* to another, you should copy any *variables nodes* containing *variables* to which the device refers *before* copying the device.

## 12.4 Variables

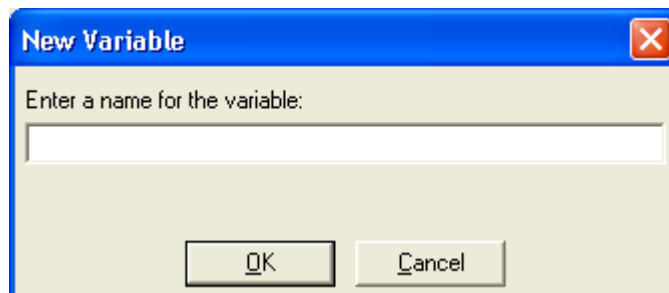
When you show a *variables node*, the *variables window* appears:



### 12.4.1 Creating Variables

- To create a new *variable*:
  - click on the  New variable button in the variables window
  - or* select **Edit/New Variable** in the variables window
  - or* click on the variables window to make it the front window, then press **Ctrl+N**.

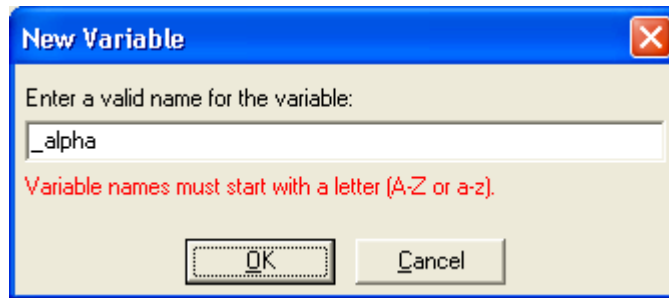
The *New Variable dialog* will appear:





➤ Type a name for the new variable, then click **OK**.

If the name you type is invalid, a warning will appear:



*Variable names* must meet the following criteria:

- must not contain spaces
- must start with a letter (A-Z or a-z)
- must contain only letters (A-Z or a-z), digits (0-9) and underscores (\_).

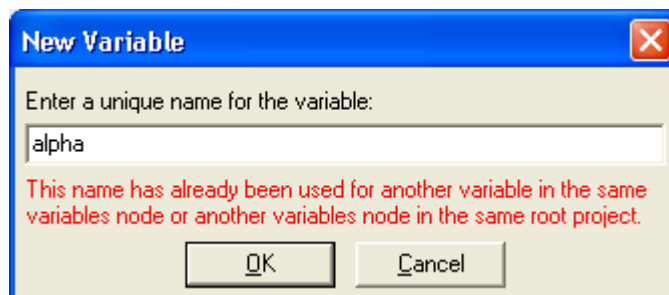
For example, the following variable names are valid:

- z
- alpha
- R\_0
- GridSpacing1Max

but the following variable names are invalid:

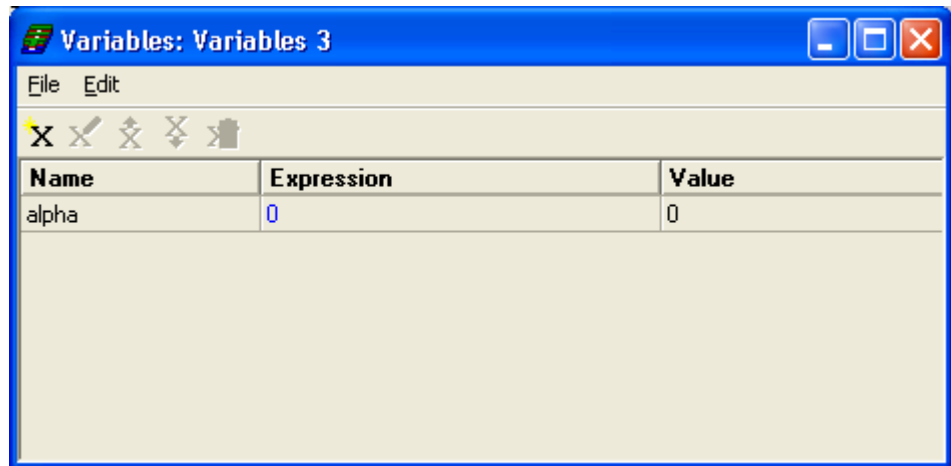
- double alpha
- 0\_R
- transmission%
- A\*

If the name you type has already been used for another variable, a warning will appear:



This warning will appear even if the name has been used for another variable in another variables node, if that variables node is in the same root project (see §12.3).

When you have entered a valid variable name, the new variable is created with an initial value of zero:



### 12.4.2 Setting Variables

- To set a *variable*, click in the **Expression** column in the variables window, then type a new *expression*.

*Expressions* can contain numbers, mathematical constants, operators and functions, and other *variables* (see §12.1).

Often, you will simply want to set a variable to a number such as **1.5** or **1.732j**.

Sometimes, you will want set a variable to an expression involving mathematical constants, operators and functions, such as **5/2** or **2\*cos(\_PI/6)**.

To use the full power of variables, you may want to set a variable to an expression involving other variables, such as **0.8\*z** or **z\*sin(alpha)**.

As you type the expression, the variable will appear as follows:

Name	Expression	Value
alpha	_PI/2	0

However, when you press **Enter**, or click elsewhere, PICWAVE evaluates the expression and the value appears in the **Value** column:

Name	Expression	Value
alpha	<u>_PI/2</u>	1.5707963

Whenever you set a variable, PICWAVE automatically updates any *parameters* or other *variables* that refer to it, and all the relevant devices will be updated to reflect the change.


For example, suppose you have variables defined as follows:

Name	Expression	Value
z	12	12
alpha	<u>_PI/2</u>	1.5707963
distance	<u>z*sin(alpha)</u>	12

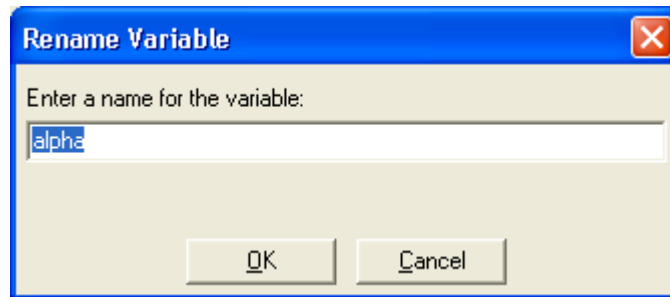
Suppose you then change the **alpha** from **\_PI/2** to **\_PI/6**. PICWAVE automatically updates not only the value of **alpha** but also the value of **distance**, since the variable **distance** refers to the variable **alpha**:


Name	Expression	Value
z	12	12
alpha	<u>_PI/6</u>	0.52359878
distance	<u>z*sin(alpha)</u>	6

### 12.4.3 Renaming Variables

- To rename a *variable*, click on it in the variables window to select it, then:
  - click on the  Rename variable button
  - or* select **Edit/Rename Variable**
  - or* press **Ctrl+R**.

The *Rename Variable dialog* will appear:





- Type a new name for the variable, then click .


If the name you type is invalid, or has already been used for another variable, a warning will appear (see §12.4.1).

### 12.4.4 Changing the Order of Variables

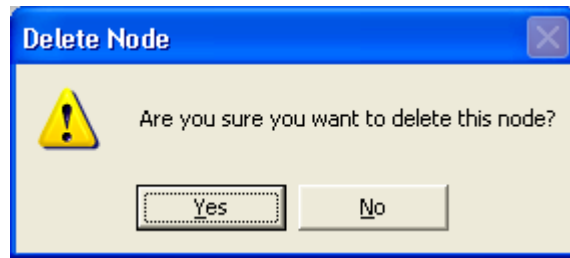
The order in which *variables* appear in the variables window does not affect their operation in any way. However, you can change the order in which variables appear for your own convenience.

- To raise a *variable* in the variables window, click on it to select it, then:
  - click on the  Raise variable button
  - or* select **Edit/Raise Variable**
  - or* press **Ctrl+A**.
- To lower a *variable* in the variables window, click on it to select it, then:
  - click on the  Lower variable button
  - or* select **Edit/Lower Variable**
  - or* press **Ctrl+L**.

### 12.4.5 Deleting Variables

- To delete a *variable*, click on it in the variables window to select it, then:
  - click on the  Delete variable button
  - or* select **Edit/Delete Variable**
  - or* press **Ctrl+D**.

You will be asked to confirm that you want to delete the variable:



- Click **Delete** to delete the variable.

You should not delete a variable to which any parameters or any other variables refer.

For example, if the variable **distance** is set to the expression **z\*sin(alpha)**, and you delete the variable **alpha**, the following message appears in the Warnings window:

**Variable distance = z\*sin(alpha) refers to undefined variable alpha**

For the purposes of evaluating the expression, the undefined variable **alpha** is taken to be zero, so the variable **distance** evaluates to zero.

## Command-Line and Client/Server Interface

PICWAVE has an integrated *Command-Line* interface and macro recording facility. The user can control all operations in PICWAVE by executing commands in the *Command-Line Window*, or running batch scripts stored in a file, generated manually or via the macro recording facility. This allows the user to set up and run a series of automated tasks without intermediate intervention.

PICWAVE also comes with a remote TCP/IP interface, which makes it possible to control PICWAVE from another program. The user can integrate PICWAVE functionality in his/her own programs to perform more complicated tasks not yet accessible via the graphical user interface. For the user's convenience, skeleton PYTHON and C++ client programs are provided in the distribution.

### 13.1 The Command-Line window

You can execute operations in PICWAVE by typing the appropriate commands in the *Command-Line Window*. This behaves like a DOS or UNIX terminal window.

- The bottom line of the window is the *prompt line*. It begins with the “>>>” character. Type your command here and press <ENTER> to execute it.
- The results of an executed command are displayed in the non-editable part of the window, just above the *prompt line*. You cannot write in this region, but you can copy from it using <CTRL-C>
- You can paste text on the clipboard on the prompt line using <CTRL-V>
- You can display a previous command on the *prompt line* by pressing <CTRL-UP>

The screenshot shows a window titled "PICWAVE Command-Line Window". The window contains a list of functions and their descriptions, followed by a prompt line. The prompt line is highlighted with a white box and labeled "prompt line". The command ">>>app.openproject" is entered on the prompt line.

```

exit    FUNCTION - (): exit application
exportcb FUNCTION - (): export local clipboard to system clipboard
findnode FUNCTION - ( path ): find subnode
getparent FUNCTION - (): get the parent
getwdir FUNCTION - (): get working directory (stored in wdir)
importcb FUNCTION - (): import local clipboard from system clipboard
numsubnodes INTEGER - number of subnodes
objtype  STRING - get the object type
openproject FUNCTION - ( filename[, nodename] ): open the specified project with the e
paste    FUNCTION - ( newname ): paste subnode from the clipboard
progversion STRING - program version
setwdir  FUNCTION - (dirpath): set working directory
showmain FUNCTION - (): show
subnodes LIST<PObject> - sub
wdir    STRING - current working ( or setwdir first)

>>>app.openproject
  
```

## 13.2 Using the Command-Line interface

Most operations executed via the graphical interface have an equivalent *Command-Line* operation. Indeed when executing a graphical operation, e.g. selecting */project/open* from the menu, the program actually generates the equivalent *Command-Line* instruction which it then sends to its own interpreter. The executed instruction therefore appears in the *Command-Line Window*.

For example, if you create a new project called “Project 1” via the *Project Tree*, the line `app.addsubnode(pdEditorProject, "Project 1")` appears in the *Command-Line*. The user could easily have executed this command directly by typing this line directly on the prompt line.

### 13.2.1 The command syntax

The best way to learn the command syntax for executing the above operations is by watching the output generated while using the GUI.

The command syntax has been deliberately kept as simple as possible. The user can perform the following operations:

1. Set PICWAVE’s *input variables* (e.g. waveguide widths, simulation wavelength, coupling coefficients etc..)
2. Execute PICWAVE’s *commands* (e.g. add a waveguide, rotate a join, run a simulation etc...).

You will notice that all variables and commands follow a syntax similar to C++:

- `rootname.obj1.obj2...variable=somevalue` sets the input parameter `rootname.obj1.obj2...variable` to the value *somevalue*.
- `rootname.obj1.obj2...command(par1,par2,...)` executes the command `rootname.obj1.obj2...command` with parameters *par1,par2,...*

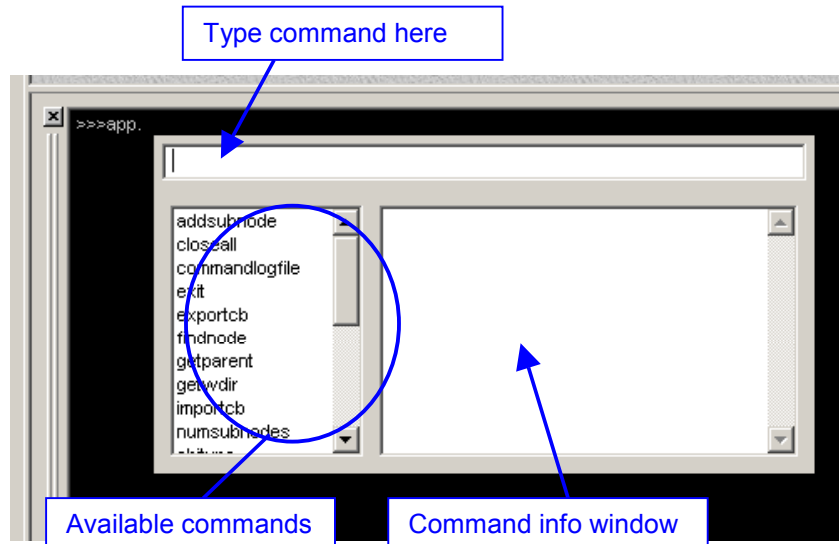
For example: we can generate a new *RWG Waveguide* by executing:

```
app.addsubnode(pdEditorProject, "Project 1")
```

As you can see, we have executed the command `addsubnode` belonging to the object `app`.

### 13.2.2 Discovering variables and functions

The *Command-Line Window* comes with a convenient command completion system, which avoids you having to repeatedly use the help command (see below). To use this feature, press the <TAB> at any stage while typing your command in the *Command-Line Window*. If possible the function or variable you are currently typing will be automatically completed, otherwise (usually if you press <TAB> after a “.”) the *command completion box* shown below will appear.



This displays all the functions and variables of the object whose name precedes the last “.”, and their corresponding info strings (the same as displayed using the `help` command). As you type in the input line of this box, the closest matching command in the available commands panel will be highlighted. An info string is displayed for the currently highlighted command. Press `<TAB>` again (or `<ESC>`) to dismiss the *command completion box*.

You can also explore the functions and variables accessible in the object `app` by typing

```
help app
```

This will give you the following output:

```
PDObject
Children:
  addsubnode      FUNCTION - ( blocktype,newname ): add a new subnode;
                  blocktype=pdEditorProject
  closeall        FUNCTION - (): close all projects
  commandlogfile  STRING
  exit            FUNCTION - (): exit application
  exportcb        FUNCTION - (): export local clipboard to system clipboard
  findnode        FUNCTION - ( path ): find subnode
  getparent       FUNCTION - (): get the parent
  getwdir         FUNCTION - (): get working directory (stored in wdir)
  importcb        FUNCTION - (): import local clipboard from system clipboard
  numsubnodes     FUNCTION - (): get number of subnodes
  objtype         STRING - get the object type
  openproject     FUNCTION - ( filename[, nodename] ): open the specified project with
                  the specified node name
  outputlogfile   STRING
  paste           FUNCTION - ( newname ): paste subnode from the clipboard
  pauseonerror    INTEGER
  setwdir         FUNCTION - (dirpath): set working directory
  showmain        FUNCTION - (): show main window
  stepbystep      INTEGER
  stoponerror     INTEGER
  subnodes        LIST<PDObject> - subnodes
  wdir            STRING - current working directory (call getwdir or setwdir first)
```

The “*children*” is the list of all functions and variables contained in PICWAVE for that particular object type. Each line gives the name of the child, the type, a short

description telling you what each “child” does and any parameters the functions might require. For example, to close all open project nodes, just type

```
app.closeall
```

Note that in the help string there was a “()”, which means that no parameters are required.

Much of what you’ll need in using the command line and writing scripts can be learnt by experiment, i.e. by a combination watching/recording (§13.2.4) the command line as you do various things through PICWAVE’s GUI, and/or using command completion box to see what functions/variables are available for each node object.

### 13.2.3 Accessing subnodes

The object `app` shown in the above example is the basic PICWAVE object, and it contains all the PICWAVE functions, such as opening projects, generating new nodes, etc. Once you have executed the command

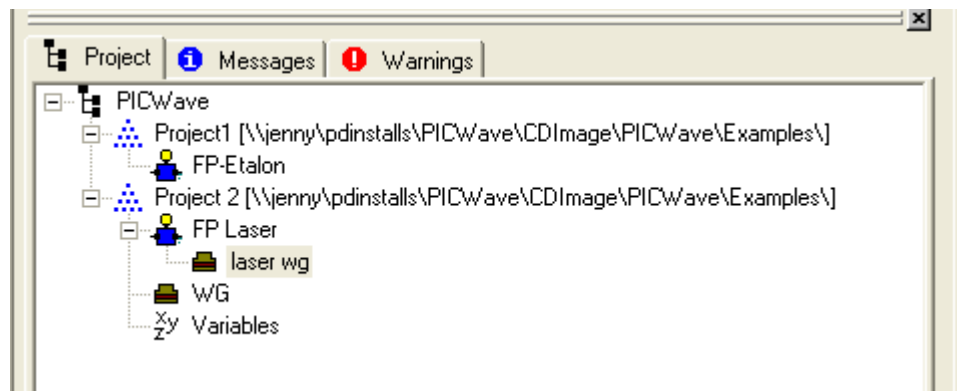
```
app.addsubnode(nodeType, test)
```

a new node named `test`, of type `nodeType`, will be appended to the subnode list `app.subnodes`. The only subnode that can be added to the `app` object is a project, so in this case `nodeType` must be ‘pdEditorProject’.

An RWG could then be added to the project node, for example, with the line

```
app.subnodes[1].addsubnode(rwg_wguide, "RWG Waveguide 1")
```

In very complex projects containing many nodes and subnodes, etc. accessing a node via the `app.subnodes` list can get quite tedious. For example, accessing the node “laser wg” selected below



has the *Command-Line* name:

```
app.subnodes[2].subnodes[1].subnodes[1].
```

Alternatively, you can use the function `app.findnode` to find this node more conveniently by specifying the node name. To do this you type:

```
Ref& a = app.findnode("/Project 2/FP Laser/laser wg")
```

As you can see, the function `app.findnode` accepts the *path name* of the node (the “/” symbol denotes the subnode) and returns a *reference* to the node, which you can assign to a *reference*. A *reference* is just a placeholder for a variable, so in the above example, the reference “a” refers to the subnode “an swg”, and can be used like any variable. E.g. you can use the help command:

```
help a etc.
```



The function `findnode` is available to all nodes, and can be used to easily find *subnodes* of a particular *node*. For example, the above RWG could be accessed by doing the following two operations:

```
Ref& a = app.findnode("/Project 2/FP Laser")
Ref& b = a.findnode("laser wg")
```

The first line returns a reference to the PICWave Circuit “FP Laser”, so to obtain the reference “b” to “laser wg”, only the *relative path* wrt “FP Laser” need be specified. Note how the conventional UNIX notation for directory is used here, so you can find any node starting from any other node by climbing up the directory tree using the “..” symbol. E.g.

```
Ref& c = a.findnode("../WG")
```

will set “c” to the node “WG”. You can specify an *absolute path* (i.e. wrt. the root) by starting the name with a “/” sign.

### 13.2.4 The log files

In the *Command-line Options* (accessible via */Options/Command-Line* on the Main Window’s menu bar) you can specify a *command log file* and an *output log file*.

If the *command log file* is set, then all commands sent to PICWAVE, via the GUI or directly via the *Command-Line Window*, will be recorded in this file. This essentially makes up the macro recording facility in PICWAVE: the resulting sequence of commands stored in the *command log file* can be re-executed at a later stage via the menu */Scripts/Run a script...* The user can of course make modifications as necessary to the file before he does this, which gives the user an extremely convenient way to generate scripts.

In the same way, if an output file is specified in the *output log file* textbox, all output will be written to this file. Note that **ONLY** the output from the PICWAVE calculations will be written to the output file, and not the echo of the command sent. If the user wishes, both the input and output can be written to one file this can be done by specifying the same filename in the two log file textboxes. However, the resulting file cannot be run as a script file.

## 13.3 Batch commands

You can execute multiple commands all at once by typing them on a the same line separated by semicolons:

```
command1;command2;command3;.....;commandN
```

### 13.3.1 An Example

The commands below construct a circuit, add a waveguide, two facets, connect them up, add an optical source and a monitor. You can enter these lines straight into the *Command-Line Window* (you can even include the comments)

```
app.addsubnode(pdEditorProject,"Project1")
Ref& fpn=app.findnode(Project1)
fpn.addsubnode(McDevice,"FP-Etalon")
fpn.subnodes[1].mcdevice.addwgsection(1,4,0)
fpn.subnodes[1].mcdevice.addfacet(1,3,0)
fpn.subnodes[1].mcdevice.addfacet(1,6,0)
fpn.subnodes[1].mcdevice.objects[2].reflcoeffte=0.3
fpn.subnodes[1].mcdevice.objects[3].reflcoeffte=0.3
fpn.subnodes[1].mcdevice.addlink(1,2,2,1,1)
```

```
fpn.subnodes[1].mcdevice.addlink(1,1,2,3,1)
fpn.subnodes[1].mcdevice.addoptosource(1,2,1)
fpn.subnodes[1].mcdevice.addmonitor(1,3,2)
fpn.subnodes[1].mcdevice.objects[7].enablespecmeas=1
```

You can now see what you can do with this object by typing:

```
help fpn.subnodes[1]
```

You will see that the circuit contains a mixture of simple variables (e.g. nodename), objects (e.g. "subnodes") and functions (e.g. "addsubnode"). Simple variables are of type INT, FLOAT or STRING. These variables can be altered and inspected.

For example, you can set the TE effective phase index of the waveguide:

```
fpn.subnodes[1].mcdevice.objects[1].effindexte=2.1
```

You can look at the new value just by typing:

```
fpn.subnodes[1].mcdevice.objects[1].effindexte
```

which will return "2.1" in the *Command-Line Window*.

If you now want to see what else is in the waveguide, you just type

```
help fpn.subnodes[1].mcdevice.objects[1]
```

As mentioned above, using the "help" command (or using the *command completion box*) as you go along you can easily discover and use all the variables and functions available in PICWAVE.

### 13.3.2 Copying commands from a text editor

You can copy/paste a list of commands from a text editor into the input panel. In this case, all the commands will be executed in one go. This implies in particular that references will be valid throughout the execution of these commands, even though the list of commands in the text editor might have contained carriage returns.

The most general form of a list of commands in a text file is:

```
command1 ;... . ;command2 //comment1
command3 ;... . ;command4 //comment2
```

etc...

### 13.3.3 Running a script file.

A script file is any text file containing a list of commands in the format described above. You can execute these commands by copying/pasting them into the *Command-Line Window*, but more conveniently you can run the script file directly selecting */Scripts/Run a script* from the menu button and selecting your script file.

Note you can also run PYTHON scripts (described in §13-9) via this command, although not directly via the *Command-Line Window*.

### 13.3.4 Managing scripts

You can install scripts so that they are directly accessible under the menu */Scripts* in the Main Window. You do this via the *Script Manager*, accessible via */Scripts/Manage scripts*. The script manager allows you to install/uninstall PICWAVE scripts residing on your hard disk.

## 13.4 Command-Line options

The *Command-Line* options (accessible via */Options/Command-Line* on the main menu) allow you to control the behaviour of PICWAVE when executing commands from the *Command-Line Window* or a script file.

If **show dialog box on error** is active, then if an error is encountered the program will pause and pop up the usual error message box. Otherwise the program will *not* pause and the error message will be printed on the output pane.

If **stop when error occurs** is active, then if an error occurs the program will terminate execution of the batch commands.

If **ask user to continue at each command** is activated, then the program will pause before each command is executed. This can be useful if the user wants to execute a script file step by step for debugging purposes, for example.

## 13.5 Syntax reference

Here we give a formal description of the *Command-Line* interface syntax.

### 13.5.1 Basics

Variables and functions are accessed by their variable name, which has the form:

`rootname.name1.name2...`

where `rootname` is one of the following:

- `app` - the fundamental application object containing all functions and properties available within the application.
- a *reference* (defined using the “Ref&” key word – see §13.2.3)
- A global object returned by a function (Defined using the “Set” key word – see §13.2.3)

### 13.5.2 Variables

Variables are of the following types:

- **Simple variables:** these are INTS, FLOATS and STRINGS.
- **Objects:** other structure, which can contain other objects or simple variables.
- **Lists:** one dimensional array containing objects or simple variables.
- **Matrix:** two dimensional array containing objects or simple variables.

### 13.5.3 References and global objects

A reference is a placeholder for a variable. You can set a reference to any variable in the application. You set a reference “a” to the variable `rootname.name1.name2` by typing:

```
Ref& a = rootname.name1.name2
```

Some functions also return references to internal objects. In this case you may assign the reference returned by such a function by typing

```
Ref& a = funcname
```

Some functions also return object copies (and not just references). In this case you may assign a *global object* to that returned by such a function by typing

```
Set a = funcname
```

Note the difference between a *global object* and a *reference*: any operation executed on a reference will be as if it is executed on the original object it refers to, while a global object is an entity in its own right.

You cannot assign a *global object* to a *reference* returned by a reference-returning function. In the same way, you cannot assign a *reference* to an *object* returned by a object-returning function.

#### 13.5.4 Commands

There are four types of commands:

**assignment:** just type “a=b”,

1. where “a” is a valid variable name and “b” is a valid value. **Note: assignments only work with simple variables, (int’s, floats, strings).**
2. **to display a value:** just type “a”, and press return. The value of **a** will appear in the output window. If “a” is a simple variable, then its value is returned, else the object type. If it is a list or an array, all values will be displayed, unless you specify a component. So for example: “name.matrix” will display the entire matrix, “name.matrix[1]” will display the first row, “name.matrix[1][2]” will display the value at column 1, row 2
3. **to display a value with a comment:** just type “a //my comment”. The comment will appear alongside the variable value.
4. **to display just a comment:** just type “//my comment”. The comment will appear in the output value.
5. **to display several values in one line:** just type “a //\nocr”. This will prevent PICWAVE from appending a carriage return to the output log.
6. **to execute a function:** just type “a(par1,par2,...)”. If there are no parameters just type “a( )” or “a”, where “a” is a valid function name.
7. **to create a reference:** just type “Ref& r=a”, where “a” is a valid variable name, or a function returning a reference to an object. Do not put a space before the “&”.
8. **to create a global object:** just type “Set r=a”, where “a” is a function returning object.
9. **to get information on a variable:** just type “help a” where “a” is a valid variable name. A complete description of the variable will be returned.

#### 13.5.5 Comments

Commands can be followed by C++-like comments [“//comment” syntax only, not “/\*comment\*/”], so long as they are followed by a new line (like in C++).

Comments will be ignored except when displaying a variable, or when typed alone, in which case they will be displayed in the output.

There is a special comment:

```
//\nocr
```

This is a reserved key word, and will tell PICWAVE not to append a carriage return in the log file when printing a variable. So the sequence of commands

```
objname.value1  
objname.value2
```

will append to the output file:  
value1  
value2  
while the sequence of commands  
objname.value1//\nocr  
objname.value2  
will append to the output file:  
value1 value2

## 13.6 Remote TCP/IP interface.

PICWAVE can be accessed remotely via TCP/IP. To do this, you must start picwave.exe with the **-pt** argument:

-pt <portNo>

<portNo> is the TCP/IP port number on which PICWAVE should communicate. PICWAVE will connect to the host machine through its domain name via port <portNo>.

Once started this way, an external client program or script will be able to communicate with PICWAVE on the specified port number - see below on how to write a client program.

### 13.6.1 Operation

When accessing the PICWAVE *Command-Line* remotely, all options that interrupt the flow of the program are disabled. In particular:

1. If a command produced an error, dialog boxes will not appear (thus interrupting the flow of the program) and the execution of batch commands will not be interrupted. PICWAVE will return the message "ERROR:<the error message that PICWAVE generated>". The user can then decide what action to take if an error occurred.
2. Option **ask user to continue at each command** is also disabled if sending a message remotely.

## 13.7 Scripting

In principle, you can write a client program or script in any computer language that can implement a TCP/IP connection, however we recommend using PYTHON, which we provide on the CD Image.

The python interface has been designed to be very simple, yet very powerful - most of the commands pass through one central function, which can accept arbitrary PYTHON expressions. It automatically returns any results from the PD application as appropriate types (strings, real, complex numbers, lists or 2D arrays thereof, etc).

As an alternative, a sample client program is also provided written in C++ together with the necessary supplementary source files (see §13-20).

### 13.7.1 Installing Python

The option to install PYTHON is given to you upon first running the setup.exe on the CD Image. If you chose not to install it at that time then you can obtain PYTHON either by:

- a) In the PythonInstall subdirectory of the Application CD, the windows installer has been supplied. To install PYTHON, run the file: python16.exe and follow the online instructions.
- b) Downloading the installer from [www.python.org/1.6](http://www.python.org/1.6).

We also suggest that you install the Scripting Environment PYTHONWIN.

- To do this, run the file win32all-133.exe.

In addition to installing the above programs, you must create *two* new *environment variables*.

- **PYTHONPATH** - This must point to the directory where all PYTHON modules distributed with the application exist.

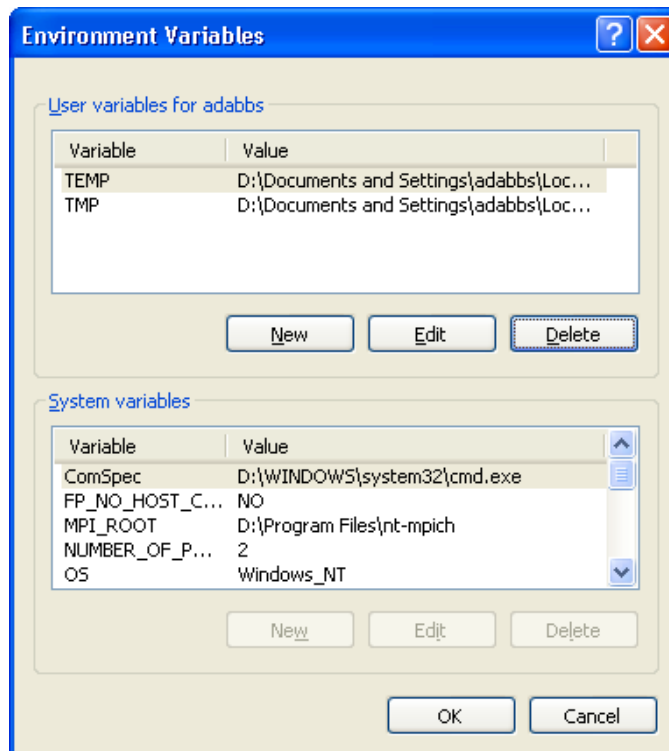
This allows PYTHON to find any extension modules without you having to explicitly reference their paths – see the PYTHON help files for more details.

- **PATH** – This must point to a directory where the Python executable is placed.

This allows you to run a script automatically from the Application Main Window.

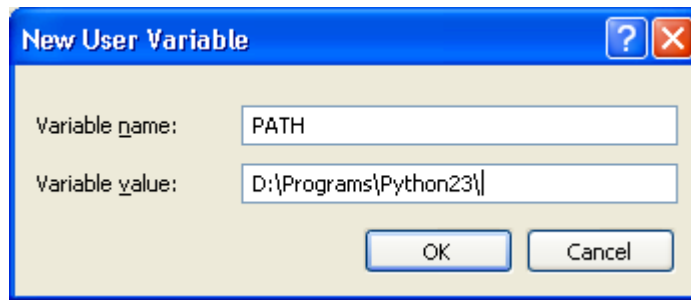
This is done as follows:

- Select Settings/ControlPanel/System/Advanced/Environment Variables. (MS Windows 2000/XP)

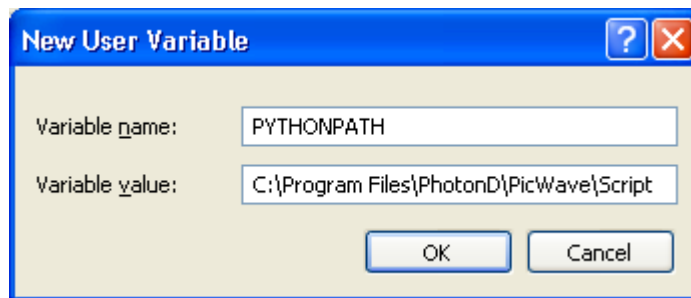


- Click **New**.
- In the Variable name box type PATH

- In the **Variable value** box type the path to where the python executable lies. E.g. C:\Programs\Python23



- Click **OK**.
- Click **New**, again.
- In the **Variable name** box type PYTHONPATH
- In the **Variable value** box type the path to where the python modules lie. E.g. C:\Program Files\PhotonD\PicWave\Scripts



- Click **OK**.

You should then have the following:

Variable	Value
PATH	D:\Programs\Python23\
PYTHONPATH	C:\Program Files\PhotonD\PicWave\Script

**Note.** The PC now has to be restarted.

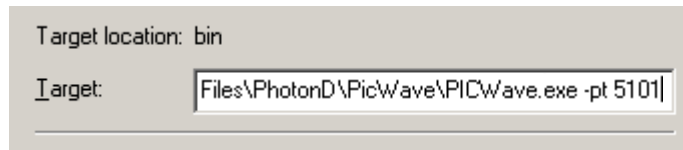
### 13.7.2 Example Python Script

Here we will create a script designed to work with the FPlaser device in the Tutorial Project with the settings as they are at the end of §2.2.5 (or a similar device); it performs the following operations:-

1. Connects to PICWAVE (already running with the project open)
2. Sets up a loop that varies the value of the TE reflection coefficients of the facets. For each value it runs a simulation, does an LI Fit, returns the threshold current and then prints the reflection coefficient value and the threshold current.

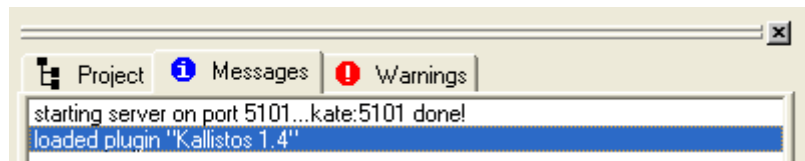
The first thing that needs to be done is to start PICWAVE ensuring that it is serving on port 5101. This is simply done:

- If PICWAVE is started via a shortcut key on the desktop, then the target (right click/properties...) should be set to be "...\PICWAVE.exe -pt 5101." See the figure below for reference.



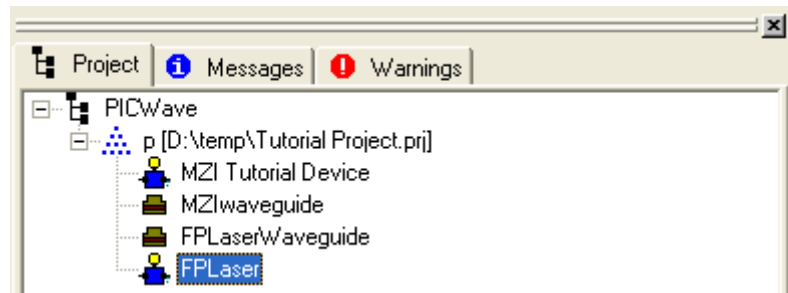
- Alternatively, if you start PICWAVE from a DOS box then use the command "PICWAVE.exe -pt 5101 localhost."

If PICWAVE has been successfully started on port 5101 you will see a message in the Messages tab of the Main Window confirming this.



Before we create and run the script, we need to open the relevant project.

- Open the project file "Tutorial Project.prj" that was created when doing the tutorial (or a similar device).



We will now write the Python Script.

- Using notepad or some other text editor, type the following into an empty text file:

```
from pdPythonLib import *
picw = pdApp()
picw.ConnectToApp()
```

The first line imports all the necessary functions and classes from the previously written module file (**pdPythonLib.py**) for running the script. The **pdPythonLib.py** module file contains one class called **pdApp**. This class contains all the functions needed to start, connect to and send messages to PICWAVE. A summary of these functions is given in §13.7.3.

The second line in the script:

```
picw = pdApp()
```

declares an instance of the **pdApp** class and stores the reference to this instance in the variable *picw*.

The third line connects to PICWAVE using one of the functions given in the table in §13.7.3 assuming that the program is already running and using the default TCP/IP port (which is port no. 5101).

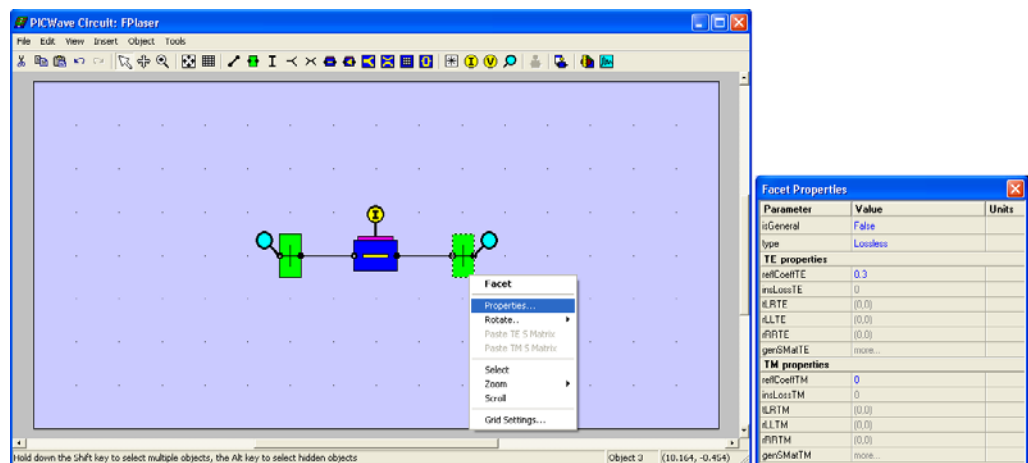


**Note.** You do not have to always connect to an already running version of the program. There is also command available to start PICWAVE. This is `StartApp()`. See the table in §13.7.3 for more details on this function.

What we wish to do now is to determine the commands to change the value of the reflection coefficients **reflCoeffTE** of the facets at either end of the active waveguide section, run the TDTW simulation and obtain the threshold. As mentioned in §13.2, most actions performed via the GUI will echo the relevant command in the Command-Line Window.

- Right-click on one of the two facets in the FPLaser device and select /Properties...
- Type 0.1 in the reflCoeffTE box.
- Close the Properties panel

This will change the reflection coefficient of the facet.



Notice that the commands

```
app.subnodes[1].subnodes[4].mcdevice.objects[2].startchange()
app.subnodes[1].subnodes[4].mcdevice.objects[2].reflcoeffte=0.1
app.subnodes[1].subnodes[4].mcdevice.objects[2].finishchange()
```

were generated in the *Command-Line*. (The numbering of nodes and objects may be different if your project/device was constructed in a different order etc). Repeat this for the other facet, to find its corresponding commands.

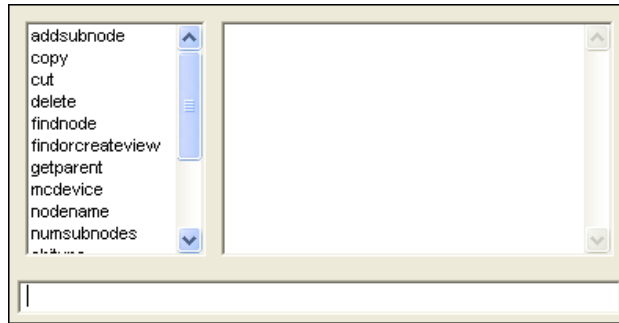
All we need now is the commands for extracting the data that we wish to collect. For this example we wish to collect the fitted threshold current value. These commands require the use of the TDTW Calculator. In the current version of PICWAVE, commands executed through the TDTW Calculator GUI do not echo in the command line, so the way forward is to use the *command completion box* (§13.2.2). to find the relevant commands.

- In the *Command-Line*, type:

```
app.subnodes[1].subnodes[4]
```

This is the location of the PICWAVE FPlaser device in the *Project Tree*.

- Press the <TAB> key. You should see the following:



- Scroll down to `tdcalculator`. Double-click on it so that it is appended to the end of your command, then append a further “.” Also so the command reads
 

```
app.subnodes[1].subnodes[4].tdcalculator.
```
- Press the <TAB> key again to bring up the *command completion box* for `tdcalculator` (the TDTW Calculator), this shows all its settings and commands
- Scroll down to `run` and double-click on it. The command is now complete and should read:

```
app.subnodes[1].subnodes[4].tdcalculator.run;
```

Press <ENTER> to run the TDTW simulation. The *Command-Line* will return zero when the simulation has finished.

We are now interested in extracting the threshold current of the laser device.

- Press <CTRL-UP> to bring up the previous command; delete the word `run` from the end and press <TAB>
- Scroll down to `rundata` and append it to the end of your command. Then append a further “.”
- Press <TAB> again to view the `rundata` (result analysis) options. Double-click `lifit`, append a further “.”, and Press <TAB> again to view the `lifit` (LI Fit) commands and parameters.
- We first need to run an LI Fit, so scroll down to `runfit` and append it to the end of your command, which should read:

```
app.subnodes[1].subnodes[4].tdcalculator.rundata.lifit.runfit
```

Press <ENTER> to run the LI Fit.

- To extract the fitted threshold current, bring up the previous command (<CTRL-UP>), delete `runfit` from the end, press <TAB>, and scroll down to `ith`. Append this to your command, so that it reads:

```
app.subnodes[1].subnodes[4].tdcalculator.rundata.lifit.ith
```

Press <ENTER>. The command line will return the threshold current value.

We now have all the commands we need, so let's finish writing a script to loop over a range of **reflCoeffTE** values and extract the corresponding threshold current values. This is done using the `Exec` method to send commands to the application.

➤ In your text editor add the following lines to your script:

```
for i in range (1, 10, 1):
    reffTE = i/10.0
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[2].startchange()")
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[2].reflcoeffte={reffTE}")
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[2].finishchange()")
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[3].startchange()")
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[3].reflcoeffte={reffTE}")
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[3].finishchange()")
    picw.Exec("app.subnodes[1].subnodes[4].tdcalculator.run")
    picw.Exec("app.subnodes[1].subnodes[4].tdcalculator.rundata.lifit.runfit")
    a = picw.Exec("app.subnodes[1].subnodes[4].tdcalculator.rundata.lifit.ith")
    print '%g %g' % (reffTE,a)
```

This loops over **reflCoeffTE** values 0.1 – 0.9 for both facets, and for each value extracts and prints the threshold current alongside the printed **reflCoeffTE** value.

### Note

- You need to include the indents for all commands in the “for loop”.
- Also note that the user can also conveniently embed any PYTHON expression in the string by enclosing it in {} as shown above. This embedding procedure is used in the third and sixth lines in the loop.
- Later within the loop, the following command:-  

```
a = picw.Exec("app.subnodes[1].subnodes[4].tdcalculator.rundata.lifit.ith")
```

assigns the number returned by *picw.Exec* into the PYTHON variable *a*.

➤ Then, outside the loop (i.e. with no indent), write:

```
raw_input(">>>>")
```

This will cause the script to wait for the user to press <ENTER> on the keyboard before continuing

➤ Next, write:

```
del picw
```

This will delete the connection and close the script.

You should now have the following:

```
from pdPythonLib import *
picw = pdApp()
picw.ConnectToApp()

for i in range (1, 10, 1):
    reffTE = i/10.0
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[2].startchange()")
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[2].reflcoeffte={reffTE}")
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[2].finishchange()")
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[3].startchange()")
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[3].reflcoeffte={reffTE}")
    picw.Exec("app.subnodes[1].subnodes[4].mcdevice.objects[3].finishchange()")
    picw.Exec("app.subnodes[1].subnodes[4].tdcalculator.run")
    picw.Exec("app.subnodes[1].subnodes[4].tdcalculator.rundata.lifit.runfit")
    a = picw.Exec("app.subnodes[1].subnodes[4].tdcalculator.rundata.lifit.ith")
    print '%g %g' % (reffTE,a)

raw_input(">>>>")
del picw
```

- Save the script as `IthScript.py` (for example) – note the PYTHON script extension `*.py`

We can now run the script as follows:

- From the PICWAVE Main Window select **Scripts/Run a script...**
- Select `IthScript.py`

The script may take a few minutes to run. When the script has finished you should see something like the following:

```

c:\ D:\Programs\Python23\python.exe
Attempting to connect to application on TCP/IP Port No. 5101
0.1 42.7175
0.2 27.1671
0.3 18.0905
0.4 14.5331
0.5 12.2028
0.6 10.489
0.7 9.57573
0.8 8.93276
0.9 8.99259
>>>

```

Notice that the script is waiting for action from the user (i.e. <RETURN>) before exiting.

This script is fairly simple and requires the user to have already set the simulation parameters (excepting the **reflCoeffTE** values). It can easily be modified to increase the degree of automation.

### 13.7.3 pdApp Class Member Summary

This section contains a summary of the four member functions of the **pdApp** class used to send messages to the Photon Design application.

#### StartApp

```
StartApp( pathname [ ,<portNo> ] )
```

This function starts and connects to the Photon Design application using the executable file that is given in *pathname*. Upon execution, the current directory becomes the directory where the Photon Design application is located. The optional parameter *portNo* specifies which port to use. The default value for *portNo* is 5101. If *portNo* is unavailable, an alternative port is automatically selected.

**Note.** In future versions, if you have a network license, it will be accessible as a computation server from anywhere on your LAN.

#### ConnectToApp

```
ConnectToApp( [hostname] [ ,portNo] )
```

This function connects to an already running the Photon Design application that is serving on *hostname* on the port given by *portNo*. Both parameters are optional with the

default *hostname* being “localhost” and the default *portNo* being 5101. If the port is unavailable, the function tries to connect with a Photon Design application that might be running on the next available port. If successful, the function returns an empty string. Upon failure, it returns a string containing an error message.

**Note.** In future versions, if you have a network license, it will be accessible as a computation server from anywhere on your LAN.

AddCmd

```
AddCmd( commStr [,varList] )
```

This command appends *commStr* to the current list of commands that are waiting to be sent to the Photon Design application (which are done so via a call to `Exec` - see below).

The variable, *commStr* can be any valid Photon Design application *Command-Line* expression, with the addition that any expression encapsulated in `{}` is first evaluated using the PYTHON interpreter (unless the `{}` are themselves encapsulated in “”).

For example the following lines:-

```
from pdPythonLib import *
f = pdApp()
f.ConnectToApp()
a = 1.103
f.AddCmd("MMI.lambda = {a}")
a = 1.55
f.Exec()
```

will set *MMI.lambda* equal to 1.103.

The optional parameter *varList* is a list of PYTHON expressions that are resolved before being embedded in *commStr*. The function will sequentially replace every `%` symbol encapsulated within `{}` with the next evaluated expression in the list. Complicated expressions can also be included in *varList*. For example:

```
from pdPythonLib import *
def myfunc:
    a = 0.103
    b = 1
    f = pdApp()
    f.ConnectToApp()
    f.AddCmd("MMI.lambda = {% + %}",[a,b])
```

This will first resolve *a*, then *b* and insert them into the string before resolving the whole expression within the `{}`.

*varList* is also useful since the `pdApp` class cannot resolve PYTHON expressions that involve local or module variables. For example:

```
from pdPythonLib import *
def myfunc():
    a = 1.103
    f = pdApp()
    f.ConnectToApp()
```

```
f.AddCmd("MMI.lambda = {a}")
```

will not convert {a} to 1.103 since the variable a goes out of scope upon execution of *AddCmd*. However:

```
from pdPythonLib import *
def myfunc:
    a = 1.103
    f = pdApp()
    f.ConnectToApp()
    f.AddCmd("MMI.lambda = {%}" , [a])
```

will work since a is now one of the members of *varList* which gets parsed within *AddCmd*.

Exec

```
Exec( commStr [,varList] )
```

This function sends commands to the Photon Design application. Following the evaluation of all expressions encapsulated in {}, *commStr* is appended to the current list of commands and then the whole list is sent to the Photon Design application. The function then receives the response from the Photon Design application. It returns this response as any of the following basic types: *a string, a floating point number, an integer or a complex number*. It can also return 1d arrays or 2d arrays of any of these basic types. The array indices have a one-to-one correspondence with those in the Photon Design application.

For example if, in a Photon Design application, one were to type the following command

```
project.MMI.cdev.smat.lr
```

into the Command Line Window, PICWAVE might return the following:-

```
lr[1][1]      (0.468,0.00745)
lr[1][2]      (0.216,0.0698)
lr[2][1]      (0.872,0.0003)
lr[2][2]      (0.134,0.00629)
```

If the equivalent PYTHON command is sent (assuming the user has an instance of PICWAVE referenced by the PYTHON variable *f*):-

```
a = f.Exec("project.MMI.cdev.smat.lr")
```

then *a* will be a **2d array** with the following values:-

```
a[1][1] = (0.468 + 0.00745j)
a[1][2] = (0.216 + 0.0698j)
a[2][1] = (0.872 + 0.0003j)
a[2][2] = (0.134 + 0.00629j)
```

Finally, mixed-type lists can also be returned. For example:

```
f.AddCmd("myfile_rwg.evlist.mlp.evstart")
```

```

f.AddCmd("myfile_rwg.evlist.mlp.evend")
f.AddCmd("myfile_rwg.evlist.svp.lambda")
f.AddCmd("myfile_rwg.evlist.mlp.nx")
[start,end,wavelength,nx,nslice]= f.Exec("myfile_rwg.nslice")

```

The last command in this list sets each of the PYTHON variables in the list to the returned value of the *AddCmd* line.

Note the optional parameter varList (described above) can also be used with the *Exec* command. This allows the calling of *Exec* from within local functions of user-defined modules and classes. See *AddCmd* for more details. The table below is a summary of these four commands.

Function Name	Parameters	What it Does	Return Value
StartApp	pathname: (No default), portNo: (default = 5101)	Starts the Photon Design application on the path given by pathname and connects to it using the port given by portNo. If this port is unavailable, it uses the next available port	An empty string if successful or a string containing an error message if it has failed at some stage.
ConnectToApp	Hostname: (default = "localhost"), portNo: (default = 5101)	Connects to an already running version of the Photon Design application given by the hostname and serving on the port given by portNo.	An empty string if successful or a string containing an error message if it has failed at some stage.
AddCmd	commStr: (No default) , varList: (default = [])	Firstly, this function evaluates any python expressions embedded in commStr. It then appends the command given in commStr to the current list of commands waiting to be executed (by a call to Exec)	None
Exec	commStr: (No default), varList: (default = [])	Firstly, this function evaluates any python expressions embedded in commStr. It then appends this to the current list of commands before sending this entire list to PICWAVE. Finally it receives the response from the Photon Design application and flushes the command list.	This function returns one the following basic types: <i>a complex number, an integer, a floating point number or a string</i> . However, it can also return <i>a 1d array or a 2d array</i> of any of those basic types or a <i>mixed-type list</i> containing any of these types.

#### 13.7.4 Python Client

*PdAppclient.py* can be used as a PYTHON interface to the scripting engine of all Photon Design (PD) applications (such as PICWAVE). This makes it possible to communicate with a running executable remotely via TCP/IP and use the power of the fully-fledged scripting language to write complex PICWAVE routines with relative ease.

To run this client program from the command prompt, simply open the Scripting Environment PYTHONWIN. Open and run the script PdAppclient.py giving as arguments:

- Port number - the TCP/IP port number
- Hostname – the name or IP address of the machine on which the PD application is running.

Alternatively, navigate to the directory that contains PdAppclient.py and type:

```
python PdAppclient.py <portNo> <hostname>
```

Another example of what can be achieved using PYTHON clients is given at the following website: [www.python.org/workshops/1997-10/proceedings/beazley.html](http://www.python.org/workshops/1997-10/proceedings/beazley.html).

Please note we are not able to provide technical support in debugging your client programs or scripts (unless technical support is required on the syntax or the *Command-Line* features of PICWAVE commands).

### 13.7.5 Using C++

As an alternative to PYTHON, we provide on the CD Image a c++ client.

There is additional documentation on writing client programs in the file `examples\SourceCode\pdAppClient.cpp`. If you wish to use another language other than C++ or PYTHON, we suggest you study the C++ example carefully.

In order to send messages to PICWAVE, you should use the *Client* class provided with the distribution (see `client.cpp`, `client.h`). The console skeleton program (`pdAppClient.cpp`) is provided showing you how to use it. To compile it, you will need all the other source files provided in the distribution. The client program is self-explanatory and provides a *Command-Line* interface for sending commands remotely to a running PICWAVE program. We suggest that you first familiarise yourself with this program by compiling it and running it. You can then use it as a starting point for writing your own more complex routines.

The Client class

The client class is quite simple to use: it only has 2 members that need explanation

```
bool Connect(int portNo, const char* serverName=NULL);  
const char* sendMessage(const char* command);
```

The first simply initialises the client and connects via TCP/IP port “portNo” to a running PICWAVE program situated on the computer with IP server name “serverName”.

The second sends the command “command” to PICWAVE. PICWAVE will then return the output once execution of the command (or command batch) is completed.

The return string consists of a series of lines of the following format:

```
RETVAL: output string
```

or

```
ERROR: error message
```

The first will be appended for every command sent that makes PICWAVE return a non-empty string. The return string is exactly the same as the one written to the output pane in the PICWAVE command window. The second will be appended whenever an error occurs.



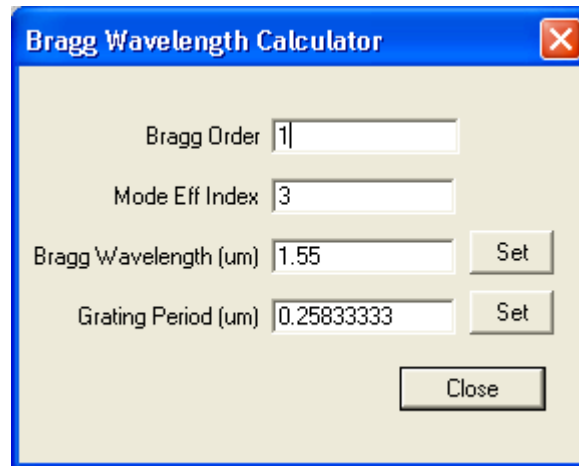
## Chapter

# 14

## Tools and Settings

### 14.1 The Bragg Wavelength Calculator

The *Bragg Wavelength Calculator*, shown in the figure below, allows you to specify the **Bragg Order** and **Mode Eff Index** and then calculate either the **Bragg Wavelength** for a given **Grating Period**, or the required **Grating Period** for a given **Bragg Wavelength**. Such parameters are needed when defining the grating coupling parameters, as detailed in §4.9.1.3, and when configuring a TDTW simulation.



#### Usage

- Type in the **Bragg Order** and **Mode Eff Index**.
- For the remaining parameters type in, according to your requirements, the known parameter. Then click the **Set** button next to the parameter you entered to calculate the unknown parameter.

The parameters are related according the equation:

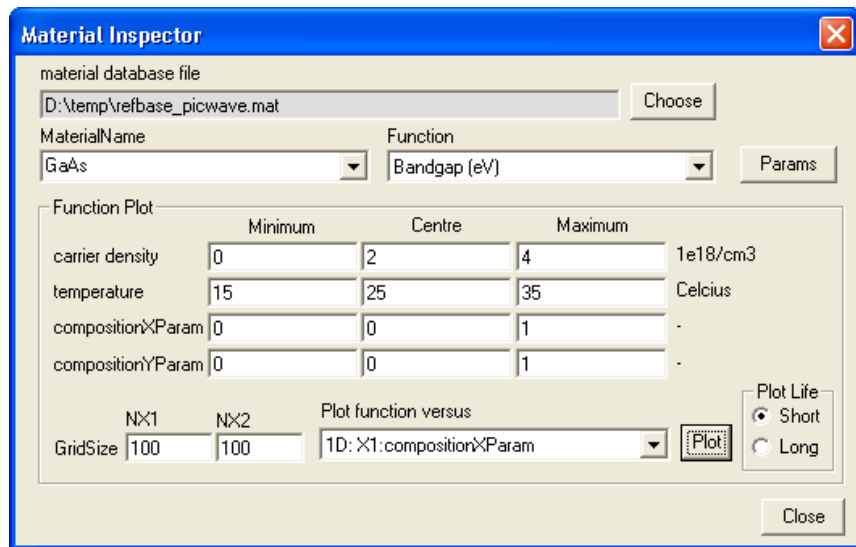
$$M(\lambda_b / n_{eff}) = 2\Lambda$$

where  $M$  is the Bragg order;  $n_{eff}$  the mode effective index;  $\lambda_b$  the Bragg wavelength;  $\Lambda$  and grating period.

### 14.2 The Materials Inspector

Various properties of the materials defined in the *Material Databases* can be inspected using the *Materials Inspector* (see figure below) accessible from the Main Window by */Tools/Materials Inspector*. This also allows you to plot (in 1D or 2D) the various property

functions (e.g. bandgap) against different variables, such as temperature and carrier density, for example.



#### Usage

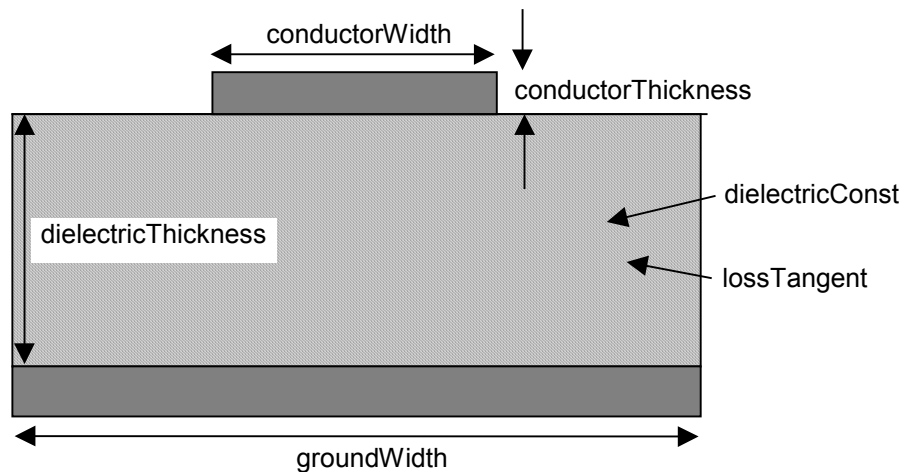
- Click **Choose** to browse for select the *Material Database* file you wish to inspect
- Then select the name of the material whose properties you wish to inspect from the **MaterialName** drop-down list, which contains all the materials defined in the chosen file. Click **Params** if you wish to view the properties defined by single numbers (e.g. dosVB, gain-eps), as opposed to polynomials and their related numbers.
- Then select the property function you wish to plot from the **Function** drop-down list
- In the **Function Plot** box, set the appropriate limits on the variables you wish to plot the property function against
- Finally, from the **Plot function verses** drop-down list, choose the particular variable(s) against which you wish to plot the property function and then click **Plot**.

Further settings are:

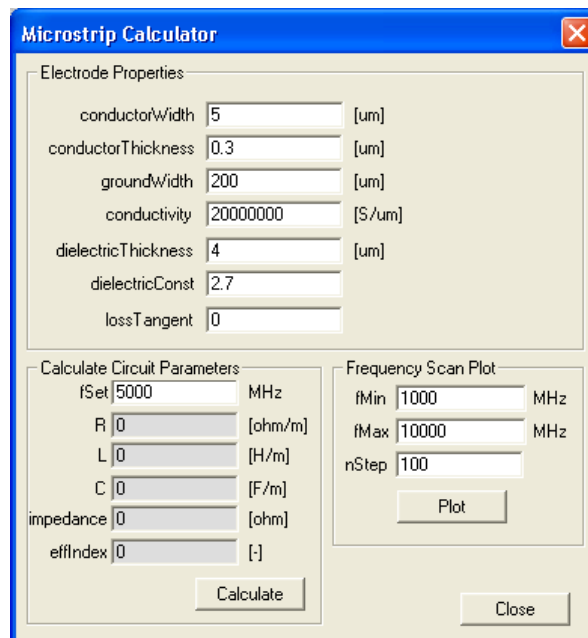
- Gridsize** the resolution of the plot
- plot life** if the plot life is short then it will disappear when the Materials Inspector panel is closed or another plot is chosen. If the plot life is long then the plot will last until it is closed down manually.

## 14.3 The Microstrip Calculator

The *Microstrip Calculator* is a convenient utility to calculate the circuit parameters needed for the *Travelling Wave Electrode Model*. This will calculate the equivalent circuit quantities for the following structure:




- Enter these parameters in the above figure into the **Electrode Properties** box of the *Microstrip Calculator* panel (shown below), along with the conductivity of the electrode. [The (dielectric) **lossTangent** [radians] can be set to non-zero to model microwave absorption by the semiconductor material].



To calculate the circuit parameters needed for the *Travelling Wave Electrode Model* (see §4.17.2) for a particular (microwave) electrical driving mode,

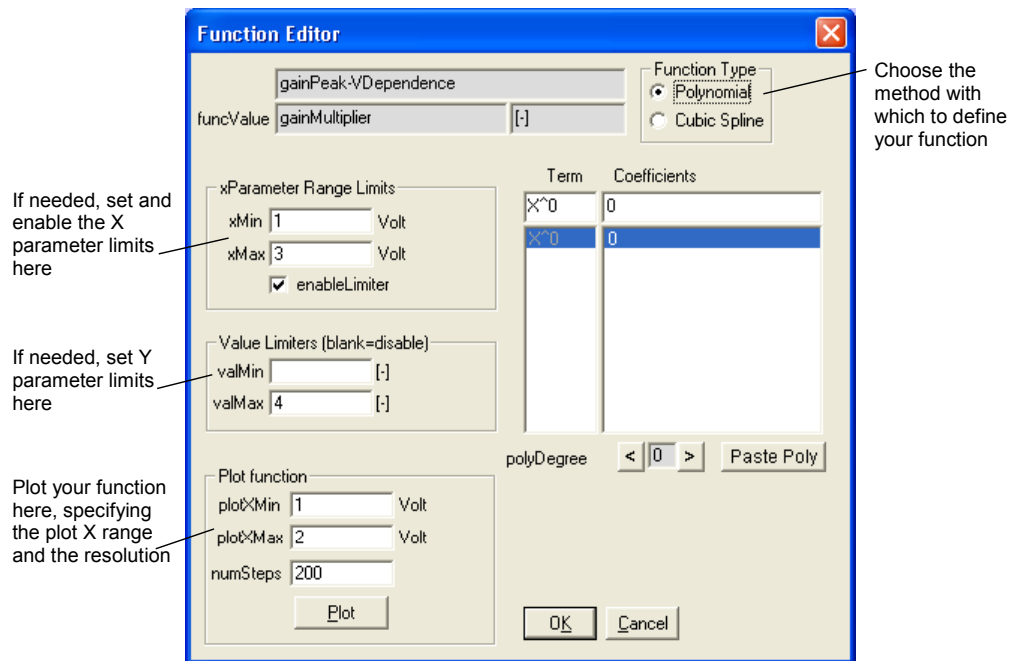
- Enter the (microwave) mode frequency in **fSet**, then click **Calculate** to calculate the effective resistance of the electrode per unit length  $R$ , the capacitance of the electrode per unit length  $C$ , the inductance of the electrode per unit length  $L$ , the impedance ( $= \sqrt{L/C}$ ) of the contact and the **effIndex** of the (microwave) mode along the electrode ( $= c/\sqrt{LC}$ ).

As you can only enter fixed values of these parameters into the *Travelling Wave Electrode Model* (which should be the values calculated at a frequency equal to the fundamental frequency of the electrical driving signal), it is useful to be able to check how accurate the model will be by plotting the same parameters as a function of frequency, in order to observe the extent to which they depart from their values at the fixed frequency for which they were calculated. All the frequency dependence is contained in the effective resistance  $R$  and the inductance  $L$ , to plot these against frequency,

- Set the frequency limits of the plot, **fMin** and **fMax**, and the number of frequency steps **nStep** in between, at which to plot the parameters.
- Then click 



## 14.4 The Function Editor

The **Electro absorption Model** for a waveguide section (§4.9.1.2) requires you to define functions for the **escapeRateFunc**, **gainPeakFunc** and **gainPeakPosnFunc** properties. Double-clicking on these properties opens up the *Function Editor* (see figure below), in which you can either define your own functions or fit experimental data with a cubic spline function.




The figure above indicates the settings common to both methods of defining functions. Instructions on the two methods of definition are given below.

### Polynomial Functions

- Select *Polynomial* from the **Function Type** box
- Use the   buttons to set the **polyDegree** (the highest power of X), the coefficients of the terms will appear in order of ascending powers of X
- Then select each coefficient line in turn and type in the coefficient values

You can also paste polynomials by copying text of the appropriate format to the clipboard (from anywhere) and clicking Paste Poly.

e.g. For the polynomial  $6 + 5x + 7x^2 + 8x^3 + 2x^4$ , the text format is 4 6 5 7 8 2. The first number gives the polynomial degree, the rest give the coefficients in order of ascending powers of x. The numbers must be separated, but can be separated by any amount of white space.

- Then click  to view the polynomial

## Cubic Spline Fitting

- Select *Cubic Spline* from the **Function Type** box
- Enter values the X and Y values of the data points, clicking **Add** each time to add the points to the list. The data points will be automatically arranged in order of increasing X if they are not already. Data points can be removed by selecting them from the list and clicking **Delete**.

You can also load data points from a (text) file by clicking **Load** and selecting the appropriate file. In the text of file, successive pairs of numbers form the (X,Y) data points; the individual numbers can be separated by any amount of white space, the data points can be written in any order.

e.g. Either of the following would load the data points (1,1) (2,4) (3,9)

1	1	1	
2	4	1	
3	9	2	4
		3	9

- Then click **Plot** to view the fit

## 14.5 Application settings

You can set a variety of default settings and control parameters for the program via the menu **/Options/Application...** These are stored to disk when quitting the program, and are read on startup. The parameters you can change are:

<b>defaultLambda</b>	the default wavelength assigned to new waveguides and used in TDTW simulations
<b>evalType</b>	the mode finder mode list can use either the propagation constant (beta) or effective index for labelling mode modes
<b>cpuPriority</b>	determines the CPU priority of the program.
<b>enableSounds</b>	make a sound when an operation is complete
<b>msgLevel</b>	(ADVANCED) determines how many messages are written to the message log while the program is calculating. A value of 1 is recommended for general use.
<b>diagnosticLevel</b>	(ADVANCED) determines how diagnostics resulting from a calculation should be displayed.

## 14.6 Device preferences

You can set the following default device preferences from any PICWave circuit device window by selecting **/Edit/Preferences**:

<b>defaultNeff</b>	default effective phase index for passive <i>effIndex</i> sections
<b>defaultNgroup</b>	default effective group index for passive <i>effIndex</i> sections
<b>defaultVgroupTol</b>	group velocity error tolerance used when computing zSteps of sections in a device

This section is designed to provide guidance in how best to achieve particular aims, using PICWAVE. The section also contains a number of hints and tricks you can use to perform your calculations more quickly or accurately. We will add sections to this Chapter as we develop our own expertise in the execution of new tasks.

## 15.1 Generating an LI Curve

LI curves must be generated by running a simulation in the time domain. Please see the Learning Guide §2 where you will find step by step instructions on generating an LI plot. You will have to decide how quickly to alter the drive current. In practise, for a “typical” laser with cavity length of a few hundred microns, you should find that you can generate a decent LI curve in 10ns to 20ns simulation time – try a few different times and compare the results. The most critical part is the turn-on at threshold which will have a significant delay – you may find that your LI curve will have a spike or other feature at turn on, due to this delay.

## 15.2 Modulation Response

You can of course run the *TDTW Calculator* with a sinusoid modulation over a range of frequencies and measure the optical response. However it is much faster to calculate the modulation characteristics from the impulse response. Here is the procedure:

1. Run your device until it has a stable output at the bias drive of interest.
2. Save the device state (from TDTW Control panel).
3. Create an electrical drive signal using an *Impulse* signal element; make the **length** of the element long enough to get the desired frequency resolution, e.g. a simulation of 10ns length will give a modulation resolution of 100MHz. Set the bias value of the signal element to be the same as that used to produce the *device state* file from which the device is being initialised. Have the impulse occur near to or at  $t = 0$ s.
4. From the TDTW Control panel, set the simulation to start from the *device state* you saved previously. You can also turn off spontaneous emission noise if the bias point is above threshold and you want to remove noise from the response spectrum.
5. Ensure that you have a monitor measuring the facet output of interest (i.e. with **enableSpecMeas** set to *On*). You do not need to add any facet instruments to this monitor for the purpose of calculating modulation responses, but you may wish to manually normalise the power modulation by the steady state (un-modulated) power, which can be recorded using a *Facet Power Instrument*.

6. Set the simulation time to the time length of the impulse signal element and run the simulation.
7. Go to the *Spectra Results* panel and set the following:
  - a) Select **Type** = *Intensity/RIN Spectrum*.
  - b) Then select the type of modulation response you wish to plot: received (elec) power (*INT/RIN elec power*), optical power (*INT/RIN - AM*) or frequency (*INT/RIN - FM*).
  - c) If you wish to normalise the modulation response by the spectrum of the modulation (current or voltage), select the electrical drive providing the modulation from the **Reference** drop-down list.
  - d) Set **tAlign** = *Start* (this will generate a spectrum starting with earliest available data, i.e. including the impulse at the beginning).
  - e) **FFT Size** to maximum
  - f) X Axis: **xmin**=0, **xmax**=10000MHz (say)
8. Click

The values entered in steps (d), (e) and (f) can be altered as desired (see §9.6.2)

## 15.3 Calculating a RIN Spectrum

You can calculate a RIN spectrum from the TDTW engine of PICWAVE. To create a good quality RIN spectrum you need to take a little care. Note that your RIN spectrum will itself always have noise in it, since the TDTW engine is a stochastic one and the RIN spectrum is determined by observing the noise over a short period of time. The longer you run your simulation the less noisy your result will be.

1. Run your device until it has a stable output at the bias drive of interest.
2. Save the device state (from TDTW Control panel).
3. Ensure you have an instrument measuring the facet output of interest and ensure its response time is set short enough.
4. Run a long calculation starting with the saved device state and with the same steady bias current i.e. no time-varying signals. We would recommend a minimum of 100ns simulation time.
5. Go to the Results/Spectra panel and set the following
  - a) Select type = **Intensity/RIN Spectrum**.
  - b) **FT Size** to maximum
  - c) X Axis: **xmin**=0, **xmax**=10000MHz
  - d) Set **FilterWidth**=5 (you can experiment with higher values, the setting will depend on how many time-steps you have in your simulation and what resolution you want in your spectrum)
6. Click

## 15.4 Generating an Eye Diagram

PICWAVE includes all the tools necessary to generate an eye diagram of your laser. This is a 2 stage process; a) setting up and running an appropriate simulation and b) generating an eye diagram from the resulting simulation.

**Run an appropriate simulation** You will need to drive your laser with a pseudo random bit sequence. To do this, add an NRZ *signal element* to the signal driving your laser- see §4.16. The element should have a length of at least 20x the clock period to give a good eye diagram.

**Generating an eye diagram** Open the TDTW Calculator Results Panel and click on the Oscilloscope Tab (see §9.7) Select the appropriate instrument - typically the facet output power. The Oscilloscope will try to detect the NRZ signal automatically but if it fails you may have to set a clock period manually by setting **Sync Trigger** to *Manual Period*. That's all there is to it!. You may want to adjust the plot a little by setting the **sampleMinT**, **sampleMaxT**, **graphMinX**, **graphMaxX** values. You may also want to adjust the position of the eye left or right by altering the **delay** parameter.



SciGraph is the graph plotting, visualisation sub-system used within Photon Design applications. Almost all of the graphs and plots you see in the application are produced by SciGraph. SciGraph also provides a variety of facilities for you to add annotations to your graph, change axis scaling, plot and export the graph.

SciGraph plots can either be *static* or *chart* plots. You cannot change the axes etc on a *static* plot. In contrast, a *chart* plot contains a copy of all the original plot data so you can alter axes, choose data sets, titles, plot type etc and re-plot everything with your chosen settings.

You can change these settings via **/Options/SciGraph** from the main window. Your changes will be saved when you exit the program.

## 16.1 Setup

SciGraph's default settings are set from the main menu **/Options/SciGraph**. You can leave most of these parameters as they are. The ones you might want to change are:

<b>Colours</b>	you can set the background colour for the SciGraph windows and the colour of the graph frame here. You should choose light or pastel colours as graphs are generally plotted in dark colours. Occasionally the program will over-ride (ignore) these colours and use its own scheme.
<b>windowx, windowy</b>	these define the default size of the plot window. You might prefer a larger or smaller size.
<b>Default font</b>	choose the font you want plots to use.
<b>Charsize</b>	you can change the size of the plot labels, axis annotations etc by altering this value.
<b>Outticks</b>	if you prefer your plots to have tick marks pointing outwards then check this box.

## 16.2 Static Plot

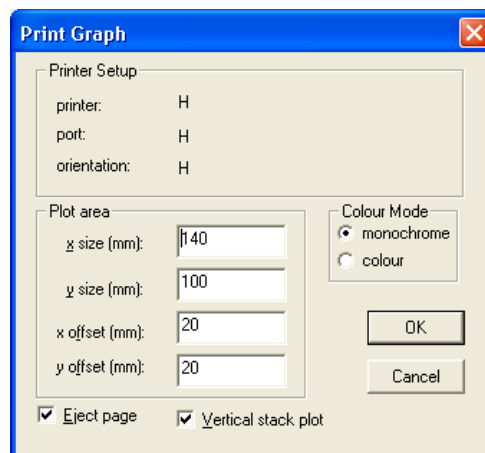
Here are all the menu options of the *static* plot. These are common to all SciGraph plots. Many of these options can be also be obtained by right-clicking on the graph.

<b>/File/Print</b>	opens a dialog box for printing your graph - see below.
<b>/File/Printer setup</b>	opens a dialog box to select the printer and set up various printer options.

<code>/Add/Text</code>	adds a line of text using the current Options. To add a subscript use the ‘_’ character and to add a superscript, use the ‘^’ character. Curly brackets denote extended sup/superscripts, e.g. “A_{loss}^{-2}” gives $A_{\text{loss}}^{-2}$ . Greek and other special characters can be added with reference to Table 16-1 at the end of the chapter.
<code>/Add/Line</code>	adds a line. Move the cursor to the beginning of the line, press the left button, drag the mouse to the end of the line and release the button.
<code>/Add/Arrow</code>	adds a line and adds an arrow head at the end point. The head size is set using your current font size.
<code>/Add/Undo</code>	undoes the most recent addition.
<code>/Options/Draw</code>	You can set the colour of future additions as well as the text font and height. The font size assumes a graph window 300mm high. There are 4 vector text fonts available - experiment to see what you prefer. Changes are saved to the registry when the program closes.
<code>/Export/Image</code>	brings up the Export Image dialog box (Windows Platform). You have a choice of two bitmapped formats - .bmp and .pcx, plus the Windows native vector .wmf format. The image can be either saved to file or put directly in the Windows clipboard (except .pcx format - file only).

### 16.2.1 Printing

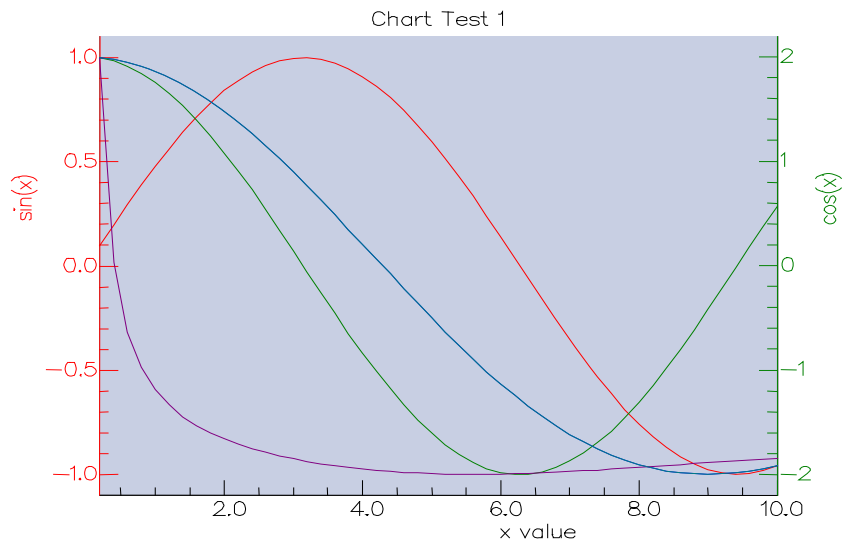
Selecting `/File/Print` brings up the dialog box shown in the figure below. You can set the size of the plot and its offset from the top-left corner of the page (all in mm).



If you want to print just one plot on this page then check the **eject page** option. If you leave it unchecked the page will not be printed until you print a plot where it is. In this manner you can get as many plots on one page as you want, by setting the x and y-offsets differently for each plot. For a more automatic multi-plot, check the **vertical stack plot** option - each succeeding plot will be automatically placed below the previous one, with **y-offset** now being the margin between plots. Such multiple plots may not work on all systems - you should experiment.

## 16.3 The XY Chart Plot

These are produced by such things as the General Scanners and provide additional functionality and plot data of the form  $y=f(x)$ . The figure below shows an example of a SciGraph chart plot. There is one additional menu option - /Options/Axes. This brings up a box for configuring the *chart*. A *chart* can have one or more data sets - you will see a list of each data set labelled Y1, Y2, Y3, Y4. Select the ones you want. Sometimes you will also be given the option of two different x-axis values X1, X2. Some data is saved as complex numbers and the module will plot complex data as amplitude and phase graphs. You cannot plot only the amplitude of a complex data set. On this box, you can also set the title for the graph and by each dataset, there are buttons marked edit. Press this to display the edit dataset box which is used to configure the data set.




Here are the list of options on the Edit Scale box:

- Position** select where the axis should be drawn - either on the left or right for a y axis or bottom/top for the x axis. Select *none* if you do not want a scale to be drawn for this data set.
- Type** select either linear or logarithmic scaling. If logarithmic scaling is used ensure that you do not have a zero or negative scale limit.
- tick style** you can elect to draw ticks along the axis and to add a grid.
- Minimum** set the scale minimum. If you leave this blank then the chart will set it automatically.
- Maximum** set the scale maximum. If you leave this blank then the chart will set it automatically.
- Annotations** the chart will use approximately this number of annotations along the length of the axis.
- Match scale** you may use this to draw the data set using the scale limits of another data set. If you use this then you may wish to set **position** to *none* to prevent the data set drawing its own scale.
- Mark style** select the style of marker for the data set.

- line style**            select the line style for the data set.
- plot style**            instruct the chart either to plot a line between data points, or to plot a marker at each data point, or both/neither.
- Label**                the axis label.

The additional menu options to export the graph are the following:

- /Export/Chart to file**        export a XY *SciGraph* chart to a .CHT file. You can re-plot a chart saved to a .CHT file by clicking on the  Open a SciGraph chart file button on the main window.
- /Export/Chart to clipboard**    export the contents of the .CHT file directly into the clipboard. You can and then paste it into a text editor to view the data as text.
- /Export/Data to clipboard**    export the data only (without the .CHT formatting) to the clipboard.

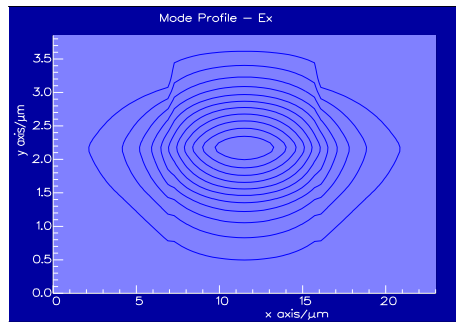
### 16.3.1 Exporting to Microsoft Excel

The function **/Export/Data to clipboard** is particularly useful if you wish to post process the data from different plots under Microsoft Excel since you only have to format it once. Once you have exported to your clipboard, paste your data into an Excel worksheet and select it. Then format your data by selecting **/Data/Text to columns...** in Excel and choose *spaces as delimiters*. If you repeat the operation with another plot the formatting will be automatic.

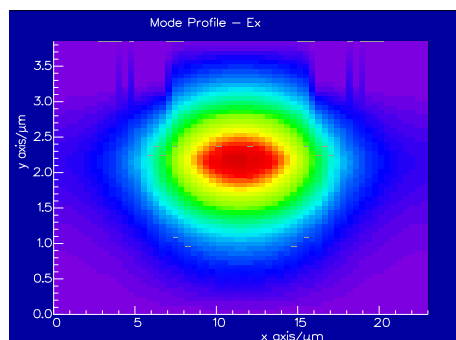
## 16.4 The 3D Chart Plot

This is used to plot data of the form  $z=f(x,y)$ . You can plot the data as one or combinations of the following plot types:

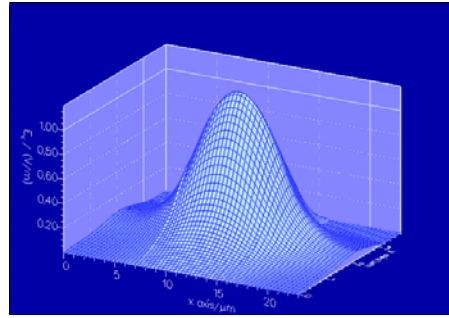
- **Contour Plot**



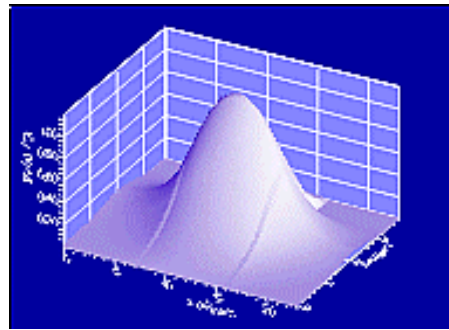
- **Colour Map**



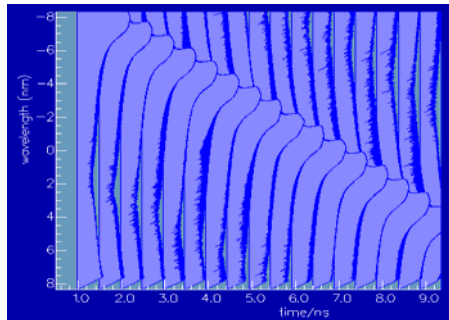
- 3d Mesh Projection



- 3d Gouraud Plot



- Slice Plot (*useful for noisy spectra*)



The 3D plot has three additional items on the Options menu from which you can set the viewpoint angle, dimensions of the 3D box, axis labels etc:

**/Options/Display Options** Selecting this activates the Display Options dialog box (*shown in figure 10-4*). From here you can choose which combination of plot types to use and various different parameters which are specific to each plot type.

**/Options/Axes** use this to change the axis labels, or scaling. You can also choose log scaling - so long as the data doesn't go through zero!

**/Options/3d Viewpoint** use this to alter the viewpoint of the mesh or the Gouraud plot, change the 3d box size, etc. Use the Preview Update button to see a wire frame preview of your settings. The **mesh skip** allows you to reduce the number of mesh lines. A value of 2 plots every other line etc. You can flip the whole plot up side down, back to front etc using the **invert axis** feature.

You will notice that when moving the mouse cursor over a Gouraud or a mesh plot, the cursor changes so it looks like:

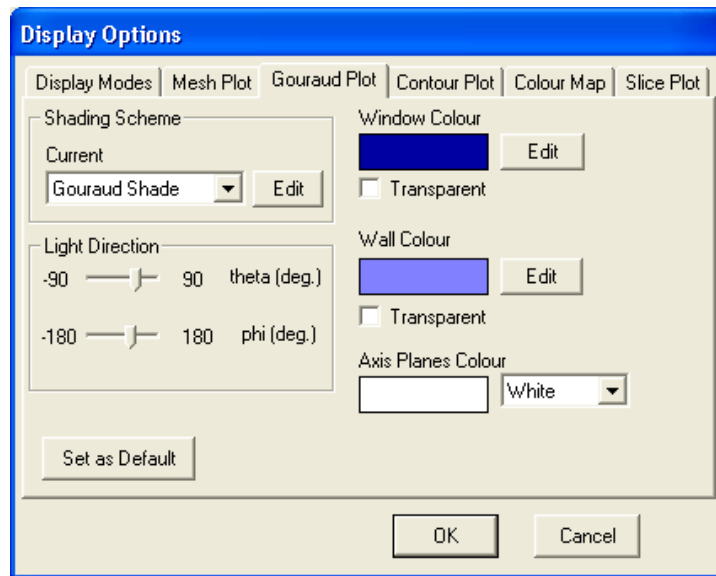


By pressing the left mouse button and ‘dragging’ this cursor, you can rotate the plot about the x and y axes. Furthermore, by holding the **SHIFT** key, the cursor is transformed to:



This time when you click and drag, rotation is performed about the z axis.

Using the display options dialog box, shown below, we can also edit the shading schemes used by the Gouraud plot and the colour plot. Clicking the relevant **Edit** button displays the Edit Scheme box. This enables you to modify the colour scheme to your liking. A colour map maps the normalised altitude “h”, of your data, which varies from 0 to 1, to a range of colours. Firstly the path goes through a “distortion”;  $f(x,y)=h(x,y)^d$ , where d is the distortion parameter. The actual colour is then determined using a linear interpolation between the *Colour Path Points* shown. For example if you want low levels to stand out more, you can reduce the “distortion” parameter below 1.0. Alternatively, you could introduce a new path point with a low f-value but a bright colour. You can have up to 5 path points. Note that only changes to the last two schemes, initially named User1, User2, will be saved when the program shuts down.

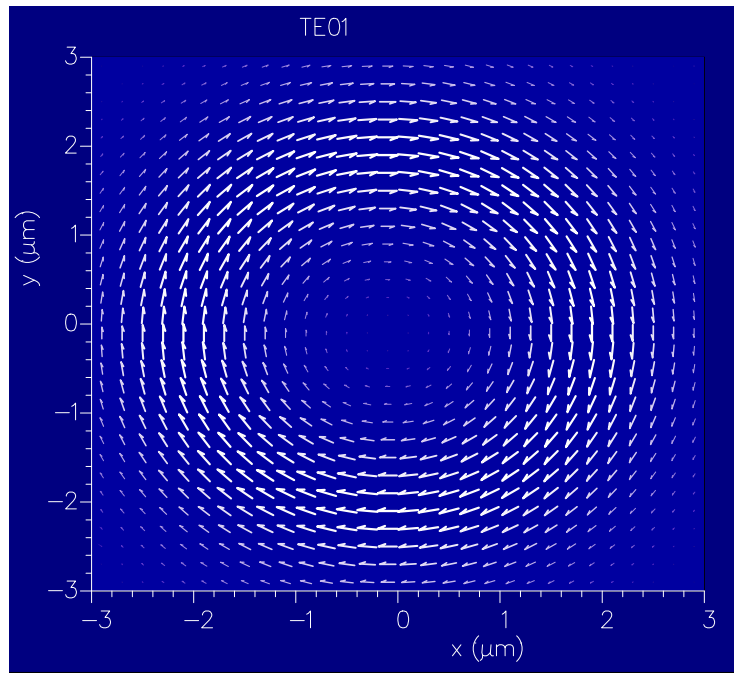


## 16.5 The Vector Field Plot

This is used to graphically represent the vector field  $\mathbf{F}(x,y) = f_x(x,y)\mathbf{i} + f_y \mathbf{j}$  where quantities given in bold type are vectors with  $\mathbf{i}$  being a unit vector parallel to the x-axis and  $\mathbf{j}$  being a unit vector parallel to the y-axis. The figure below shows an example.

There are two extra menu options that can be chosen within plots of this type. `/options/axes` allows you to change the axes minimum, maximum and scaling values. `/options/Display Options` activates the Display Options dialog box which contains the colour schemes, line thickness and other options which are used in the plot.

*NOTE: the availability of this plot will depend on the application.*



**Table 16-1 SciGraph greek and special characters**

Sequence	displayed as	name	sequence	displayed as	name
\a	$\alpha$	alpha	\A	A	
\b	$\beta$	beta	\B	B	
\c	$\chi$	chi	\C	X	Chi
\d	$\delta$	delta	\D	$\Delta$	Delta
\e	$\epsilon$	epsilon	\E	E	Epsilon
\f	$\phi$	phi	\F	$\Phi$	Phi
\g	$\gamma$	gamma	\G	$\Gamma$	Gamma
\h	$\eta$	eta	\H	H	Eta
\i	$\iota$	iota	\I	$\int$	integral
\j			\J		
\k	$\kappa$	kappa	\K	K	Kappa
\l	$\lambda$	lambda	\L	$\Lambda$	Lambda
\m	$\mu$	mu	\M	M	Mu
\n	$\nu$	nu	\N	N	Nu
\o	$\omega$	omega	\O	$\Omega$	Omega
\p	$\pi$	pi	\P	$\Pi$	Pi
\q	$\partial$	partial	\Q		
\r	$\rho$	rho	\R	P	Rho
\s	$\sigma$	sigma	\S	$\Sigma$	Sigma
\t	$\tau$	tao	\T	T	Tao
\u	$\upsilon$	upsilon	\U	Y	Upsilon
\v			\V		
\w	$\vartheta$	theta	\W	$\Theta$	Theta
\x	$\xi$	xi	\X	$\Xi$	Xi
\y	$\psi$	psi	\Y	$\Psi$	Psi
\z	$\zeta$	zeta	\Z	Z	Zeta



# Chapter 17

## Appendix

### 17.1 Validity tests

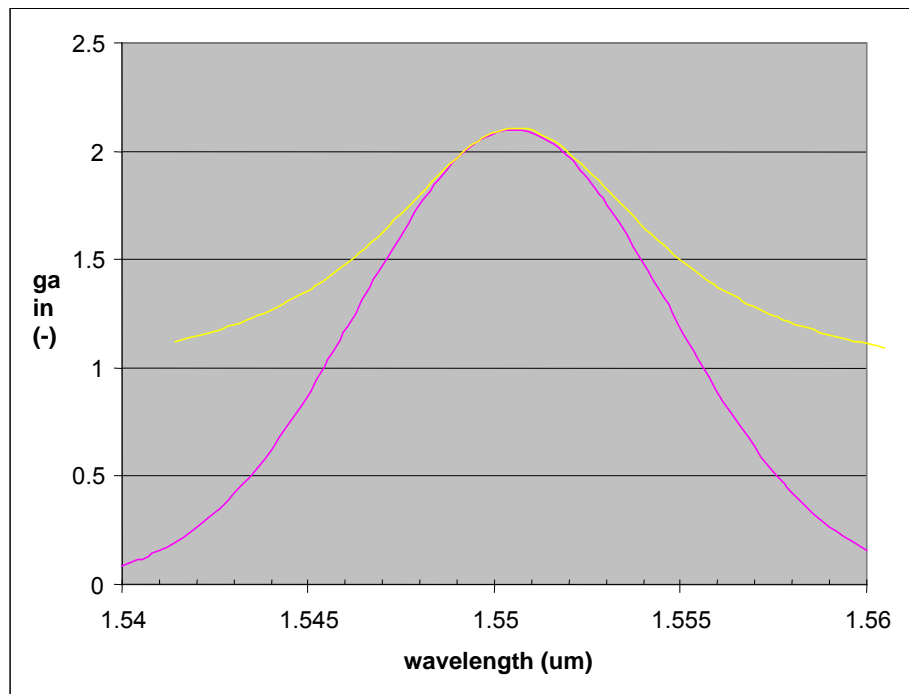
Test1 Mode Gain and the Gain Model

The wavelength dependence of the gain model defines the gain peak and the curvature:

$$g(\lambda) = g_0(Ne) + g_2(\lambda - \lambda_0)^2$$

The gain peak wavelength  $\lambda_0$  and the curvature constant  $g_2$  are defined in the Material Database. A simple test of the gain curvature is performed using the data:

ValidationTests.prj/SimpleSection.prj/GainCurveTest.



**Figure: Gain Curves** Top line is time domain result, lower line is plotted from Material Inspector results, taking section length and confinement factor into account.

This is a section of active waveguide with no grating, no facet reflections, so that it can be used as a TWA. The diagram below shows the resulting gain of the device, calculated with the time-domain model, by scanning the wavelength over a time of 10ns, and keeping the carrier density fairly constant. Noise sources have been turned off to produce a cleaner response.

Other parameters:

Cavity length: 160um      Carrier density: 1.885x10<sup>18</sup>/cm<sup>3</sup>  
 Peak Material Gain: 309.67/cm      Confinement Factor: 0.151

The graph plots show agreement on the gain peak position and the gain curvature, as required. The gain model in the time domain always tends to unity away from the gain peak – as is shown in the plot.

### **Other Tests**

You will find other tests and validation documents on your CD under the directory:  
 PicWave\Documents\ValidationDocs

## 17.2 Mode Properties - Definitions

### 17.2.1 Mode Loss

The mode loss can be estimated from the real-math waveguide mode as:

$$\alpha = \frac{\omega}{4} \frac{\int_{-\infty}^{\infty} n_r(s) \cdot \alpha(s) E^2(s) \cdot ds}{\int_{-\infty}^{\infty} P_z(s) \cdot ds}$$

where the refractive index profile is  $(n_r(s) + j \cdot \alpha(s) / (2 \cdot k_0))$  and E is the total electric field. This formula is a very good estimate for all mode polarisations<sup>12 13</sup>. This quantity is named **modeLoss0V** on the *Inspect Mode Panel* (see §5.7)

### 17.2.2 Confinement Factor

The confinement factor is calculated as:

$$\Gamma = \frac{\epsilon_0 v_c}{4} \frac{\int^R n_r(s) \cdot E^2(s) \cdot ds}{\int_{-\infty}^{\infty} P_z(s) \cdot ds}$$

where R denotes a region of the waveguide defined by the user and the lower integral is evaluated over the whole cross section.  $v_c$  is speed of light in vacuum. It is important to note that the sum of confinement factors covering all regions does **not** add up to 1.

---

<sup>12</sup> T.D. Visser, D. Lenstra, H. Blok “Polarisation sensitivity of the amplification in semiconductor optical amplifiers”, Proceedings SPIE, Photonics West, San Jose, Vol. 2994, pp. 611-622, 1997.

<sup>13</sup> T.D. Visser, B. Demeulenaere, J. Haes, D. Lenstra, R. Baets and H. Blok, “Confinement and modal gain in dielectric waveguides”, J. Lightwave Tech., Vol. 14, No. 5, pp. 885-887, May 1996.

The confinement factor is calculated so that the loss for the mode is given approximately by:

$$\alpha_{\text{mode}} \cong \Gamma_R \alpha_R$$

where R is some region of the waveguide with material absorption coefficient  $\alpha_R$ . *Do not use the confinement factor* as a “filling factor” – it does not represent the fraction of the mode power in the region R (see below).

The second confinement factor **gammaEy** includes just the Ey component of the field. This may be useful in certain quantum well phenomena.

**Note on accuracy:** The algorithms for the confinement factor and fill factor attempt to estimate these values even for layers that are much smaller than the grid size that the mode profile is evaluated on. Thus you do not need a grid spacing less than the layer thickness to get reasonable results. However shrinking your grid size will always improve your accuracy. Note also that these algorithms will not give sensible values for metallic or nearly metallic layers or even for modes that are interacting with metals.

### 17.2.3 Filling factor

The *filling factor*, denoted **fillFac**, is a measure of the fraction of the mode power flux in region R. It is defined:

$$\text{fillFac} = \frac{\int^R P_z(\underline{s}).ds}{\int^\infty P_z(\underline{s}).ds}$$

**Note on accuracy:** See “Note on accuracy” under Confinement Factor above.

### 17.2.4 TE Fraction

The **TEfrac** is the fraction of the Poynting vector with horizontal electric field:

$$\text{TEfrac} = \frac{\int^\infty (E_x \cdot H_y).ds}{\int^\infty P_z(\underline{s}).ds}$$

### 17.2.5 Intensity

When evaluating the mode profiles, the exact definition of field intensity is given by:

$$I(x, y) = \text{Re}\{\varepsilon(x, y)\} \frac{1}{4} \mathbf{E}(x, y) \cdot \mathbf{E}(x, y)^* + \frac{\mu_0}{4} \mathbf{H}(x, y) \cdot \mathbf{H}(x, y)^*$$

This is the local *energy density* and has units nJ/m<sup>3</sup>

In contrast, the power is defined by the Poynting vector:

$$\underline{P}_z(x,y) = (\underline{E} \wedge \underline{H}) \cdot \underline{z}$$

### 17.2.6 Effective Core Area

The **Effective Core Area** is defined as below:

$$A_{eff} = \frac{\left(\int I(x, y) dx dy\right)^2}{\int I(x, y)^2 dx dy}$$

### 17.3 Dispersion and Group index

The **dispersion** is defined as below:

$$dispersion = \frac{d}{d\lambda} \left( \frac{d\beta}{d\omega} \right) = -\frac{1}{2\pi c} \frac{d}{d\lambda} \left( \lambda^2 \frac{d\beta}{d\lambda} \right)$$

where  $\omega = 2\pi c / \lambda$

The **group index** is defined as below:

$$\text{Group index} = N_{eff} - \lambda \frac{dN_{eff}}{d\lambda}$$

where  $N_{eff}$  is the effective index of the mode ( $N_{eff} = \beta / k_0$ )

### 17.4 Spontaneous Emission and Spontaneous Coupling Factor (sponBetaNA)

The fraction of spontaneous-emitted light coupling into the laser modes is an important quantity determining the shape of the threshold knee and turn-on speed. This fraction has two components:

- a) the fraction of the emitted light that falls within the numerical aperture of the waveguide mode – clearly light emitted perpendicular to the laser axis will escape.
- b) The fraction of the emitted light that is emitted in a wavelength within the linewidth of the laser.

The second of these factors is taken into account automatically by the program, so the **sponBetaNA** parameter that you enter in the program should include only the geometric component – the first listed above.

In PICWAVE, the spontaneous emission is assumed to be emitted in a flat spectrum (white noise) across the free spectral range of the simulation (**lamRange**). In reality of course the spontaneous emission has very approximately a Gaussian-like shape. This means that the program in general would slightly under estimate the spectral part of the spontaneous coupling factor. If you want to get the spontaneous coupling exactly right then you may want to include in **sponBetaNA** a correction to take account of the exact shape of the spontaneous emission spectrum. Future versions of the program may allow you to define the spectral shape of the spontaneous emission.

## 17.5 The ASCII Mode Format file (.amf)

Mode profiles calculate within PICWAVE can be exported in a binary (.bmf) or text (.amf) format. These are equivalent in the information they store, but the text format is preferred if, say, you need to export/import data to/from a spreadsheet program.

The amf string uses c++ syntax. It is of the form: %[Field width].[precision][format]

There are three different formats:

e: Exponential format,

f: Fixed format,

g: General

E.g. %10.6f gives us a list of the form:

-0.092786    0.031694

%10.6e gives us:

-9.278633e-002    3.169446e-002

Other points to note:

- Lines with Ex components should only exist if hasEx is set to 1. This is true for all components.
- If profile is real (`iscomplex=0`), Ex,Ey,Hx,Hy components are the real part, Ez,Hx components are the imaginary part (all other components are zero). Else if profile is complex then component lines include both real and imaginary components:

real\_Ex(1,1) imag\_Ex(1,1) ... real\_Ex(nxseg,1) imag\_Ex(nxseg,1)

etc...

# Index

2

**2D View**, 2-42

*2D-Poly-Data*, 10-3

–

*\_PI*, 2-2

A

**A\_eff**, 5-18

*abs*, 2-3

*absolute path*, 3-5

*acos*, 2-3

**Active**, 5-13

*active layer*, 4-33, 9-4

*Active Layer Z Profiles*, 9-18

*active layers*, 11-2, 11-4

*active modes*, 4-8, 4-9, 4-35, 5-2, 5-13

**aLayer1**, 4-28

**aLayer2**, 4-28

**alConnections**, 2-22, 4-23, 11-2

**alGainKappa**, 6-1

**allSamePeriod**, 6-4

**alpha**, 5-9

**ampl**, 2-14

**ampVal**, 4-31, 4-32

*arg*, 2-3

**arm1Length**, 4-11

**As-frac**, 2-21

*ASCII Mode File (AMF)*, 5-18

*asin*, 2-3

*atan*, 2-3

**autoRun**, 5-15

**AutoRun**, 8-3

B

**biasVal**, 4-31, 4-32

*Binary Mode File*, 5-18

**Bragg Order**, 14-1

**Bragg Wavelength**, 14-1

*Bragg Wavelength Calculator*, 4-7, 4-1

*Build EV List Panel*, 2-8, 5-12, 5-13, 5-16

C

**C**, 4-35

**calcTemp**, 5-10

**carrierAccelFactor**, 9-5, 9-6

**cfseg**, 2-21, 5-10

*child*, 3-4

*children*, 3-3

*chirp*, 6-2

**chirpFrac**, 6-2, 6-4

**chirpPower**, 6-2, 6-4

*circuit*, 2-3, 2-14, 4-1, 4-2, 4-3, 4-4, 4-5, 4-5, 11-3

*circuit canvas*, 2-3, 2-21, 4-1, 4-21

**clockPeriod**, 9-14

**coeffKappaTE**, 4-11

**coeffKappaTM**, 4-11

*Colour Map*, 5-17, 5-18

**Colour Map**, 2-42

*Coloured noise*, 9-5

*command completion box*, 3-2, 3-3, 3-6, 3-13, 3-14  
*command log file*, 3-5  
*Command-Line*, 3-1, 3-2, 3-4, 3-7, 3-9, 3-13, 3-14, 3-17, 3-20  
*Command-line Options*, 3-5  
*Command-Line Window*, 3-1, 3-2, 3-5, 3-6, 3-7  
**Complex**, 5-15  
**conf. fac Ey**, 5-18  
**conf. factor**, 5-18  
*confinement factor*, 2-9  
*contact*, 4-33, 11-1  
*Contact Instruments*, 4-25, 4-33  
*Contact Properties*, 4-30, 4-34, 11-3  
**contactName**, 4-34  
*contacts*, 11-2  
*Contour Plot*, 5-17, 5-18  
*Control*, 2-18  
**coordinate system**, 5-15  
*CoreCell*, 4-7, 4-30, 9-4, 9-19, 11-2, 11-4  
*cos*, 2-3  
*cosh*, 2-3  
**coupCoeffL1R1TE**, 4-14, 4-15  
**coupCoeffL1R1TM**, 4-14, 4-15  
**coupCoeffL1R2TE**, 4-16  
**coupCoeffL1R2TM**, 4-16  
**couplingCoeff**, 4-10  
**CouplingParms**, 4-7, 6-1, 6-6, 6-7  
**cpuPriority**, 14-5  
*cross-section state*, 9-21  
**crossSectionDef**, 2-11, 2-21, 4-6, 5-1  
**crossSectionDeff**, 4-6, 4-8  
**csfeg**, 5-18  
*current drive*, 2-22, 2-23, 4-23, 4-33, 11-1  
*Current Drive Properties*, 2-23, 4-24  
*Current Flow Model*, 2-22, 4-23, 4-33, 11-2  
*Current Instrument*, 4-24, 4-29, 4-30  
*Current Instrument Properties*, 2-23, 4-24  
*Current Instruments*, 4-30  
*CurrentFlowModel*, 2-22, 4-23, 11-2  
**CurrentFlowModel**, 2-22, 4-7, 4-23, 9-4, 11-2  
**currInstrument**, 4-24  
D  
**defaultLambda**, 14-5  
**defaultNeff**, 14-5  
**defaultNgroup**, 14-5  
**defaultVal**, 10-11  
**defaultVgroupTol**, 14-5  
**delay**, 2-27, 9-14, 9-18, 15-3  
*delete*, 3-3  
*Delete Node dialog*, 4-3  
**deltaLambda**, 7-8  
**deltaLength**, 4-11  
**depth**, 4-34  
*device*, 2-1, 2-2, 2-3, 2-21, 2-28, 4-2, 2-4, 2-5  
*Device Properties dialog*, 2-2  
*device state*, 9-4, 9-5, 9-6, 5-1  
*device window*, 2-2, 2-3, 2-10, 2-22, 3-1  
**diagnosticLevel**, 14-5  
*differential quantum efficiency*, 2-25, 9-16  
**Dimension**, 5-20, 5-23  
*directional coupler*, 4-5, 4-10, 4-11, 4-15  
*Directional Coupler Properties*, 4-11  
**dispersion**, 5-18, 17-4  
**Display Instrument**, 2-27, 9-14  
*Display Modes*, 2-26  
**dlambda**, 2-16, 4-27  
**dQE**, 2-25, 9-16, 9-17  
*Drive Instruments*, 4-25, 4-29  
*drive node*, 4-33  
**driveName**, 4-24  
**drivePosn**, 4-34  
**driveR**, 4-34  
E  
*e*, 5-20, 5-23  
*ea-modulator*, 4-23  
**Effective Core Area**, 17-3

*Effective Index Solver*, 2-7, 5-10, 5-13, 5-16, 5-20, 5-22

**effGroupIndexTE**, 2-11, 4-6, 4-8, 4-10, 4-11

**effGroupIndexTM**, 4-6, 4-8, 4-10, 4-11

**effIndexTE**, 2-11, 4-6, 4-8, 4-10, 4-11

**effIndexTM**, 4-6, 4-8, 4-10, 4-11

**effLossTE**, 4-8, 4-10, 4-11

**effLossTM**, 4-8, 4-10, 4-11

*electrical signal*, 4-33

**Electro absorption Model**, 4-7, 14-4

**element type**, 2-23, 2-27

**enable**, 2-23, 4-29, 4-30

**enableSounds**, 14-5

**enableSpecMeas**, 2-24, 2-32, 4-25, 4-26, 9-7, 9-8, 15-1

**end**, 5-15

**EndN**, 8-6, 8-7, 8-11

**endVal**, 2-23, 4-31

*environment variables*, 3-10

**escapeRateFunc**, 4-7, 14-4

*etch depth*, 2-4, 2-7

**etchDepth**, 2-6, 5-4, 5-5

*etching*, 2-6

**evalType**, 14-5

*exp*, 2-3

*expression*, 2-7

*Expressions*, 2-1, 2-2, 2-2, 2-3, 2-7

**ExtrapAndRefit**, 8-6, 8-7

**ExtrapBeginN**, 8-7

**ExtrapDownStartN**, 8-7, 8-12

**ExtrapolateUpwards**, 8-7, 8-12

**ExtrapUpStartN**, 8-6, 8-7

*eye diagram*, 2-27, 2-28, 9-13, 9-14

## F

*facet*, 2-22, 4-12, 4-13, 4-17, 4-18, 4-21

*Facet Power Instrument*, 2-16, 2-24, 4-25, 4-26, 5-1

*Facet Power Instrument Properties*, 4-26, 4-27

*Facet Properties*, 4-12

*Facet Wavelength Instrument*, 4-25, 4-26

*Facet Wavelength Instrument Properties*, 4-27

*FDM Solver*, 2-7, 5-10, 5-13, 5-20, 5-21, 5-22

**FFT Size**, 9-8, 9-9, 9-10, 9-12, 15-2

**fftSize**, 4-27, 9-13

*Fibre Solvers*, 5-10

**fill factor**, 5-18

**fillFac**, 17-3

*filling factor*, 7-3

*filter*, 9-9

*filter length*, 2-31, 7-1, 7-2, 7-4

**filterWidth**, 4-27, 9-9, 9-12, 15-2

*Finite Impulse Response Filter Section*, 2-28, 4-5, 4-11, 7-1

*FIR filter*, 2-29, 2-30, 2-31, 2-32, 7-1, 7-2, 7-3, 7-4

*FIR section*, 2-28, 2-29, 2-30, 2-31, 2-32, 4-5, 4-11, 7-1, 7-2, 7-3, 7-4

*FIR Section Properties*, 7-4

**firLength**, 2-31, 7-4

**FitFrac**, 8-6, 8-13

**fitStatus**, 9-18

**flags**, 6-6, 6-7

**font**, 16-1

**format**, 5-18

*free spectral range*, 9-8

**FT Size**, 15-2

**Function**, 8-3, 8-4

*Function Editor*, 4-7, 4-4

## G

*gain*, 4-23

*gain fitting parameters*, 2-40, 8-3, 8-4, 8-5, 8-10, 8-11, 8-12, 8-13

*Gain Fitting Parameters Panel*, 2-40, 8-3, 8-4, 8-5, 8-11, 8-12

**gainKappa**, 6-1

**gainPeakFunc**, 4-7, 14-4

**gainPeakPosnFunc**, 4-7, 14-4

**GainScale**, 8-7

**gainVolFrac**, 6-1



**gammaEy**, 5-18, 17-3  
*general join*, 2-13, 4-16, 4-17, 4-19, 4-22, 4-35  
*General Join Properties*, 4-17  
*general scattering matrix*, 4-12, 4-17  
*General Scattering Matrix editor*, 4-17, 4-20  
**genSMat**, 4-17  
**genSMatTE**, 4-13, 4-14, 4-15, 4-17, 4-18  
**genSMatTM**, 4-13, 4-14, 4-15, 4-18  
**GivePriorityToPeak**, 8-6  
*global object*, 3-7, 3-8  
**gOrder**, 6-1, 6-4  
**gPeriod**, 6-1, 6-4  
**gPhase**, 6-1  
**graphMax**, 9-16  
**graphMaxX**, 9-15, 15-3  
**graphMaxY**, 9-15  
**graphMinX**, 9-15, 9-16, 15-3  
**graphMinY**, 9-15  
**graphStep**, 9-2, 9-3  
*grating*, 4-7, 6-1, 6-7  
*grating layer*, 6-4  
*grating order*, 6-1, 6-4  
*grating period*, 6-1, 6-4  
**Grating Period**, 14-1  
*grating phase shift*, 6-1  
*grating profile*, 4-7, 6-1, 6-3, 6-4, 6-5, 6-6, 6-7  
*grating set*, 4-7, 6-1, 6-3, 6-4  
*Grating Set Editor*, 6-3, 6-6, 6-7  
**gratingNo**, 6-7  
*gratings*, 4-7  
**gratPhaseShift**, 4-7, 6-3  
*grid*, 3-6  
*Grid Settings dialog*, 3-6  
**Gridsize**, 5-19, 14-2  
**group index**, 5-18, 17-4  
H  
*handles*, 3-1  
**HCurv**, 5-20  
**Hsymmetry**, 5-20, 5-21  
I  
*imag*, 2-3  
**Imax (mA)**, 9-17  
**Imin (mA)**, 9-17  
**impedance factor**, 5-11  
**ImpedFact**, 5-5  
*imported S-Matrix spectrum*, 2-29, 2-30, 2-31, 7-1, 7-2, 7-3, 7-4  
*Impulse*, 4-32, 5-1  
**InitialStateMode**, 9-4, 9-5  
**inOrOut**, 2-16, 4-26  
*input variables*, 3-2  
**insLossTE**, 4-13, 4-14  
**insLossTM**, 4-13, 4-14  
*Inspect FIR filter*, 2-30, 7-3, 7-4  
*Inspect Mode*, 5-15  
*Inspect Mode Panel*, 2-8, 2-10, 5-14, 5-17, 7-2  
*Inspect Mode panel*, 5-17  
*Instrument Results*, 9-6  
*instruments*, 2-18, 4-25, 4-26, 9-6  
**instruments**, 2-15, 2-24, 4-7, 4-25  
*Instruments Results*, 2-18, 2-25  
**Intensity**, 2-19  
*Intensity/RIN spectrum*, 9-10  
**Intensity/RIN Spectrum**, 15-2  
**interfacePosn**, 6-4  
**interval**, 9-10  
**isGeneral**, 4-13, 4-14  
**isLossy**, 4-14  
**isReflecting**, 4-14  
**isSymmetric**, 4-14  
**Ith**, 2-25, 9-16, 9-17  
J  
*join*, 2-12, 2-13, 3-4, 4-1, 4-12, 4-12, 4-15, 4-21  
*Junction Current Density Instrument*, 4-28

*Junction Current Density Instrument Properties*, 4-29

*Junction Current Instrument*, 4-28, 4-30

*Junction Current Instrument Properties*, 4-29

*Junction Gain Instrument*, 4-28

*Junction Gain Instrument Properties*, 4-29

*Junction Instruments*, 2-24, 4-7, 4-25, 4-28

*Junction Ne Instrument*, 4-28

*Junction Ne Instrument Properties*, 4-28, 4-29

*Junction Temperature Instrument*, 4-28

*Junction Temperature Instrument Properties*, 4-29

*junction voltage*, 10-10, 11-2

*Junction Voltage Instrument*, 4-28

*Junction Voltage Instrument Properties*, 4-29

K

*Kappa Calculator*, 6-1, 6-6, 6-7

L

**L**, 4-35

**Lambda**, 2-7

**lambdaCentre**, 2-18, 2-25, 2-29, 2-30, 2-38, 4-27, 7-3, 8-1, 8-4, 8-5, 8-10, 9-2, 9-4

**lambdaMax**, 7-7

**lambdaMin**, 7-7

**LambdaShift**, 8-7

**lamRange**, 2-18, 2-29, 2-30, 2-31, 2-40, 7-2, 7-3, 7-4, 9-2, 17-4

**lateralRange**, 9-19

*layer*, 2-4, 2-5, 5-10

**layerRange**, 9-19

**length**, 2-11, 2-14, 2-21, 2-23, 2-27, 2-30, 4-6, 4-9, 4-10, 4-11, 4-31, 4-32, 12-1

**lhsBC, topBC, botBC, rhsBC**, 5-5

*LI Fit*, 2-25, 9-6, 9-16, 9-17

*LI plot*, 2-26

*lininterp*, 2-3

*link*, 2-14, 2-22, 3-3, 4-21

*ln*, 2-3

*local SWGs*, 5-7

**longLife**, 2-19, 9-7, 9-8

**lossKappa**, 6-1

**lossTangent**, 14-3

M

*Mach Zehnder Interferometer*, 4-5, 4-11

*Mach Zehnder Interferometer Properties*, 4-11

*Main*, 2-24

**mat**, 2-20, 2-21, 2-35, 5-10

**Material**, 2-39, 8-2, 8-3, 8-4

*Material Database*, 2-20, 5-5, 5-10, 10-1, 10-15, 4-1, 4-2

**material database**, 5-5

**Material Database File**, 8-3

*Materials Inspector*, 10-15, 4-1

*max*, 2-3

**max**, 10-11

**max Nmodes**, 2-7, 2-20, 5-15, 5-22

**maxHeight**, 6-4

**maxLambda**, 2-40, 8-5

**maxN**, 2-40, 8-4, 8-6, 8-7, 8-11, 8-11, 8-12

**maxVal**, 2-27, 4-31

**merge active layers**, 9-4

*Microstrip Calculator*, 4-34, 11-3, 4-2, 4-3

*min*, 2-3

**min**, 10-11

**Min TEfrac**, 2-7

**minHeight**, 6-4

**minLambda**, 2-40, 8-5

**minLength**, 9-11

**minN**, 2-40, 8-4, 8-12

**minPeakWidth**, 9-11

**minVal**, 2-27, 4-31

*Mode builder*, 5-15

*Mode Data*, 9-20

**Mode Eff Index**, 14-1

*Mode Finder*, 2-7, 2-8, 2-10, 5-12, 5-13, 5-14, 5-15, 5-16

*Mode List*, 2-8, 4-35, 5-2, 5-12, 5-13, 5-14, 5-15, 5-16, 5-22, 6-7  
*mode loss*, 2-9  
*Mode Power*, 9-19  
*Mode Solver*, 5-15  
*Mode Tracker*, 9-10, 9-11, 9-12, 9-13  
**ModelIndex**, 4-22, 4-27, 4-35  
**modeLossOV**, 5-18, 17-2  
**modeNo**, 6-7  
**modePol**, 2-11, 4-6, 4-10, 4-11, 4-22, 4-27, 4-35  
**modPol**, 4-29  
*modulation response*, 9-10  
*MOLAB*, 5-12, 5-13, 5-14, 5-15, 5-22  
*MOLAB Options*, 2-7, 2-8, 2-11, 2-12, 2-20, 5-16, 5-22  
*MOLAB Options Panel*, 5-13  
*monitor*, 2-15, 2-15, 2-16, 2-24, 4-24, 9-7  
**Monitor**, 2-19  
*monitor instruments*, 4-25  
*Monitor Instruments*, 4-25, 4-26  
*Monitor Instruments panel*, 2-15  
*Monitor Instruments Panel*, 4-25, 4-26  
*Monitor Properties*, 2-15, 2-16, 2-24, 4-25, 4-26  
**monitorName**, 4-25  
*move*, 3-2  
**msgLevel**, 14-5  
*Multiple Lorentzian Gain Fitter*, 2-33, 2-34, 2-39, 2-41, 2-42, 8-1, 8-2, 8-13, 11-7  
*multiple Lorentzian gain model*, 2-33, 2-38, 2-39, 2-41, 2-42, 8-1, 8-2, 11-7  
**mx:AI-frac**, 2-35  
*MZI*, 4-5, 4-11  
  
**N**  
**NO**, 2-40, 8-5  
**ncellW**, 2-21, 4-7, 11-4  
**neff**, 4-35  
**New element type**, 4-32  
*New Node dialog*, 2-1, 2-2, 4-1, 4-3, 2-4  
  
*New Variable dialog*, 2-5  
*node*, 2-5, 3-5  
*noise*, 9-5  
*noise sources*, 9-5  
*notes*, 4-3, 4-4  
**NPeaks**, 9-11  
*NRZ*, 4-31, 4-32  
**NScale**, 8-7  
**nSteps**, 7-7  
*nudge*, 3-2  
**numBins**, 9-15, 9-16  
**numContacts**, 2-21, 4-7, 4-23  
**NumExtrapPoints**, 8-7  
**NumLambdaSamples**, 8-6  
**numLHSPorts**, 4-17  
**NumLoopSteps**, 8-6  
**numRHSPorts**, 4-17  
**numSubSections**, 4-9  
**numTEModes**, 4-8, 4-35  
**numTMModes**, 4-8, 4-35  
**nx**, 5-16, 5-17, 5-18, 5-19, 5-20, 5-22  
**nx, ny**, 5-15  
**NxTransparency**, 9-4, 9-5  
**ny**, 5-16, 5-17, 5-18, 5-19, 5-20, 5-22  
  
**O**  
*object*, 3-8  
**offset1**, 4-34  
**offset2**, 4-34  
*Optical Power Instrument Properties*, 4-22  
*optical signal*, 2-13, 4-33  
*optical source*, 2-13, 2-14, 4-16, 4-22, 4-33  
*Optical Source Properties*, 2-13, 4-22, 4-29  
*optical spectrum*, 9-8  
*Optical Wavelength Instrument Properties*, 4-22  
**optimisation**, 5-15  
**optoSourceName**, 4-22  
*origin*, 3-2  
*Oscilloscope*, 2-27, 2-28, 9-6, 9-13, 9-14, 9-15

*output log file*, 3-5

## P

*parameter*, 2-1, 2-2, 2-7

*parameter editor*, 5-15

*path name*, 3-4

**PeakCurWeight**, 8-6

**PeakFitFrac**, 8-6

**phaseShift**, 6-4, 6-5

**plot life**, 5-17, 14-2

**plotInMonitor**, 2-18, 2-25, 4-25, 9-1

**PmlWidth**, 5-5

**Polarisation**, 8-2, 8-4, 9-20

*polish*, 5-16

*Polish Existing List*, 5-16

*port*, 2-12, 2-13, 2-14, 3-4, 4-6, 4-12, 4-22

*pow*, 2-3

*power coupler*, 4-10, 4-15, 4-16, 4-17, 4-18

*Power Coupler Properties*, 4-15

*Power Instrument*, 4-22, 4-29

*power splitter*, 2-12, 4-10, 4-13, 4-14, 4-15, 4-17, 4-18, 4-21

*Power Splitter*, 2-12

*Power Splitter Properties*, 4-14

**powerInstrument**, 4-22

*Pre-Fit Analysis*, 2-39, 2-40, 8-3, 8-4, 8-6, 8-11

*project*, 2-1, 2-2, 2-3, 4-1, 4-3, 4-4, 5-1, 5-7, 6-3, 2-4

*Project Node*, 5-7

*Project Node Properties dialog*, 4-2

*Project tab*, 4-1, 2-4

*Project Tree*, 2-1, 2-2, 2-3, 4-1, 4-2, 4-3, 4-4, 5-7, 5-8, 2-4, 3-2, 3-13

*Project Window*, 2-1

*prompt line*, 3-1

*properties*, 3-3, 4-5

*Properties dialog*, 4-5

*Pulse Analysis*, 9-15, 9-16

**Pulse-W**, 9-16

## Q

*Quantum Efficiency Instrument*, 4-25, 4-26

## R

**R**, 4-35

**Ral**, 11-2

*Ramp*, 4-31

**RandSeed**, 9-5

*real*, 2-3

*Real FDM Solver*, 5-12

**realKappa**, 6-1

**ref index**, 5-9

**ref index**, 2-5

*reference*, 3-4, 3-7, 3-8

**Reference**, 2-19, 9-8

*reference section*, 2-17, 2-18, 2-28, 4-9, 9-2, 11-4

**reflCoeffTE**, 4-13, 4-14, 4-15, 4-16

**reflCoeffTM**, 4-13, 4-14, 4-16

**reflCoupCoeffTE**, 4-14

**reflCoupCoeffTM**, 4-14

**refractive index**, 12-1

**refSectionName**, 2-17, 2-18, 2-29, 9-2

*relative path*, 3-5

*Rename Variable dialog*, 2-8

**responseTime**, 2-16, 4-27, 4-28

**respTime**, 9-11

*RIN spectrum*, 9-10

**rix (aniso)**, 5-10

**rLLTE**, 4-13

**rLLTM**, 4-13

**rms error**, 9-18

*Root Box*, 5-13

*root project*, 2-4, 2-5

*rotate*, 3-4, 4-5, 4-12

**rRRTE**, 4-13

**rRRTM**, 4-13

*Run-time Monitor Window*, 2-18, 2-25, 4-25, 9-1, 9-2

*RWG*, 2-3, 2-4, 2-11, 2-12, 2-20, 4-5, 4-6, 4-8, 4-9, 4-35, 5-1, 5-2, 5-3, 5-4, 5-6, 5-8, 5-13, 6-1, 6-3, 6-5, 6-6, 6-7

*RWG Editor*, 2-4, 2-6, 2-20, 5-2, 5-4, 5-6, 5-7, 5-9

**RWG Name**, 6-7

*RWG Waveguide*, 5-1, 5-2, 5-6, 5-7, 5-12, 2-4, 3-2

*RWG Window*, 2-4, 2-6, 2-7, 2-12, 2-20, 5-1, 5-2, 5-4, 5-6, 5-8, 5-10, 5-14, 10-1

*RZ*, 4-32

S

*S-Matrices*, 4-12

*S-Matrix*, 4-12, 4-17, 4-18, 4-19, 4-20

*S-Matrix spectrum*, 2-29, 2-31, 7-1, 7-2, 7-3, 7-4, 7-5, 7-6, 7-7, 7-8

*sampled S-Matrix spectrum*, 2-31, 7-1, 7-2, 7-3, 7-4

**sampleMaxT**, 9-15, 15-3

**sampleMinT**, 9-15, 15-3

**saturation**, 9-18

**scale**, 16-3

**scanner**, 5-14

*SciGraph*, 6-4

*Script Manager*, 3-6

*scroll cursor*, 3-4

*scroll tool*, 3-1, 3-4

*section*, 2-12, 2-13, 2-14, 3-4, 4-1, 4-5, 4-21, 11-3

*section length*, 4-6

**sectionName**, 2-21, 4-6, 4-10, 4-11

**sectionType**, 2-11, 2-21, 4-6, 4-23

**seed**, 2-27, 4-32

*select cursor*, 3-1

**select field**, 5-17

*select tool*, 3-1, 3-2, 3-3

**set totalTime**, 2-24

**Show Interpolations**, 8-4

**side loss**, 5-18

*side mode suppression ratio*, 9-12

*sign*, 2-3

*signal*, 4-33

**signal**, 2-13, 2-23, 4-22, 4-24

*Signal Editor*, 2-13, 2-14, 2-23, 4-22, 4-24, 4-30, 9-2

*signal element*, 2-27, 4-31, 4-32, 5-3

*signal function*, 4-30

*signal-element*, 4-31

*sin*, 2-3

*Sine*, 4-31

*single Lorentzian gain model*, 2-33, 2-41, 10-12, 11-5

**skipStep**, 4-27, 9-13

*slice*, 2-4, 2-5, 2-6, 2-7, 5-1

*SMSR*, 9-12, 9-13

*SMSR Instrument*, 4-25, 4-26, 9-12, 9-13

*SMSR Instrument Properties*, 4-27

*snap spacing*, 3-2, 3-7

*snapping to grid*, 3-2, 3-6, 3-7, 3-7

**Solver**, 5-15

*Solver Parameters*, 5-16

*Source Instruments*, 4-25, 4-29

*spaces as delimiters*, 6-4

*Spectra Results*, 2-18, 2-19, 2-26, 9-6, 9-7, 9-11, 9-12, 5-2

*spline*, 2-3

**sponBetaNA**, 4-7, 17-4

*sqrt*, 2-3

**start**, 5-15

**startVal**, 2-23, 4-31

**style**, 16-3

*subnodes*, 3-5

*substrate temperature*, 4-6

*SWG*, 2-4, 2-5, 2-7, 2-9, 2-20, 5-1, 5-2, 5-5, 5-6, 5-7, 5-9, 5-10, 6-5, 6-6

*SWG Editor*, 2-4, 2-20, 5-2, 5-5, 5-10, 5-18

*SWG node*, 5-8

*SWG Waveguide*, 5-7

*SWG Window*, 5-8

*SWGs*, 5-6

**switchLevel**, 9-15, 9-16

**Sync Trigger**, 2-27, 9-14, 15-3

T

**tailWidth**, 2-31, 7-4

**tAlign**, 15-2

*tan*, 2-3

*tanh*, 2-3

*taper section*, 4-8, 4-9, 5-1, 9-20

*Taper Section Properties*, 4-9

*taper section.*, 4-35

*TDTW*, 9-1, 11-3

*TDTW Calculator*, 2-16, 2-17, 2-18, 2-19, 2-24, 2-25, 2-27, 4-25, 8-1, 8-4, 8-5, 9-1, 9-3, 11-4, 5-1

*TE frac*, 2-9

**TE frac**, 5-18, 17-3

**TE fractions**, 5-15

**TE mode idx**, 9-20

*TE S-Matrix*, 4-18

*TE-fraction*, 5-14

**temperature**, 5-5, 8-3, 8-4, 12-1

**termPosn**, 4-34

**termR**, 4-34

*Text Block*, 5-2

**thickness**, 2-5, 5-9

*threshold current*, 2-25, 9-16

**Ti**, 4-32

*Time Domain Results*, 2-18, 2-19, 2-25, 2-32, 4-25, 9-1, 9-2, 9-6, 9-7

*time domain travelling wave*, 11-3

**Time Evolving**, 2-19, 2-26, 9-8, 9-9

*time window*, 9-8

**Time Window**, 9-8, 9-9, 9-10

*time-averaged spectrum*, 9-8, 9-9

*time-resolved spectrum*, 2-26, 9-8, 9-9, 9-10

**tLRTE**, 4-13

**tLRTM**, 4-13

**TM**, 4-15, 4-17

**TM mode idx**, 9-20

*TM S-Matrix*, 4-18

**totalTime**, 2-17, 2-18, 2-27, 9-1

**Tp**, 2-27, 4-31

**Tp2**, 4-31

**Tr**, 4-31

*Travelling Wave Electrode Model*, 4-30, 4-33, 4-34, 11-1, 11-2, 4-2, 4-3

**tRef**, 9-10

**tRLTE**, 4-13

**tStep**, 2-17, 2-18, 2-28, 2-30, 2-31, 2-38, 4-9, 8-1, 8-4, 8-5, 8-10, 9-2

**Tsub**, 2-21, 4-6

**twmEnable**, 4-34

**twModel**, 4-34

**type**, 2-19, 4-13, 6-4

U

**userfile**, 6-4

V

**valueName**, 4-25

*variable*, 2-1, 2-2, 2-3, 2-4, 2-5, 2-7, 2-8

*Variable names*, 2-6

*Variables*, 2-1, 2-3

*variables node*, 2-3, 2-4, 2-5

*Vector Plot*, 5-17, 5-18

*voltage drive*, 4-23, 4-33, 11-1

*Voltage Drive Properties*, 4-24

*Voltage Instrument*, 4-24, 4-29, 4-30

*Voltage Instrument Properties*, 4-24

**voltInstrument**, 4-24

**Vsymmetry**, 5-20, 5-21

W

*waveguide section*, 2-3, 2-10, 2-11, 2-12, 2-21, 2-22, 2-24, 4-5, 4-6, 4-35, 5-1, 11-2, 11-4

*Waveguide Section Instruments*, 2-24, 4-7, 4-28

*Waveguide Section Properties*, 2-10, 2-21, 2-22, 2-24, 4-6, 4-23, 4-28

**Wavelength**, 12-1

*Wavelength Instrument*, 4-22, 4-29

**wavelengthInstrument**, 4-22

*white noise*, 9-5  
**width**, 2-6, 2-35, 5-4, 5-5  
**width1**, 4-34  
**width2**, 4-34  
*window*, 2-5  
**windowW**, 9-8, 9-10

X

*X Axis*, 9-9  
*X Section*, 9-21  
*X Section state*, 9-18  
**X-Axis**, 2-19  
**xLateral1**, 4-28  
**xLateral2**, 4-28  
**XMax**, 4-5  
**xmax**, 5-17, 5-19, 9-9, 15-2  
**XMin**, 4-5  
**xmin**, 5-17, 5-19, 9-9, 15-2  
**xsize**, 12-2

Y

*Y-Junction*, 4-5, 4-10  
*Y-Junction Properties*, 4-10  
**ymin**, 5-17, 5-19  
**ymax**, 5-17, 5-19

Z

**Z**, 4-35  
*z-element*, 4-30, 9-19, 11-3, 11-4  
**zFrac**, 9-21  
*Z-profile*, 9-18, 9-19, 9-20  
**zEnd**, 9-20  
**zFrac**, 4-30  
**zFraction**, 2-24, 4-28  
**zg1, zg2, zg3**, 6-4, 6-5  
**ZMax**, 4-5  
**ZMin**, 4-5  
*zoom in cursor*, 3-4  
*zoom out cursor*, 3-4  
*Zoom to Aspect Ratio dialog*, 3-5  
*zoom tool*, 3-1, 3-4, 3-5  
**zStart**, 9-20  
**zStep**, 2-17, 2-18, 2-24, 2-29, 2-30, 4-9, 4-30, 7-2, 7-3, 9-2, 9-4, 11-4