PHOTOSS

The Photonic System Simulator

User Manual

PHOTOSS 5.92

Creation Date:
October 16, 2013

Jens Lenge
Celler Straße 25
59192 Bergkamen

PHOTOSS manual in LaTeX by:
Dipl.-Phys. Matthias Westhäuser and Dipl.-Inf. Nicolas Luck
Extensive PHOTOSS manual rewrite of basic concepts chapters by:
Dipl.-Phys. Matthias Westhäuser

Contents

1 Introduction

1.1 Foreword

This chapter deals with a general introduction to the PHOTOSS application and its underlying concepts and has been written to give you a general overview of its capabilities. All concepts will be explained in a concise manner; much more detailed information on each concept can be found in the following chapters of the manual as well.

You may also notice that PHOTOSS has undergone several major additions to its list of capabilities. For example, the PScript concept has been developed to enhance PHOTOSS with its own script engine. Because the functionality of PScript is so extensive, these concepts will be described in much more detail in separate manuals.

1.2 What is PHOTOSS?

PHOTOSS is a stand-alone package for the simulation of photonic transmission systems. It does not depend on an underlying simulation framework.

The user interface allows the graphical design and analysis of optical systems by placing and connecting components on the grid. The extensible component library contains predefined components with customizable parameter sets.

Various components are represented by models of different complexity: Physical models enable deep insight into the phenomena influencing a system, while behavioral models account for a simple adaptation of component properties to experimental data. Additional user-specific models can be added using the included programming interface.

In order to overview and manage complex setups more easily, components may be grouped together into reusable sub-networks, thereby forming functional units such as wavelength converters, optical add-drop multiplexers, or cross-connects.

1.3 Underlying Concepts

The main purpose of the PHOTOSS application is to carry out numerical *simulations* of optical / photonic transmission systems. Generally, a simulation of these systems incorporates the interplay between underlying PHOTOSS concepts to produce an accurate emulation of the real, physical system. In this subsection we will address the individual parts of a simulation and the concepts behind them. A more detailed overview of each concept can be found in the subsections of chapter 3; concepts will be emphasized using italics to denote them in this subsection.

In general, a *simulation* of PHOTOSS is stored in a simulation file which has a designated format - the ".pho format". A simulation file can be loaded, modified, saved and - of course - be transferred to any number of computers on which you may want to run this simulation. Depending on the complexity and content of the simulation, additional files have to be copied along with the simulation file[1].
You may always open any number of simulations in PHOTOSS at the same time. However, only one simulation may be executed in a single instance of PHOTOSS. If you want to execute multiple simulations at the same time, you should consider using multiple instances of PHOTOSS. There is generally no interaction between simulations of different PHOTOSS instances on any conceptual level.

[1] e.g. files holding a bit pattern, wave plate distribution, PScripts, etc.

Each individual simulation contains a set of *simulation parameters* which applies only to its own scope. Naturally, different simulations may have different sets of simulation parameters. The scope of simulation parameters is global for each simulation.

A PHOTOSS simulation naturally contains one or multiple *components*. Components may represent physical models - e.g. a silica fiber or a photo diode - or algorithmic/numerical models - e.g. components used for digital signal processing (DSP) such as a Constant Modulus Algorithm (CMA). There is a large number of available components which are already included in PHOTOSS but you may always use other concepts and interfaces to create your own components and use them with PHOTOSS, if necessary. Interaction between user-defined and standard components can be carried out via “Interfaces”.

Each PHOTOSS component has a set of one or more *component parameters*. They work only on the scope of the component itself. However, there is an exception to this rule: Special components such as the *Network* and the *Iterator* may also contain further components in a nested sub-structure within them. Thus, parameters of networks and iterators may also apply to the scope of all included (sub-)components inside them.

Parameters in PHOTOSS have an allowed parameter range corresponding to physical or numerical constraints (not all parameter values are appropriate for each parameter). Parameters may be changed to any value inside the allowed range. Component Parameters may also use a Simulation Parameter or a formula as placeholders or a formula instead of a concrete value.

Components in PHOTOSS may have in-ports or out-ports. Components may have either no in- or out-port or a defined positive number of in- and/or out-ports.
Components are connected to each other by *links* which connect their in- and out-ports. A simulation link does not represent any physical “fiber” or “cable” - it is a fictional concept to illustrate the connection and sequence of components. Any number of links may begin at an out-port of a component; however, only one link may end in each in-port of a component.
Depending on their in- and out-port configuration, components may be considered as transmitters (e.g. the Pulse Generator or a Laser Source), intermediate components (e.g. the silica fiber), or receivers (e.g. an Eye Analyzer). Some components do not fit into this strict scheme and may be categorized as “something-in-between’ - e.g. multiplexers.

In PHOTOSS, “information” which is exchanged between components via the links is represented by a *signal*. Generally, a signal can be defined in the optical domain (amplitude, phase and polarization) or in the electrical domain (amplitude and phase only). However, the signal itself contains not only the electrical or optical field, but also additional information which is passed along between the components (see below).

A PHOTOSS signal is a complex structure including also the modeling of *Noise* and the *Modulation* and underlying bit pattern of a signal. Generally, each signal has certain physical characteristics concerning its *Simulation Band and Channel Structure*. These are both determined by the simulation parameters and the setup of the simulation links (e.g. when multiplexers are used).
PHOTOSS signals are only working on the scope of a sequence of links and components. Normally, signals are not exchanged between simulations. If, however, this should become necessary, special components such as the File Loader and File Saver may be used to load and save signals to files to incorporate them in different simulations.

Aside from the connection of components via links, there is also an additional concept that allows for the creation of an informational path in the optical domain when including polarization mode dispersion effects in the simulation - the *PMD Path*. Some polarization-sensitive components (e.g. the PMD and PDL Emulator or the Birefringent Element) may be “members” of this specific PMD Path and they propagate information relevant to the estimation of the overall Jones Matrix for the specific PMD Path of a simulation. This information is not propagated as a signal or within the signal structure inside the simulation links.

When a simulation in PHOTOSS is carried out, a *Simulation Run* will be executed. This run is comprised of several sub-steps which make up the whole process of carrying out a simulation. These substeps include the determination of the sequencing of the components (a transmitter has to be calculated before the receiver connected to it, etc.) which is denoted as *Scheduling*. After that, the signal is propagated along all paths defined by the components and their links. After all components have consecutively carried out their calculations, and the signal has "reached" the last component in the sequence, the simulation is finished.

During a simulation run, the *Random Number Generation* may automatically be carried out to emulate statistical processes such as the modeling of noise or determination of a random walk optimization procedure.

Once a simulation is finished, *Results* are available and have been calculated by the components. Not all components actually define results, but observables of interest are often available after the simulation run is completed. Results can be displayed in the GUI or written to individual result files in a text format. Some Results can be displayed best by making use of visualization - e.g. by plotting a graph or creating a diagram. Some components support this illustration by use of *Visualizers*. Visualizers are integrated in components and can be utilized after the simulation once the according component has finished its calculation. A visualizer may also be called up while the simulation is still running.

The *Path Analysis* is a special tool which is relevant for looking at the propagation of an optical field *inside* the Single Mode Fiber component of PHOTOSS. In contrast to all other components, the Path Analysis offers predefined receiver structures *inside* the Single Mode Fiber to allow for the calculation of results as the signal propagates the whole distance through the fiber. This tool may for example become relevant when the power distribution along the fiber is of interest and nonlinear effects are being considered.

In addition to all other concepts presented above, the following concepts may be utilized to create, manage and handle even more complex simulations in PHOTOSS:

If you are interested in carrying out simulations for more than one set of parameters, a *Parameter Variation* may be used. In a Parameter Variation, all Simulation Parameters may be varied in as many steps as needed. Steps may be created systematically according to a simple rule (minimum value, maximum value and step size) or non-systematically (any sequence of valid values).
A Parameter Variation is not limited to a one-dimensional variation - instead, it may be a variation along any Simulation Parameter dimension, thus allowing for very complex and demanding calculations.
In a Parameter Variation, multiple simulation runs are carried out where each simulation run corresponds to one set of Simulation Parameters drawn from the variation set. Typically, the Simulation Parameters are used as placeholders for one or more component parameters.
After the Parameter Variation has finished, the results of all variation runs are available - usually stored in result files in text format to achieve the best-possible overview and allow for post processing with special data handling tools.

The underlying structure of a Parameter Variation is a PHOTOSS Script - or, in short, a *PScript*. A PScript consists of an algorithmic sequence of commands which make up e.g. a Parameter Variation or the automatic construction of a simulation (including component creation, linking, etc.). The algorithmic sequence may be highly complex - much higher than in a "Standard Parameter Variation". The PScript tool allows for the easy creation of user-scripts which may be used to create even the most complex variations and e.g. to automatically start the analysis of simulation results by invoking other applications.

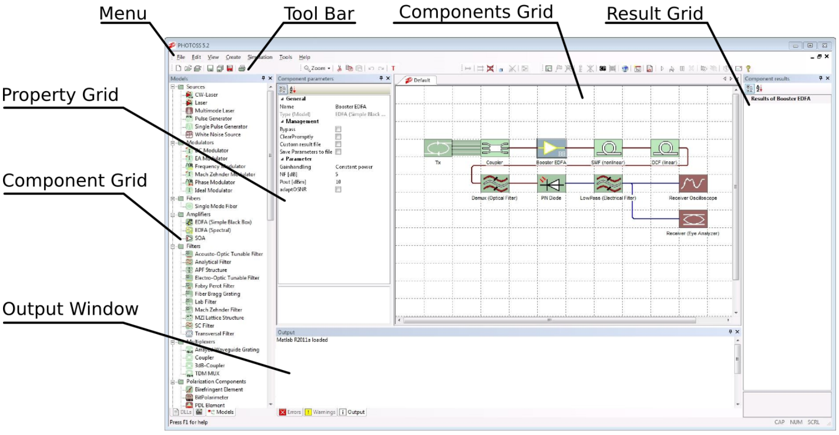
2 Graphical User Interface

2.1 Introduction

This chapter deals with the graphical user interface of the PHOTOSS application. Using the GUI is the easy most way to quickly create and modify your simulation or to take a look at simulation results. You can, however also control all relevant PHOTOSS operations by using a PScript. This may be beneficial for large and complex parameter variations which are carried out and which do not need user-GUI interaction. Please keep in mind that the layout of the PHOTOSS GUI may depend on your type of operating system and (e.g.) on your chosen “window styles”.

2.2 Main Window

After starting PHOTOSS, the main window displayed in figure 2.2.1 appears. At the top, you find the menu bar along with several tool bars. The major part of the main windows is occupied by one or more component grids, where simulation setups or sub-networks are assembled by connecting elementary components. The component tree is docked to the left by default. It provides access to all available network components, which can be placed on a grid by dragging and dropping them with the computer mouse. When a simulation is running, the output window at the bottom provides information about the current simulation status and serves as a log for the simulation progress.



General
Name Booster EDFA
Type (Model) EDFA (Simple Black ...
Management
Bypass □
ClearPromptly □
Custom result file □
Save Parameters to file □
Parameter
Gainhandling Constant power
NF [dB] 5
Pout [dBm] 10
adaptOSNR □

Matlab R2011a loaded

Figure 2.2.1. PHOTOSS Main Window.

Note that most toolbars and auxiliary windows may be freely moved, resized and docked to almost every edge of the main window (or of other toolbars) in order to customize the appearance of the workspace to meet the user’s preferences. Therefore the appearance and position of these elements may differ from the above picture.

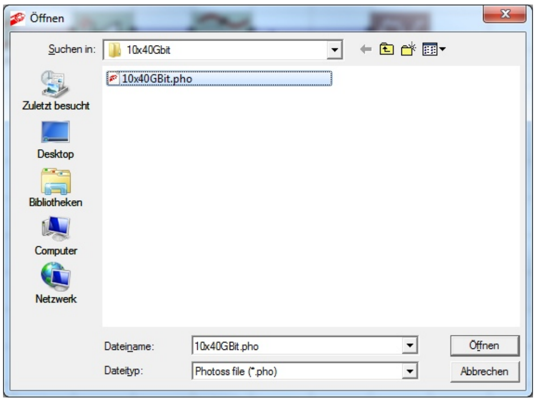
The different parts of the main window are explained in more detail in subsequent sections.

2.3 Main Toolbar

The main toolbar provides functions for file handling, editing, printing, and access to the online help system.

- **New.** Create a new simulation. A file selector dialog will appear where you can specify a filename and a directory for the new project. PHOTOSS simulation files use the file extension ".pho".

- **Open.** Open a simulation file. Select an existing file (.pho) in the file selector dialog.



Suchen in: 10x40Gbit

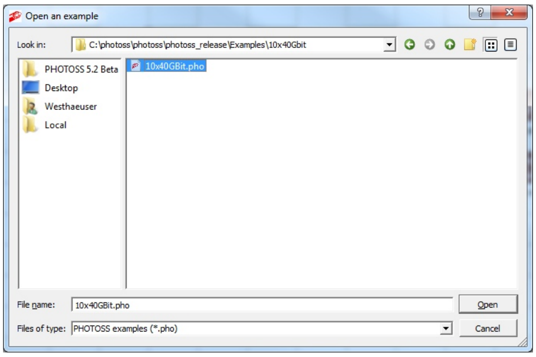
10x40GBit.pho

Dateiname: 10x40GBit.pho

Dateityp: Photoss file (*.pho)

Figure 2.3.1. PHOTOSS File Open Dialog Window.

- **Open an example.** Open an example file. Select an existing file (.pho) in the file selector dialog. Various example simulation setups are present in order to easily get familiar with PHOTOSS.



Look in: C:\photoss\photoss\photoss_release\Examples\10x40Gbit

10x40GBit.pho

File name: 10x40GBit.pho

Files of type: PHOTOSS examples (*.pho)

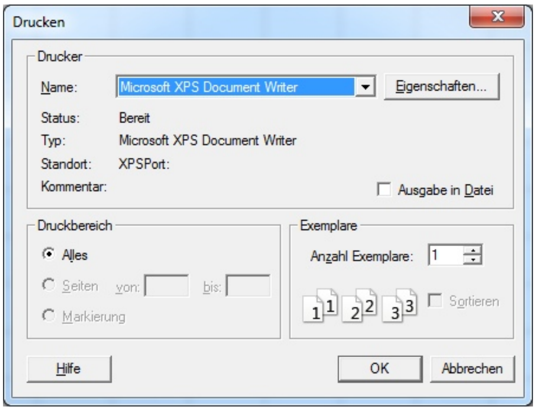
Figure 2.3.2. PHOTOSS Open Example Dialog Window.

- **Save / Save as.** Save the current simulation file (.pho). "Save" keeps the existing path and filename while "Save as" opens a file selector to choose a new path or filename.

- **Save all.** Save all currently opened simulation files (.pho). "Save all" keeps the existing paths and filenames, thereby overwriting previously saved files.

- **Save component parameters.** Save a Rich Text Format (.rtf) file which lists the parameters of all components used in the current simulation, as well as the simulation parameters and the path analysis parameters. This makes it easy to generate data sheets for word processing applications such as Microsoftï¿½ Wordï¿½.

- **Print.** Print the active simulation setup.



Drucken

Drucker

Name: Microsoft XPS Document Writer

Status: Bereit

Typ: Microsoft XPS Document Writer

Standort: XPSPort:

Kommentar:

□ Ausgabe in Datei

Druckbereich

● Alles

○ Seiten von: bis:

○ Markierung

Exemplare

Anzahl Exemplare: 1

□ Sortieren

Figure 2.3.3. PHOTOSS Print Dialog Window.

2.4 Edit Toolbar

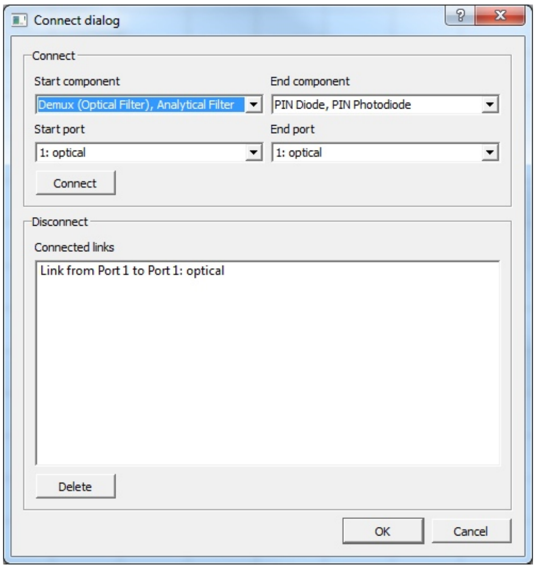
The edit toolbar provides undo/redo functionality, zooming, cut/copy/paste functions, and access to the text tool.

- **Undo / Restore.** Undo or restore the last action.
- **Zoom.** Adjust the zoom factor using a pop-up menu.
- **Cut.** Cut selected elements from the component grid.
- **Copy.** Copy selected elements from the component grid.
- **Paste.** Paste previously cut/copied elements to the component grid.
- **Text.** Insert text on the component grid.

2.5 Link Toolbar

The link toolbar consists of tools for generating networks, connecting components and setting an optical path for further analysis.

- **Connect tool.** Opens the “Connect Components” dialog to connect currently selected components. See section “Connecting Components” for details.



Connect dialog

Connect

Start component: Demux (Optical Filter), Analytical Filter

End component: PIN Diode, PIN Photodiode

Start port: 1: optical

End port: 1: optical

Disconnect

Connected links: Link from Port 1 to Port 1: optical

Figure 2.5.1. PHOTOSS Connect Components Dialog Window.

- **Connect selected components.** Connect all selected components as far as the respective number of input and output port match. See section “Connecting Components” for details.
- **Disconnect all.** Delete all links connecting the selected components.
- **Bypass component.** Skip the calculation of the selected components without actually deleting them. The components may be re-activated later on.
- **Undo bypass.** Re-activate the selected components.
- **Open network.** Open sub-networks in order to edit different hierarchical layers.

2.6 Simulation Toolbar

The simulation toolbar contains tools for running simulations, setting variables and editing parameter variation and path analysis settings.

- **Edit path.** Edit path analysis parameters. Refer to section "Path Analysis" for details.
- **Set path.** Adds the selected components to a path for a subsequent path analysis.
- **Delete path.** Remove the selected components from an analysis path.
- **Set pmd-path.** Adds the selected components to a pmd-path for a subsequent path analysis.
- **Delete pmd-path.** Remove the selected components from an analysis path.
- **Global variables.** Edit global variables used in parameter variations. Refer to section "Parameter Variation" for details.
- **Online variation.** Vary a previously defined global variable "on the fly" and immediately see the resulting influence on the system.
- **Simulation parameters.** See section "Simulation Parameters" for details.
- **Start.** Start a simulation or continue a paused simulation.
- **Start parameter variation.** Start a previously defined parameter variation.
- **Pause.** Suspend a running simulation.
- **Clear simulation.** Reset simulation related data.
- **Start batch mode.** Start the calculation of all open simulation files. The simulations are run in the order the corresponding files were opened.
- **Clear all simulations.** Reset simulation related data of all simulations running in batch mode.
- **Simulation status.** Display simulation status, progress and approximate end time.
- **Send e-mail to PHOTOSS team.** Send an e-mail to the PHOTOSS team. The current simulation file (.pho) can be attached (optional).
- **About.** Show the PHOTOSS version and copyright information.

2.7 Component Tree

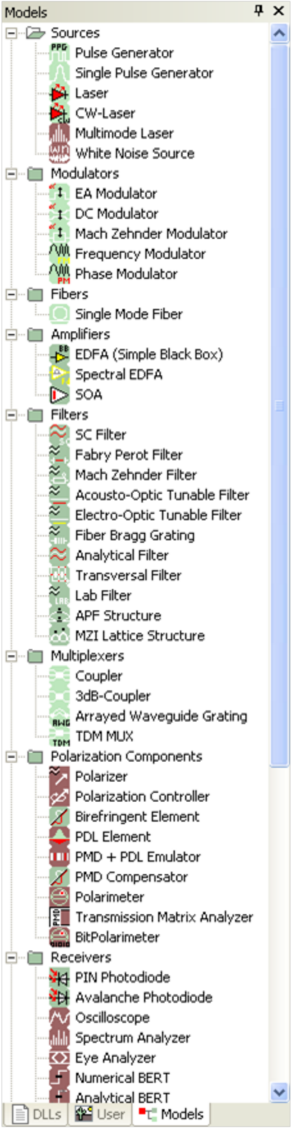


Figure 2.7.1. PHOTOSS Component Tree.

The component tree window is docked to the left edge of the main windows by default. It provides access to the available component models which can be dragged onto a component grid with the mouse. Three different views are available via the tabs at the bottom of the window: Models, User, and DLLs.

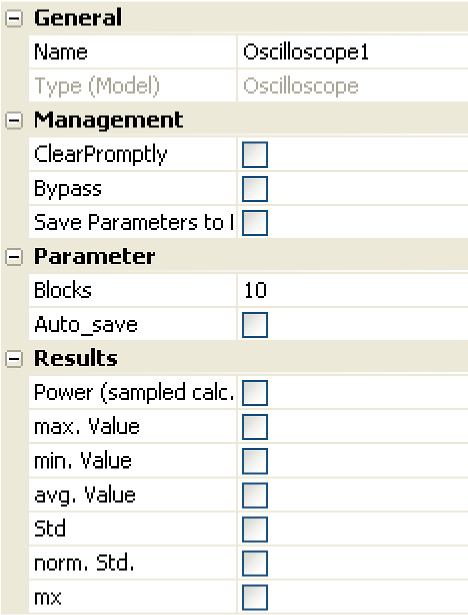
The Models view is a hierarchical list of all available component models in terms of the underlying simulation algorithm. Each model represents a specific algorithm or functionality with a corresponding set of customizable parameters.

When a model is dragged onto a component grid, a network component based on this model is created and initialized with default parameters. You may afterwards customize the parameter settings by double-clicking the component on the grid.

Clicking tree items with the right mouse button will open context menus that allow modifying the structure of the tree by creating, renaming or deleting folders or nested subfolders. You may also move the model icons to different folders by dragging them within the tree. However, you may neither delete nor duplicate models in this view. The User view provides easy access to pre-defined parameter sets for specific models. Choose from the standard entries or define your own network components with customized parameter values and place them in the user tree by dragging them onto the tree folder symbols with the mouse. When a parameter set is dragged onto a component grid, a network component based on the underlying model is created and initialized with the pre-defined parameters instead of its default settings. Of course you can still modify the parameters later on by double-clicking the component. In contrast to the Models view, the hierarchical structure of the user tree may be customized without restrictions.

The DLLs view displays all available component models (just like the Models view does), with the difference that the models are displayed in a static hierarchy of folders that represent the DLL (dynamic link library) files from which the corresponding models were imported. This way you can see which component DLLs have been imported and which component models are provided by these DLLs. You may drag models from the DLLs view onto a component grid like you can with the Models view, but you cannot modify the hierarchical structure of this view.

2.8 Property Grid and Result Grid



General	
Name	Oscilloscope1
Type (Model)	Oscilloscope
Management	
ClearPromptly	☐
Bypass	☐
Save Parameters to I	☐
Parameter	
Blocks	10
Auto_save	☐
Results	
Power (sampled calc.	☐
max. Value	☐
min. Value	☐
avg. Value	☐
Std	☐
norm. Std.	☐
mx	☐

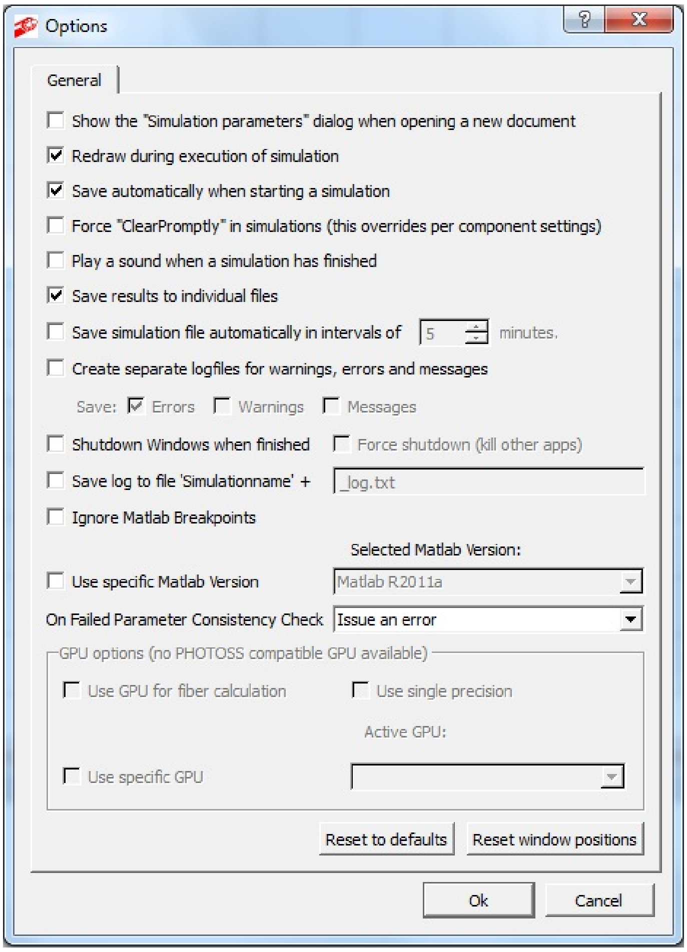
Figure 2.8.1. Component Property Grid

After left-clicking a component in the Component Grid its parameters are shown in the Property Grid and its results can be seen in the Result Grid. You may also edit the component's parameters using the Property Grid. The Property Grid works just like the Standard Component dialog that pops up, if you double-click a component. The Result Grid also works for links between components: left-click a link and the according parametric signal will show up in the Result Grid.

By default the Property Grid is located on the left side of the main window. The Result Grid can be found on the right side.

One may edit the parameters of more than one component simultaneously by selecting several components of the same type and change the parameters within the property grid. Components parameters with identical numerical values will show their values, component parameters with differing values will hide the numerical value.

2.9 Options Dialog



☐ Show the "Simulation parameters" dialog when opening a new document
☑ Redraw during execution of simulation
☑ Save automatically when starting a simulation
☐ Force "ClearPromptly" in simulations (this overrides per component settings)
☐ Play a sound when a simulation has finished
☑ Save results to individual files
☐ Save simulation file automatically in intervals of 5 minutes.
☐ Create separate logfiles for warnings, errors and messages
Save: ☑ Errors ☐ Warnings ☐ Messages
☐ Shutdown Windows when finished ☐ Force shutdown (kill other apps)
☐ Save log to file 'Simulationname' + _log.txt
☐ Ignore Matlab Breakpoints
Selected Matlab Version:
☐ Use specific Matlab Version Matlab R2011a
On Failed Parameter Consistency Check Issue an error
GPU options (no PHOTOSS compatible GPU available)
☐ Use GPU for fiber calculation ☐ Use single precision
Active GPU:
☐ Use specific GPU

Figure 2.9.1. Options Dialog

Select the *Options* entry from the *Tools* menu to show the *Options* dialog. Here you can change some preferences to simplify your work with PHOTOSS. These settings are saved separately for every PHOTOSS user on your computer.

The following options can be set:

- **Show the "Simulation parameters" dialog when opening a new document.** If this button is checked (default), PHOTOSS will open the Simulation parameters dialog each time you create a new simulation. Otherwise, the simulation parameters are set to default values.

- **Redraw during execution of simulation.** If this button is checked (default), PHOTOSS will redraw the component grid each time a new component has finished its calculation; progress markers will be shown on the grid. If unchecked, the grid will not be refreshed. Unchecking this option may result in significant decrease of simulation time when dealing with many short simulations that each include a large number of components.

- **Save automatically when starting a simulation.** If this button is checked (default), PHOTOSS will automatically save the current simulation when you start the simulation process. You will no longer be able to undo any changes.

- **Play a sound when a simulation has finished.** If this button is checked (not default), a sound will be played when a simulation run is complete.

- **Save results to individual files.** If this button is checked (default), PHOTOSS will save all active results to their own individual file in the simulations directory.

- **Create separate log files for warnings, errors and messages.** If this button is checked (not default), the log files containing warnings, errors and message will be saved to separate files. You can also specify which output you want to save.

- **Shutdown Windows when finished.** If this button is checked (not default), PHOTOSS will shutdown the operating system when a simulation finishes.

- **Force shutdown (kill other apps).** Some applications pause the shutdown process to ask the user whether to save unsaved changes for example. If this button is checked, PHOTOSS will force the operating system to shutdown regardless of other applications which may be crashed or pause the shutdown process deliberately. (Enabled if 'Shutdown Windows when finished' is checked.)

- **Save log to file "Simulationname+_log.txt".** If this button is checked, PHOTOSS will save the text which is displayed in the output window to a file which name is the name of the simulation, followed by a user defined extension.

- **Ignore MATLAB® Breakpoints.** If this button is checked (not default), breakpoints which have been set in PHOTOSS MATLAB® components will be ignored.

- **Use Specific MATLAB® Version.** This menu can be used for choosing another version of MATLAB® than currently active. Having multiple instances of PHOTOSS running, this feature will be deactivated because it sets global paths and registry entries. It might also affect other programs in their choice of the MATLAB® version to use. In this case, use PHOTOSS to set the specifically used MATLAB® version back to the newest available.

- **GPU Options.** These options allow for the selection of specific GPU devices to speed up the calculations of a nonlinear fiber component. See also section 3.25 on page 104.

- **Use GPU for fiber calculation.** If this option is enabled (and a compatible GPU is installed), the GPU will be used to carry out calculations for single mode fiber components. If the option is disabled, the single mode fiber will be computed on the CPU (default).

- **Use specific GPU.** If enabled, you can select a specific compatible GPU for the fiber calculation. This can be of interest if several compatible GPUs have been installed on the same machine.

- **Use single precision.** If this option is checked (not default), the GPU will be forced to use its single precision units for the calculation. This mode may be several times faster than the double precision mode (default). Please keep in mind that the single precision mode may result in less accurate results!

- **Reset window positions.** The button Reset window positions resizes and repositions all docking windows to their default size and position.

3 Basic Concepts

3.1 Simulation

3.1.1 Underlying Concept

The main purpose of the PHOTOSS application is to setup, modify and run numerical simulations to model the behavior of optical transmission systems. In this context, a simulation is comprised of a set of components with individual parameters which are connected to each other via "links". The simulation also contains a given set of built-in parameters to describe the physical model of an optical transmission system. The PHOTOSS application can be used to open a multiple number of simulations at the same time. In this case, each simulation has its own set of simulation parameters.

It is possible to carry out any number of simulations consecutively but only one simulation may be run at a time. It is, however, also possible to use multiple instances of the PHOTOSS application at the same time to carry out multiple parallel simulations.

Simulations are saved to simulation files in .pho (or .phox) format and can be loaded. Simulations which were created with older version of PHOTOSS can always be opened in newer versions of the application and compatibility is maintained. On the other hand, opening simulations which were created with a newer version in an older application may not be possible, since new components or simulation concepts may not have existed in the older version.

The graphical representation of a simulation is the so-called simulation grid which is shown in the main window. Each component has a position on the grid and simulation parameters of the simulation can be accessed using the simulation parameters dialog (see section 2).

3.1.2 Summary

- A simulation is comprised of a set of linked components and simulation parameters
- A simulation can be saved and loaded to/from a simulation file
- A simulation file has the default extension .pho (or .phox)
- A simulation file can be copied to and run on various machines
- A simulation may consist of multiple simulation runs (in case of a parameter variation) and/or multiple blocks
- Aside from the simulation file, additional files may be necessary for the execution of a simulation - e.g. input text files, PScripts, etc.
- There is no interaction between multiple simulations on a conceptual level; however, interaction can be constructed by using PScripts, File Loaders and Savers or Import/Export text files if necessary.
- PHOTOSS is downward compatible: Simulations created with older versions can always be opened with newer versions but not the other way round.

3.2 Examples

3.2.1 Underlying Concept

Your PHOTOSS installation always comes along with a large number of “examples”. “Examples” denote one or more simulation files with a prearranged set of components and parameter settings to demonstrate a certain functionality of the PHOTOSS application and/or some more complex components. They can be used to get a good overview on how to work with PHOTOSS and how to achieve corresponding simulation concepts and setups.

To access the example library, simply click on the “Open an example” button in the GUI to open the “Open Example” dialog (see also page 15). Now you can browse though the example folders and open a simulation of interest.

Some examples will assume that MATLAB® has been installed on your machine - they can be recognized by the “MATLAB” prefix in their folder name. You will also find PScript examples (also called *code samples*) in your example folder. PScript examples can also be directly opened from the PScript console (see the PScript manual for further details).

3.2.2 Summary

- The PHOTOSS installation includes example setups to help you quickly understand simulation concepts
- Examples can be accessed via the “Open an example” button in the GUI
- PScript examples are also included in the PHOTOSS installation

3.3 Simulation Parameters

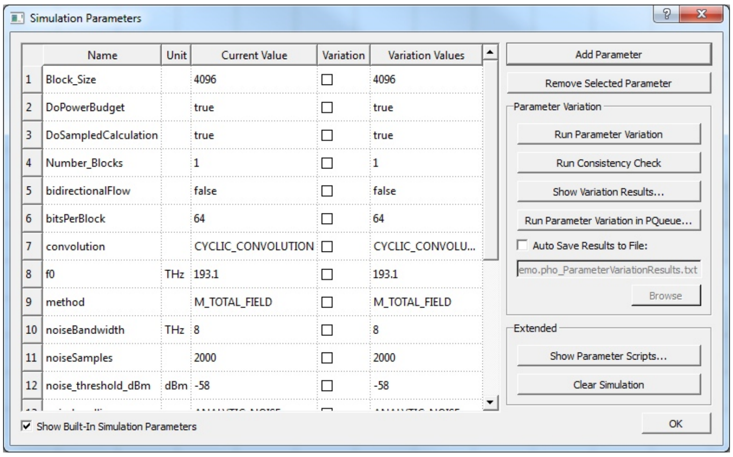
3.3.1 Underlying Concept

Each PHOTOSS simulation has its own, individual set of simulation parameters. This set can be divided into two groups:

1. Built-in simulation parameters
2. Custom simulation parameters

Built-in simulation parameters are used to describe the physical parameters for modeling optical and electrical signals for the current simulation. These parameters for example concern the simulation band in the frequency or time domain. Some built-in simulation parameters also represent a choice for a specific simulation model - e.g. the modeling of noise which may be either numerical or analytical (see section 3.9).

The built-in parameters are always present in a simulation - they can not be deleted or created. Access to built-in simulation parameters can be gained either through a specialized dialog (see fig. 3.3.2) or the overview dialog by clicking on the checkbox “Show built-in simulation parameters” (see fig. 3.3.1).



	Name	Unit	Current Value	Variation	Variation Values
1	Block_Size		4096	□	4096
2	DoPowerBudget		true	□	true
3	DoSampledCalculation		true	□	true
4	Number_Blocks		1	□	1
5	bidirectionalFlow		false	□	false
6	bitsPerBlock		64	□	64
7	convolution		CYCLIC_CONVOLUTION	□	CYCLIC_CONVOLU...
8	f0	THz	193.1	□	193.1
9	method		M_TOTAL_FIELD	□	M_TOTAL_FIELD
10	noiseBandwidth	THz	8	□	8
11	noiseSamples		2000	□	2000
12	noise_threshold_dBm	dBm	-58	□	-58

☑ Show Built-In Simulation Parameters

□ Auto Save Results to File: emo.pho_ParameterVariationResults.txt

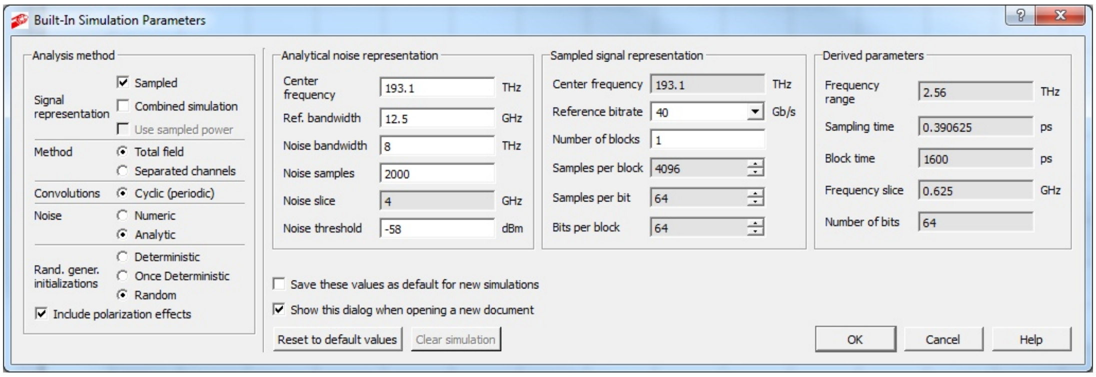
Figure 3.3.1. Overview dialog for both custom and built-in simulation parameters. This dialog shows both the built-in and custom simulation parameters (if created).

Custom simulation parameters can be created and deleted by the user and can also be accessed by the overview dialog (fig 3.3.1).

Both types of simulation parameters have the following attributes:

- Name: The name of the simulation parameter. Must be unique.
- Unit: Unit of the simulation parameter; a unit is not mandatory.
- Current Value: The current value of the simulation parameter. Must be a valid value for a built-in simulation parameter or placeholder. May also be a formula.
- Variation: If checked, this parameter is being treated as part of a parameter variation (see section 3.13).
- Variation Values: Vector which specifies a list of all values a parameter takes during a parameter variation.

3.3.2 List of Built-In Simulation Parameters



Analysis method

Signal representation: ☑ Sampled ☐ Combined simulation ☐ Use sampled power

Method: ● Total field ○ Separated channels

Convolutions: ● Cyclic (periodic)

Noise: ○ Numeric ● Analytic

Rand. gener. initializations: ○ Deterministic ○ Once Deterministic ● Random

☑ Include polarization effects

Analytical noise representation

Center frequency	193.1	THz
Ref. bandwidth	12.5	GHz
Noise bandwidth	8	THz
Noise samples	2000	
Noise slice	4	GHz
Noise threshold	-58	dBm

Sampled signal representation

Center frequency	193.1	THz
Reference bitrate	40	Gb/s
Number of blocks	1	
Samples per block	4096	
Samples per bit	64	
Bits per block	64	

Derived parameters

Frequency range	2.56	THz
Sampling time	0.390625	ps
Block time	1600	ps
Frequency slice	0.625	GHz
Number of bits	64	

☐ Save these values as default for new simulations

☑ Show this dialog when opening a new document

Figure 3.3.2. Dialog for the built-in simulation parameters. It shows up automatically when creating a new simulation.

This section lists all built-in simulation parameters of PHOTOSS. You will also find much more detailed information on the underlying concepts of the signal representation, noise modeling and simulation methods in the following sections. The simulation parameters are:

- Signal representation:
  - Sampled: If checked, a sampled run will be executed (default). If unchecked, only the parameterized run will be executed. Also see section 3.10.
  - Combined simulation: This is an option to save memory in one-block-simulations (see section 3.10 and 3.24). This option cannot be enabled in simulations with more than one block.
  - Use sampled power: If checked, PHOTOSS will calculate the power of the signal from the sampled signal representation and not from the less accurate parameterized signal representation. Only available if a “Combined simulation” has been selected, unchecked by default.
- Method (see section 3.8):
  - Total field: Use total field mode. Only one simulation band will be simulated.
  - Separated channels: Use SC mode. A simulation band will be separately simulated for each signal channel.
- Convolutions (see section 3.15):
  - Cyclic (periodic): Convolutions (e.g. for FIR filtering) are carried out cyclically; not linearly.
- Noise (see section 3.9):
  - Numeric: Noise modeling is carried out numerically. Numerical BERTs have to be used.
  - Analytic: Noise modeling is carried out analytically. Analytical BERTs and analytical noise built-in simulation parameters (see below) have to be used.
- Random generator initializations (see section 3.21):
  - Deterministic: PHOTOSS will use a fixed seed for the random number generator each time a simulation is run but *not* each time a block is processed in a multi-block simulation or each time a run of a parameter variation is carried out.
  - Once Deterministic: PHOTOSS will use a fixed seed for *every* simulation run of a parameter variation.

  - Statistical: PHOTOSS will use a random seed each time a simulation is run.
  - randomGeneratorSeed: Determines the seed for the deterministic random generator mode. Must be set to a negative number.

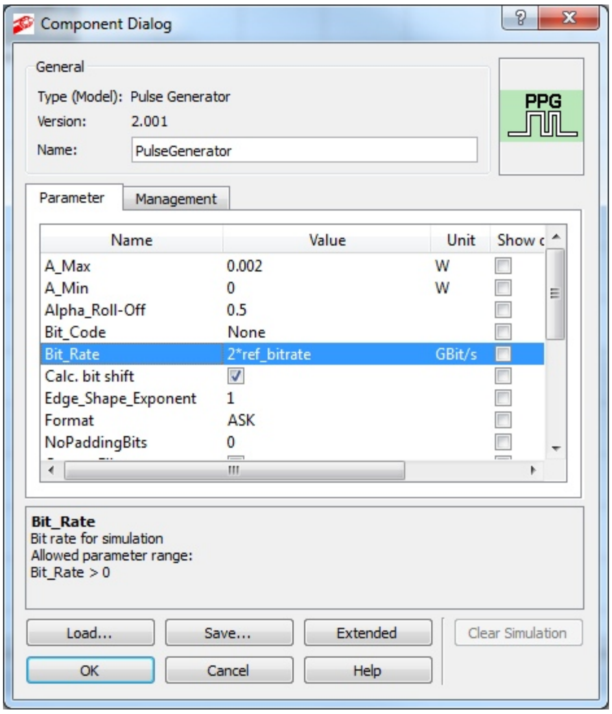
- Include polarization effects:
  - Unchecked (default): The sampled signal is only calculated for one polarization axis; polarization effects like polarization mode dispersion (PMD) and polarization-dependent loss are not calculated.
  - Checked: The sampled signal is calculated for the x- and y-polarization axis. Polarization effects can be calculated.

- Analytical noise representation (see section 3.9):
  - Center frequency: Center frequency of the reference noise simulation band.
  - Ref. bandwidth: Reference bandwidth of the noise simulation band.
  - Noise bandwidth: total bandwidth of the noise simulation band.
  - Noise samples: Number of processed noise samples inside the noise simulation band.
  - Noise slice: Size of frequency interval for each noise sample in the noise simulation band.
  - Noise threshold: Minimum power level for which a signal is still treated to be as a WDM signal. If the signal power drops below the noise threshold, the signal will be considered to be “noise only” and not be processed.

- Sampled signal representation (see section 3.8):
  - Center frequency: Center frequency of the simulation band in total field mode.
  - Reference bitrate: Reference bitrate for the simulation band; influences the derived parameters “Frequency range”, etc.
  - Number of blocks: Number of blocks which will be executed when the simulation is run. Each block holds “Samples per block” samples and “Bits per block” bits.
  - Samples per block: Number of samples per block. Equals “Samples per bit” times “Bits per block”.
  - Samples per bit: Number of samples per bit.
  - Bits per block: Number of bits per block

- Derived parameters (see section 3.8): These parameters can not be changed directly. Their values are derived from the parameters ”Reference bitrate”, ”Samples per block”, “Samples per bit” and “Bits per block”.
  - Frequency range: Size of the simulation band in the frequency domain.
  - Sampling time: Duration of one signal sample in the time domain.
  - Block time: Duration of one block in the time domain.
  - Frequency slice: Size of one signal sample in the frequency domain.
  - Number of bits: Total number of bits per simulation run. Equals “Bits per Block” times “Number of blocks”.

3.3.3 Simulation Parameters as Placeholders



General
Type (Model): Pulse Generator
Version: 2.001
Name: PulseGenerator

Name	Value	Unit	Show c
A_Max	0.002	W	□
A_Min	0	W	□
Alpha_Roll-Off	0.5		□
Bit_Code	None		□
Bit_Rate	2*ref_bitrate	GBit/s	□
Calc. bit shift	☑		□
Edge_Shape_Exponent	1		□
Format	ASK		□
NoPaddingBits	0		□

Bit_Rate
Bit rate for simulation
Allowed parameter range:
Bit_Rate > 0

Figure 3.3.3. The simulation parameter “ref_bitrate” is used as placeholder in a formula for the parameter “bitrate” in the Pulse Generator.

All simulation parameters of a simulation may act as a sort of “global variables”. This means that they can be accessed from any other component or (sub-)network of the same simulation and - of course - by other simulation parameters. Thus, all simulation parameters can be used as placeholders for any component parameter. They may also be part of a formula for each component parameter (see figure 3.3.3). The value which is actually used for the component parameter when the simulation is executed is normally the “current value” of the simulation parameter. However, during the execution of a parameter variation (see section 3.13), this is the current *variation* value of the simulation parameter of the current variation (sub-)run.

3.3.4 Summary

- Simulation Parameters work on a global scope for each simulation and all components but not between multiple simulations
- Simulation Parameters can be divided in built-in and custom Simulation Parameters
- Built-in Simulation Parameters can only be modified - they can not be deleted or created
- Built-in Simulation Parameters define the physical model which is used to simulate the optical transmission system. They affect the simulation band, noise modeling, convolutions, etc.
- Some built-in Simulation Parameters are read-only. These parameter are also called “Derived Parameters” and are always calculated automatically depending on the values of other Simulation Parameters
- Custom Simulation Parameters can be created, modified and deleted

- Simulation Parameters may be used as placeholders for any component (or sub-component) parameter or in a component parameter formula
- Simulation Parameters may also be used as placeholders in formulas for other Simulation Parameters
- Simulation Parameters are stored in signal files when using file savers
- Simulation Parameters cannot be changed during the a simulation run but they can be varied in a Parameter Variation

3.4 Components

3.4.1 Underlying Concept

In the PHOTOSS application, components are used as “black-box model representations” for real-life components of optical transmission systems such as e.g. a silica fiber, an Erbium-Doped Fiber Amplifier (EDFA) or an Eye Analyzer. These models are considered to be “black boxes” because they are not transparent - their inner workings can only be accessed through a set of component-specific parameters and input or output signals.

Normally, each PHOTOSS component is comprised of any number of input ports (also called “in-ports”) and output ports (“out-ports”) and a set of individual parameters (see section 3.5). During a single simulation run, component parameters may not be changed. Additionally, each component parameter may only assume a valid value inside a defined parameter range of the component.

If more than one component is present in a simulation (which is often the case), components are connected to each other via component links (see section 3.6). As a rule of thumb, components in a simulation can only interact with each other by modifying the signal which propagates through their links. There are, however some exceptions from this rule: Components which can be PMD path members may also draw on the PMD path construct which is not part of the simulation parameters or the signal structure (see section 3.22 for further details).

Some components also (internally) make use of the built-in simulation parameters of the simulation. Naturally, not every set of simulation parameters is appropriate for the usage of every component. For example, an Analytical BERT receiver may not be used while numerical noise modeling has been activated in the simulation parameters. In these cases, PHOTOSS will automatically issue a warning or an error if the simulation parameter setting collides with the component settings so you can adjust either value(s) to meet the requirements.

3.4.2 Creating and Handling Components

PHOTOSS Components can be created by dragging them from the “Component Tree” (see section 2.2) onto the grid: Click it with the left mouse button, hold down the button while you move the mouse onto the grid, and release the button to place the component at the current position. The new network component will appear as depicted below (the example shows a *Single Mode Fiber* component).

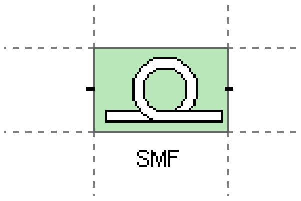


Figure 3.4.1. Component “SMF” on the PHOTOSS grid.

Once you have placed one or more components on the grid, you may:

- **Select a single component:** Just click it with the left mouse button.
- **Select multiple components:** Click the left mouse button in the free space beneath a component and hold it down while you move the mouse. A rectangular selection region is drawn. When you release the button, all components in the rectangle are selected.

- **Select additional components:** Click them while holding down the *Control (Ctrl) key*. They are added to the current selection without de-selecting other components.
- **Move components:** Click a component with the left mouse button, hold down the button and move the mouse. Release the button to move the component to the current position. When multiple components are selected and you move one of them, all other selected components are moved accordingly. Note that no component may be placed on top of another one. If you do, the moved component will automatically be placed on a free space nearby.
- **Delete components:** Press the *Delete (Del) key* to remove the selected component(s) from the grid.
- **Perform cut, copy, and paste operations:** Use the *Cut*, *Copy*, and *Paste* entries in the Edit menu to cut (or copy) selected components to the clipboard and paste them back from the clipboard. A "cut & paste" operation effectively moves the components to where they are pasted while removing them from their origin, while "copy & paste" creates new copies at the destination without deleting the respective sources. Use these operations e.g. to transfer components between different grids when you are working with multiple networks or sub-networks, or use "copy & paste" to create duplicates of the selected components on the same grid. You may use the key commands *Ctrl+X* (cut), *Ctrl+C* (copy), and *Ctrl+V* (paste) instead of the corresponding menu entries.

3.4.3 Component Names

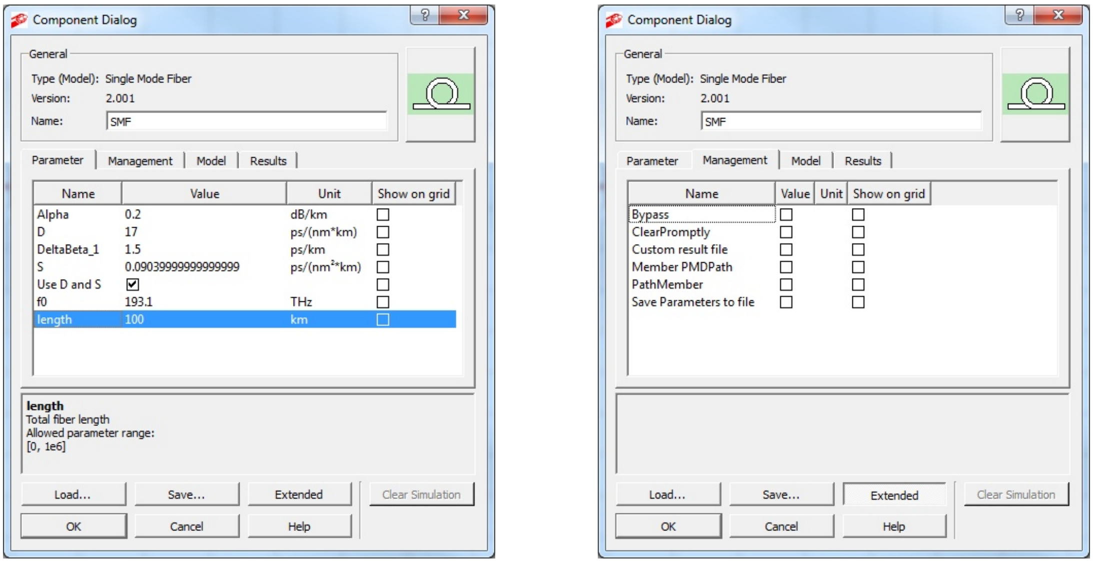
A component is fully characterized by its type (e.g. "Single Mode Fiber") and its name (e.g. "SMF"). The name of a component may be changed in the component dialog (see below). However, two components on the same network level (see section 3.18) may never share the same name - even if they are of different component types. Trying to assign a name to a component which is already in use will result in an error.

When creating a component by dragging it from the model tree on the grid or by copying an existing component, the name of the component is automatically modified to ensure that its name is unique for each network level. For example, if you drag a component of type "Single Mode Fiber" on the grid for the first time, its name will be "SMF". Dragging a second component onto the grid will assign the name "SMF0" to it and dragging further components on the grid will automatically increase the numerical suffix in its name by one.

It should also be noted that components on *different* network levels are allowed to have the *same* name. This is possible, because internally components are addressed using their full path - including the names of the mother network(s) and may be of interest when using PScript to address components from different networks (see the PScript manual for further details on this topic).

3.4.4 Component Parameter Tabs

To change the parameter(s) of a component, double click on the desired component on the simulation grid to open the component dialog (see fig. 3.4.2) and click on the "Parameter" tab. This tab generally refers to the physical parameters of a component. Normally, only the most relevant parameters for each tab are shown in the dialog. There are, however, some component parameters which are of interest to the advanced PHOTOSS user. These parameters are called "Extended Parameters" and can be displayed by clicking on the "Extended" button of the dialog. If you are interested in a detailed information for a component parameter, simply click on the parameter and take a look at the bottom of the dialog (see left in fig. 3.4.2). There, a description of the parameter and its allowed parameter range are shown. If you need further information about the parameter or the inner workings itself, click "Help" to open the manual entry for the currently selected component.



Type (Model): Single Mode Fiber
Version: 2.001
Name: SMF

Name	Value	Unit	Show on grid
Alpha	0.2	dB/km	□
D	17	ps/(nm*km)	□
DeltaBeta_1	1.5	ps/km	□
S	0.09039999999999999	ps/(nm^2*km)	□
Use D and S	☑		□
f0	193.1	THz	□
length	100	km	□

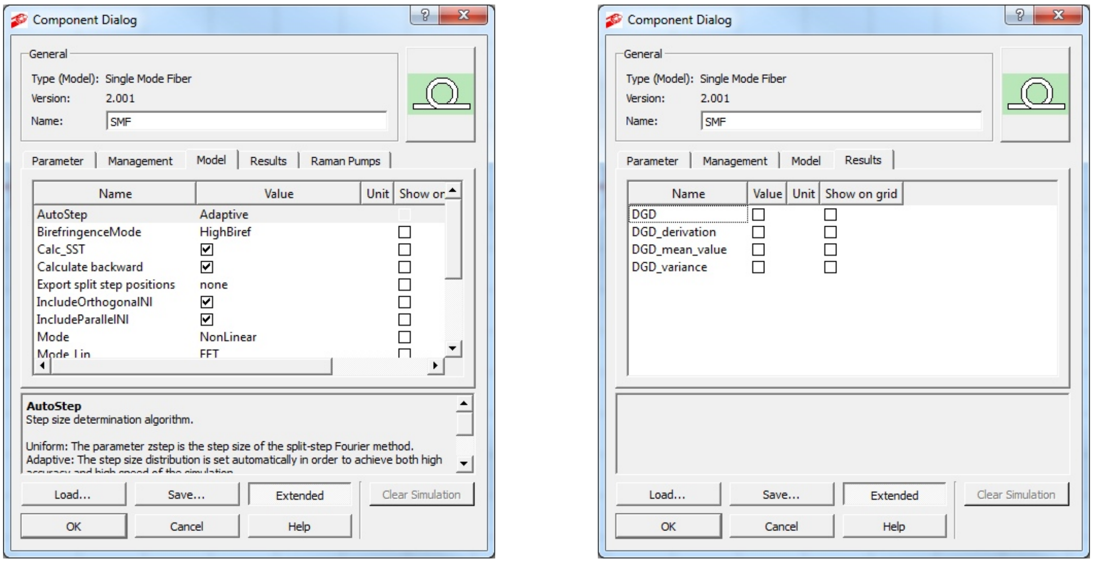
length
Total fiber length
Allowed parameter range:
[0, 1e6]

Type (Model): Single Mode Fiber
Version: 2.001
Name: SMF

Name	Value	Unit	Show on grid
Bypass	□		□
ClearPromptly	□		□
Custom result file	□		□
Member PMDPath	□		□
PathMember	□		□
Save Parameters to file	□		□

Figure 3.4.2. Component dialog for a component of type “Single Mode Fiber”. Left: Parameters section. Right: Management section.

All components also offer some settings which are related to the simulation environment itself and not to its physical representation. These parameters can be found by clicking on the “Management” tab (right in figure 3.4.2). A detailed description of these parameters which applies to all components can also be found in section 3.5.



Type (Model): Single Mode Fiber
Version: 2.001
Name: SMF

Name	Value	Unit	Show or
AutoStep	Adaptive		□
BirefringenceMode	HighBiref		□
Calc_SST	☑		□
Calculate backward	☑		□
Export split step positions	none		□
IncludeOrthogonalNI	☑		□
IncludeParallelNI	☑		□
Mode	NonLinear		□
Mode_Lin	FFT		□

AutoStep
Step size determination algorithm.

Uniform: The parameter zstep is the step size of the split-step Fourier method.
Adaptive: The step size distribution is set automatically in order to achieve both high accuracy and high speed of the simulation

Type (Model): Single Mode Fiber
Version: 2.001
Name: SMF

Name	Value	Unit	Show on grid
DGD	□		□
DGD_derivation	□		□
DGD_mean_value	□		□
DGD_variance	□		□

Figure 3.4.3. Left: Special tab for physical fiber modeling. Right: Result tab.

Some components may have special groups of parameters which have their own tabs in the component dialog - for example the “Single Mode Fiber” component (see fig. 3.4.3, left). Also shown in figure 3.4.3 on the right side is the result tab. Many PHOTOSS components can be used to calculate specific results which are available to the user after the simulation has finished. More details about component results can be found in section 3.16.

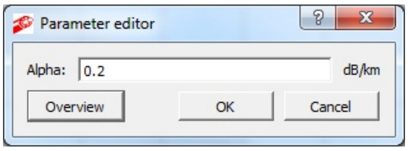
All component parameters may also be saved to or loaded from a text file. This can be useful when you want store specific, complex parameter settings and if you want to ensure that these are identical for different simulations. Just click on “Load” or “Save” to load or save the component parameters.

If the option “Show on grid” has been selected, the specific parameter value is also visible on the PHOTOSS grid next to the component.

3.4.5 Changing Component Parameters

When a parameter is selected in a component parameter tab, a short description appears in a tooltip window to the upper right of the dialog. The actual value of a parameter can be edited either in-place by directly clicking it in the *Value* column or by double-clicking the parameter which will open a separate edit window. The following basic parameter types are used:

1. **Numeric values** of type *integer* or *double* (floating point numbers) are directly entered into the corresponding edit field. Instead of numeric values, *double* parameters may also be assigned a mathematical expression (formula) that involve mathematical constants, functions, or global variables (see section *Parameter Variation*). Use the *Overview* button to display available variables, constants, and functions in a separate window.

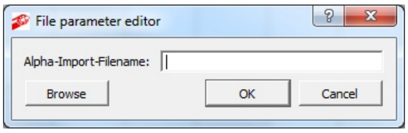


Alpha: 0.2 dB/km

Figure 3.4.4. Numerical Parameter dialog

2. **Enumerations and Boolean** (true/false) parameters are chosen from a dropdown listbox that contains all available choices.

3. **String or File** parameters are entered directly (like numeric values). In addition, you may use *Parse*{*x*} once or more in the string, where *x* stands for a mathematical expression that results in a numeric value to be inserted into the string. Like for numeric parameters, the expression may involve mathematical constants, functions, or simulation parameters (see section Parameter Variation on page 71).



Alpha-Import-Filename:

Figure 3.4.5. File save dialog. The Browse button will open a file selector dialog which allows picking an existing filename as the entered string value.

The *Load* and *Save* buttons at the bottom of the component dialog serve to read the component’s parameters from a file on the disk or write the current parameters to a file. When loading parameter files, please make sure the file was written by a component of the same type.

The *Extended* button will show (or hide) advanced parameters that are not shown in standard mode. Extended parameters offer full control over the modeling of the component, but typically require more background knowledge and should be chosen with care.

Once a new value has been entered for a component parameter, a consistency check will automatically be carried out to ensure that the new value is valid for the component parameter. If the value is not valid, it will be rejected (default) and the user will be informed.

3.4.6 Component Ports

Components are connected at their respective input and output ports. On the component grid, input ports are displayed as small black rectangles on the left edge of a component while output ports are located on the right edge. Depending on the type or settings of a component, there may be none, one, or multiple input or output ports.

The ports may be of different types depending on which kind of signal is delivered by an output port (resp. expected by an input port). PHOTOSS deals with three types of signals:

- **Optical amplitude signals** (defined as the square root of the optical power),
- **electrical voltage signals**, and
- **electrical current signals**.

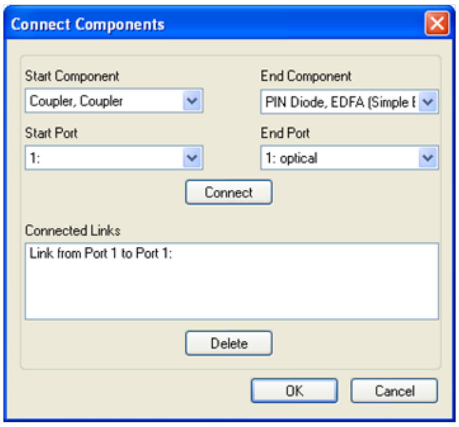
3.4.7 Linking Components

As explained above, PHOTOSS components are usually connected to each other by using “Links” (see section 3.6). There are three ways to link network components on the component grid:

1. **Manual connection:** First click the source component’s output port, then click the destination component’s input port. The new link is immediately created and displayed. When the mouse is moved over a port, the port will be highlighted to indicate its “hit area”. Manual connection is an intuitive way to connect single components with few ports.

2. **Connect multiple components:** Select all components to connect, then click the ⇉ toolbar button or the Create/Connect selected components menu entry or simply press the hot key “CTRL + F”. The components will all be connected automatically based on the following rules:
   - The components are connected in the order they are located on the grid (top left to bottom right): first line left to first line right, then second line
   - Subsequent components are connected port by port: output #1 to input #1, output #2 to input #2,
   - The number and types (electrical or optical) of each component’s output ports must match the number and types of the subsequent component’s input ports. Otherwise the corresponding link will not be created.

3. **Connect tool:** Select two or more components, then click the ↦ toolbar button or the *Create/ Connect-Dialog* menu entry to open the *Connect Components dialog*.



Start Component: Coupler, Coupler

End Component: PIN Diode, EDFA (Simple E

Start Port: 1:

End Port: 1: optical

Connected Links: Link from Port 1 to Port 1:

Figure 3.4.6. The connect tool can be used to connect multiple components at once.

Select the start and end component and the respective output and input port from the drop-down list controls, then click *Connect* to establish a corresponding link. You may define several links in the above manner before you leave the dialog. The *Connected Links* list is automatically updated in accordance with the selected components. Finally confirm your settings with *OK* (or discard them with *Cancel*).

The connect tool is useful for components with a large number of ports that cannot be graphically separated on the grid.

The automatic connection tool is a quick way to automatically connect multiple components at once as long as the above conditions are met.

3.4.8 Unlinking Components

Existing links can be selected with the mouse by left-clicking in them and removed with the *Delete* button. If two or more components are selected, you can also use the “Disconnect multiple components” button or use the hot key “CTRL + D”. Deleting a component also automatically deletes its connection links as well.

3.4.9 Bypass Components

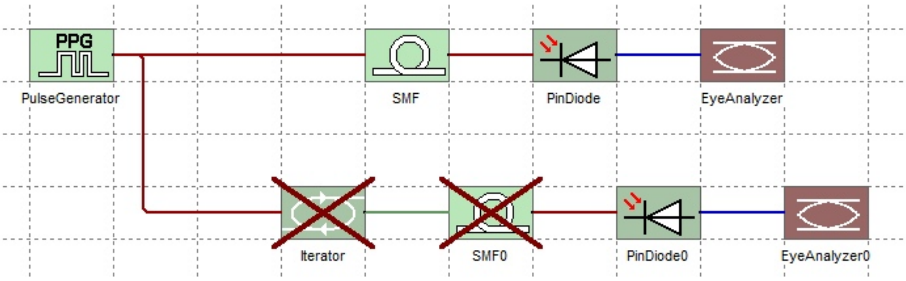


Figure 3.4.7. In this simulation, components “Iterator” and “SMF0” are being bypassed. They will not modify the signal and will not be processed.

Sometimes, it may be interesting to run the same simulation twice - for example a setup for a short reach optical transmission system with and without dispersion compensation using a dispersion compensating fiber (DCF). Of course, it would be possible to simply delete the DCF component in the second

run, but a much easier way to still retain your component (and its parameters) in the simulation is to *bypass* the component (fig. 3.4.7).
Bypassed components are completely ignored in the simulation run - they do not influence the PHOTOSS signal or load from/save to files. No internal calculations of the components are executed. Thus, bypassed components can never produce results or include visualizer data.

Some PHOTOSS components can not be bypassed: This applies to all components which act as multiplexers (e.g. the Coupler) or as signal sources (e.g. the Pulse Generator or a CW Laser).

In order to bypass / remove bypass of a component there are several ways:

1. Select component(s) and use the hot key “F11” to bypass, “F12” to remove bypass
2. Select component, right-click on it and select “Bypass component” from the context menu
3. Select component(s) and use the “Bypass” or “Remove Bypass” buttons
4. Open component dialog, switch to tab “Management” and check or uncheck the “Bypass” parameter

Keep in mind that some components - e.g. all signal sources such as the Pulse Generator and all multiplexers such as Coupler - can *not* be bypassed.

3.4.10 Types of Components

Aside from “normal” black box type components such as the “Single Mode Fiber” or the ”Eye Analyzer” there are also two special types of components which have unique abilities. These are the Network Component (see 3.18) and the Iterator Component (see 3.19). Both will be described in more detail in their own sections.

3.4.11 Creating Custom Components

Creating your own custom components and using them within the PHOTOSS application is also possible - e.g. by using the MATLAB® Interface. This will be described in detail in section 4.7 on page 120.

3.4.12 Summary

- Components contain parameters which determine their functionality
- Each component on the same network level has its own unique name
- Components interact with each other via simulation links
- Components may have (multiple) in- and out-ports
- Any component out-port may be the beginning of multiple outgoing links
- A component in-port may only be the end of one incoming link
- Components may support or require different sets of chosen Simulation Parameters
- Components may produce results
- Components can be created, copied and deleted
- Custom components can be realized by using the MATLAB® Interface

- Iterators and Networks are special components with unique abilities (see below)
- Components can save their parameters to a text format file
- Components can save their results to a text format file
- Components may not be modified or moved during a simulation run

3.5 Parameters

3.5.1 Underlying Concept

PHOTOSS uses *simulation parameters* to configure the simulation environment (see section 3.3) and *component parameters* to modify the settings of all components (see section 3.4); component parameters are often abbreviated as "parameters". Most component and simulation parameters are hard-coded - they can not be created or deleted. It is however possible to create custom simulation parameters (also see section 3.3) and some components like the network or iterator also offer the creation of custom parameters (see section 3.18 and 3.19).

3.5.2 Parameter Scopes

A parameter must apply to one of the following scopes:

1. Global. Simulation parameters can be accessed from everywhere in the simulation and be used as placeholders.

2. Local (including subnetworks). A custom parameter of a network or an iterator may be used as placeholder in all components residing inside the network/iterator on a lower hierarchy level.

3. Component. Most parameters are only used inside a black box component and can not be accessed from outside the component. They can not be placeholders.

3.5.3 Parameter Attributes

In the most simple way, a PHOTOSS parameter is comprised of the following attributes:

1. Name: The name of a parameter must be unique in its scope. Thus, one component must never have two parameters with the same name and two simulation parameters must also never share the same name.

2. Value: The value of a parameter can be of the following data types: `integer`, `bool`, `double`, `string` and `formula` (see below). Trying to enter parameter values of the wrong data type will result in an error.

3. Unit: A parameter may have a unit (optional). Units are *not* used internally for automatic conversion, etc. and can only be assigned to custom parameters.

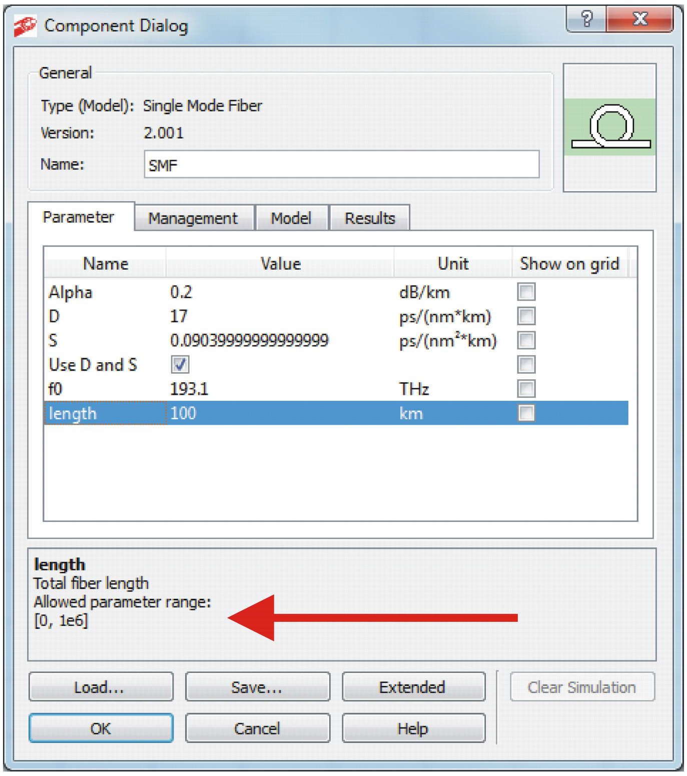
4. Variation / Variation values: Simulation parameters may also be part of a parameter variation (see section 3.3 and 3.13). In this case, the variation flag of that parameter is checked and all parameter values of the variation are stored in the "Variation values" vector.

3.5.4 Changing Parameters

Parameters can be changed before a simulation is executed. Once the simulation is started ("running"), parameters can not be changed until the simulation has finished. After a simulation has finished, it needs to be cleared to change a parameter - in that case, the results are reset to always ensure a consistency between the entered set of parameters and the displayed results. Thus, once you change the parameters of a simulation which has been finished, you need to re-run the simulation to obtain the results for your new parameter set. Changing simulation parameters can be done by accessing the simulation parameter dialog (see section 3.3); changing component parameters can be done by opening the component dialog by double clicking left on the desired component (see 3.4).

3.5.5 Parameter ranges and parameter consistency checks

Each component parameter always has a clearly defined parameter range which specifies the valid parameter values. This parameter range can be seen at the bottom of the component parameter description when left-clicking once on the parameter you are interested in. Each time you try to change a parameter, a parameter consistency check will be executed automatically. It does not matter whether you change the parameter using the GUI or a *PScript* command like *setParameterValue()*.



General

Type (Model): Single Mode Fiber

Version: 2.001

Name: SMF

Name	Value	Unit	Show on grid
Alpha	0.2	dB/km	□
D	17	ps/(nm*km)	□
S	0.09039999999999999	ps/(nm²*km)	□
Use D and S	☑		□
f0	193.1	THz	□
length	100	km	□

length
Total fiber length
Allowed parameter range:
[0, 1e6]

Figure 3.5.1. Allowed parameter ranges are displayed on the bottom of the parameter description area. The length of an SMF component may not be smaller than 0 km or larger than 10^6 km.

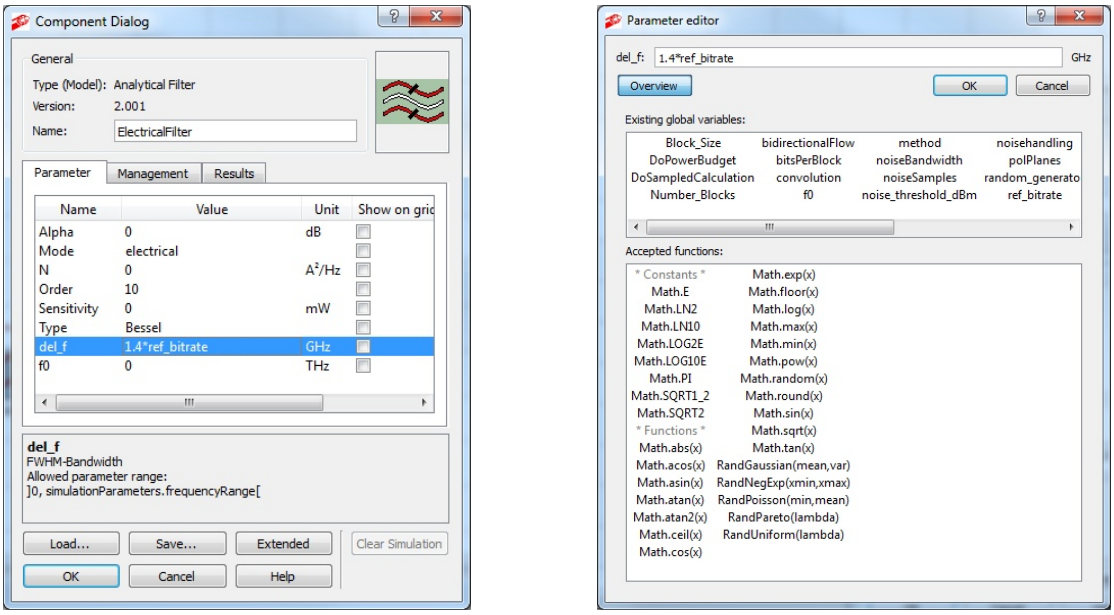
The parameter checker is responsible for evaluating the value you have supplied. Its response depends on the corresponding PHOTOSS option *On Failed Parameter Consistency Check* in the PHOTOSS *option dialog*. Its default behavior is to reject values which are outside the allowed parameter range and to reset the parameter to the last valid value instead while issuing a warning in the PHOTOSS console.

The parameter consistency checker ensures that all PHOTOSS components will always operate on a set of parameters which are consistent in a *general way* (e.g. the length of a fiber cannot be negative, etc.). The parameter consistency checker does *not* evaluate, whether the settings and values are plausible, since this evaluation is dependent on the current simulation setup and *cannot be done automatically*.

3.5.6 Formulas as Parameter Value

Instead of a current value, a component or simulation parameter may also consist of a formula. This can be very helpful when a component parameter is dependent on e.g. a simulation parameter. For example, the bandwidth of an electrical filter in relation to the reference bit rate of the simulation may be expressed as

$$\Delta f_{filter} = 1.4 \cdot \text{ref\_bitrate}. \tag{1}$$



General
Type (Model): Analytical Filter
Version: 2.001
Name: ElectricalFilter

Name	Value	Unit	Show on gric
Alpha	0	dB	□
Mode	electrical		□
N	0	A^2/Hz	□
Order	10		□
Sensitivity	0	mW	□
Type	Bessel		□
del_f	1.4*ref_bitrate	GHz	□
f0	0	THz	□

del_f
FWHM-Bandwidth
Allowed parameter range:
]0, simulationParameters.frequencyRange[

del_f: 1.4*ref_bitrate GHz

Existing global variables:

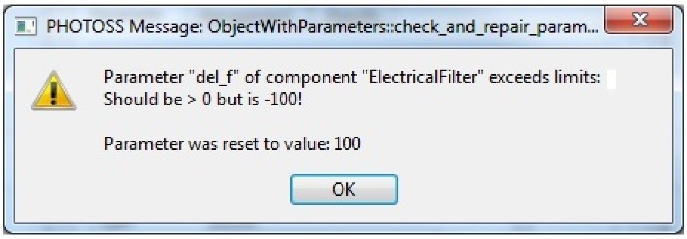
Block_Size	bidirectionalFlow	method	noisehandling
DoPowerBudget	bitsPerBlock	noiseBandwidth	polPlanes
DoSampledCalculation	convolution	noiseSamples	random_generato
Number_Blocks	f0	noise_threshold_dBm	ref_bitrate

Accepted functions:

* Constants *	Math.exp(x)
Math.E	Math.floor(x)
Math.LN2	Math.log(x)
Math.LN10	Math.max(x)
Math.LOG2E	Math.min(x)
Math.LOG10E	Math.pow(x)
Math.PI	Math.random(x)
Math.SQRT1_2	Math.round(x)
Math.SQRT2	Math.sin(x)
* Functions *	Math.sqrt(x)
Math.abs(x)	Math.tan(x)
Math.acos(x)	RandGaussian(mean,var)
Math.asin(x)	RandNegExp(xmin,xmax)
Math.atan(x)	RandPoisson(min,mean)
Math.atan2(x)	RandPareto(lambda)
Math.ceil(x)	RandUniform(lambda)
Math.cos(x)	

Figure 3.5.2. Component dialog for a component of type “Electrical Filter”. Left: Parameters section. Right: Parameter Editor.

The left side of figure 3.5.2 shows how this expression can be inserted into the parameter del_f. If you want to get an overview of all placeholders which are available in the scope of your component parameter, simply double-click on the parameter you wish to change to open the “Parameter editor” and select “Overview”. The dialog on the right of figure 3.5.2 will appear and show all available parameters - double click on any parameter name to add it to the formula. Additionally, a list of all available hard-coded functions is displayed - double click on a function to add it to your formula.



PHOTOSS Message: ObjectWithParameters::check_and_repair_param...

Parameter "del_f" of component "ElectricalFilter" exceeds limits:
Should be > 0 but is -100!

Parameter was reset to value: 100

Figure 3.5.3. If the expression for a component value would result in an invalid value, the parameter consistency checker will alert the user and reject the formula.

Whenever possible, PHOTOSS tries to evaluate the expression of a given formula prior to a simulation run to execute a parameter consistency check (see above). If - for example - you try to enter the formula

$$\Delta f_{filter} = -1 \cdot 100. \qquad (2)$$

for the bandwidth of the electrical filter, the error message of figure 3.5.3 will be displayed and the invalid formula will be rejected. However, some formulas can only be evaluated at runtime. This applies e.g. to a formula which uses a function that calls up random numbers. If - at runtime - these functions would produce invalid values for a component parameter, the default behavior of PHOTOSS is that the parameter consistency check will fail, the simulation will be stopped and an error will be displayed.

3.5.7 Extended Parameters and Derived Parameters

Some components - for example the single mode fiber - have a very large number of parameters and not all parameters are always shown in the GUI for reasons of simplicity. To show and change extended parameters, simply open the component dialog and left-click on the “Extended” button (see section 3.4).

Additionally, a few components contain parameters which are mutually dependent on each other. For example, changing one or two Stokes parameters “s1” and “s2” of the component “Polarization Controller” influences the third Stokes parameter “s3”. It will automatically be adjusted to always maintain a consistent form together with the other parameters. Using these parameters in a parameter variation requires special care: Each set of the parameter variation must carry a valid value for *each* of the mutually dependent parameters - if this is not the case, the parameters will be automatically adjusted.

3.5.8 Summary

- Parameters apply to the scope of a simulation, components on lower hierarchy levels or a component
- Each component parameter and built-in simulation parameter has a range of allowed values
- A consistency check is carried out each time a parameter is changed
- A parameter definition consists of a parameter name, value and unit
- Parameters may consist of a (current) value, a formula, a placeholder (e.g. a simulation parameter) or a combination of all elements.
- Formula expressions are evaluated and checked prior to a simulation run if possible. If not possible, they are checked at runtime.
- An overview of available formula functions and placeholders can be shown in the GUI
- Parameters may be of the type `bool`, `integer`, `double`, `string/filename` and the selection from a combo box.
- Parameters can not be changed during a single simulation run
- Parameters can be varied for multiple simulation runs using a Parameter Variation
- Parameters can be divided in “basic” and “extended” parameters; only the former are always shown in the GUI
- Some parameters are only relevant when a specific operating mode of a component is chosen
- Parameters which are non-relevant to the currently chosen operating mode of the component are not shown in the GUI
- Component parameters may be shown below the component on the grid in the GUI
- The current values of the component parameters may be saved to text format files

3.6 Links

3.6.1 Underlying Concept

In the PHOTOSS application, components are connected to each other by “links”. A signal is propagated through the links from left to right. These links are not representations of actual physical components and only illustrate the signal flow in PHOTOSS.

3.6.2 Link Types

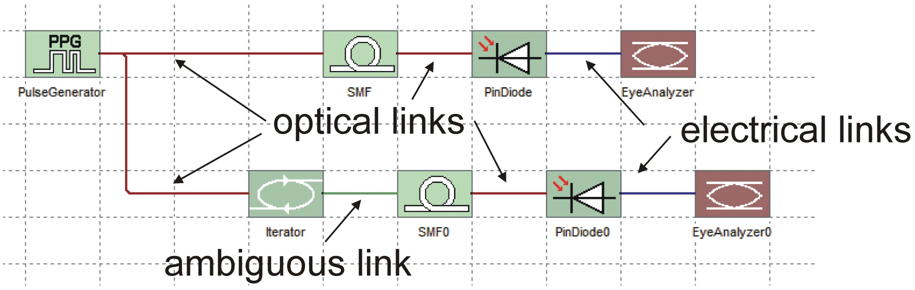


Figure 3.6.1. Different link colors represent different signal domains.

The links between connected ports are depicted as colored lines. Each color represents a different signal domain (see figure 3.6.1):

- red links mark optical connections,
- blue links mark electrical connections, and
- green links mark connections that can either be optical or electrical.

The latter case can occur with multi-purpose components that can deal with both optical or electrical signals (e. g. the calculator component, a network or an iterator).

3.6.3 Creating and Deleting Links

Creating and deleting links is described in detail in section 3.4.

3.6.4 Links and Ports

Most PHOTOSS components have at least one in- and one out-port. If you wish to execute a simulation run, the in-ports of *all* components in the simulation must be connected; empty in-ports are not allowed. The connection of an out-port is always optional.
A component may receive only one incoming link per in-port. The number of out-going links at an out-port is not limited. In order to “combine” the output signals of two or more components, multiplexers such as the Coupler, the 3dB-Coupler or the Arrayed Waveguide Grating have to be used.

3.6.5 Accessing Signal Information through a Link

Once a simulation run has finished, the signal information is stored in all simulation links as a default. This information contains both the parameterized signal and the sampled signal (see section 3.7). To access this information, you may right-click on a link to call up a context menu (fig. 3.6.2). You access the parameterized signal information by clicking on the “Channel data” entry. You can also

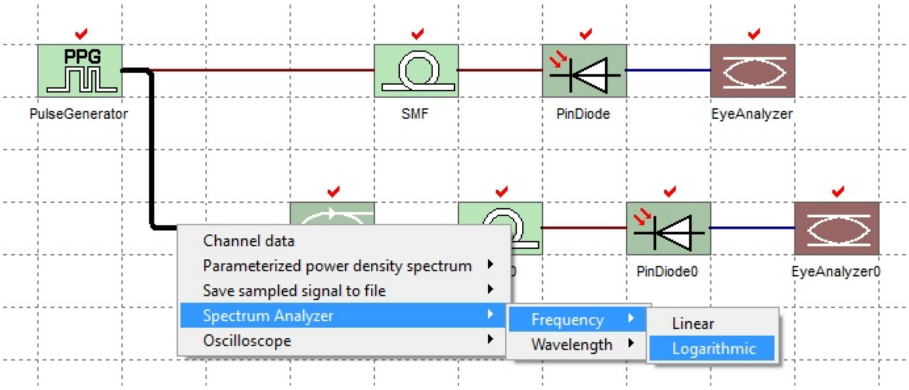


Figure 3.6.2. Right-clicking on a link will bring up a context menu where you can access signal information.

use Visualizers (see section 3.17) like the Oscilloscope or the Spectrum Analyzer to plot the sampled signal. Additionally, the sampled signal may be saved to a file (this works the same way as if you were using a “File Saver” component after the link).

3.6.6 Summary

- Links connect PHOTOSS components with each other
- A PHOTOSS signal is propagated through the links
- Links do not represent physical “links” and only illustrate the signal flow
- Links hold the full signal information (parameterized and sampled signal) after the simulation has finished
- The signal information of the links can be viewed in the GUI (e.g. by using a Visualizer)
- The signal information can also automatically be cleared after the calculation to save memory
- Links can be created manually (Shortcut CTRL+F) or by the Connect Dialog in the GUI
- Links can not be modified, created or deleted during a simulation run
- Different link colors represent different signal domains
- Red link colors represent optical signals
- Blue link colors represent electrical signals
- Green link colors may be either optical or electrical and are ambiguous

3.7 Signals

3.7.1 Underlying Concept

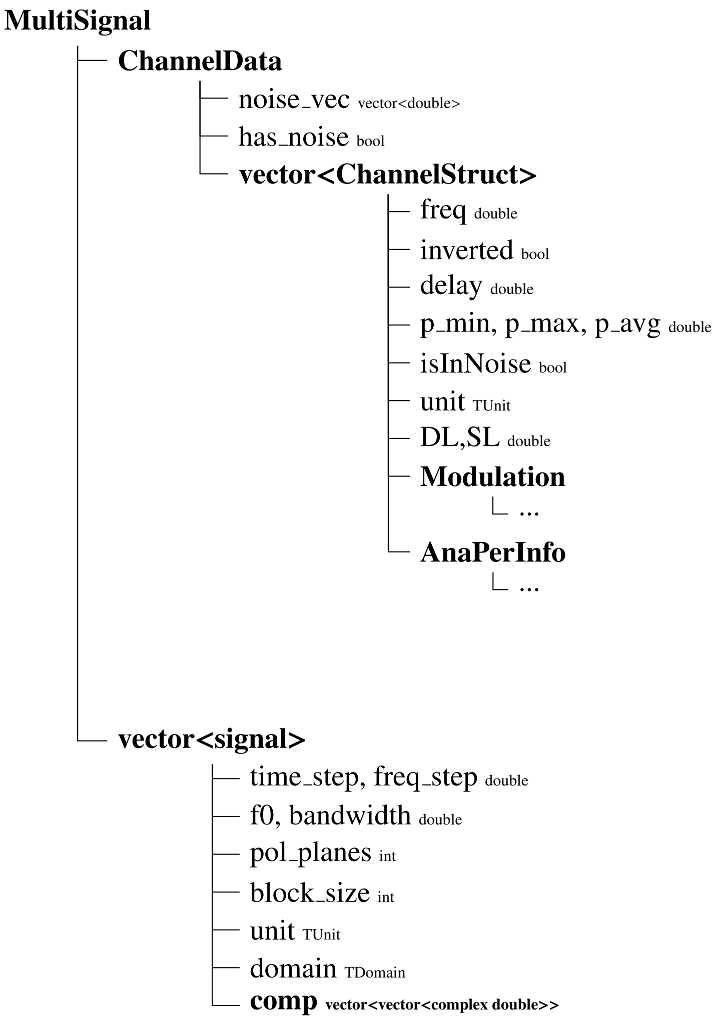


Figure 3.7.1. MultiSignal structure

PHOTOSS uses a “MultiSignal” to propagate information through a simulation setup. Depending on the choice of simulation parameters, the “MultiSignal” structure may include various elements and substructures. In general, two substructures can be distinguished which are also closely related to the simulation runs (see section 3.10). They are:

1. the “channeldata” - which is calculated in the parameterized signal run

2. the “(sampled) signal” - which is calculated in the sampled signal run

The “MultiSignal” structure and its subcomponents are depicted in figure 3.7.1. The subcomponents will be described in the subsections below. Only one MultiSignal can be propagated *per PHOTOSS link*.

3.7.2 The Parameterized Signal (Channeldata)

This signal structure is closely connected with the parameterized simulation run (see section 3.10) and will only be calculated and modified during this run. Its contents, however, are also used in the sampled run. Each MultiSignal contains only one channeldata. The channeldata contains mostly scalar parameters which describe the signal which are divided into further substructures / elements:

- **noise_vec**: This vector holds the noise samples for the analytical noise model - see section 3.9 for more information. This vector is present for both the analytical and the numerical noise modeling mode but will only be used in the analytical noise modeling mode.
- **has_noise:** Boolean value which is set to *true* once the signal has propagated through a component which adds noise to the signal - e.g. an EDFA.
- **vector<ChannelStruct>:** This substructure contains the channel information for each of the *n* simulated parameterized channels and is explained in section 3.8.5 in more detail.

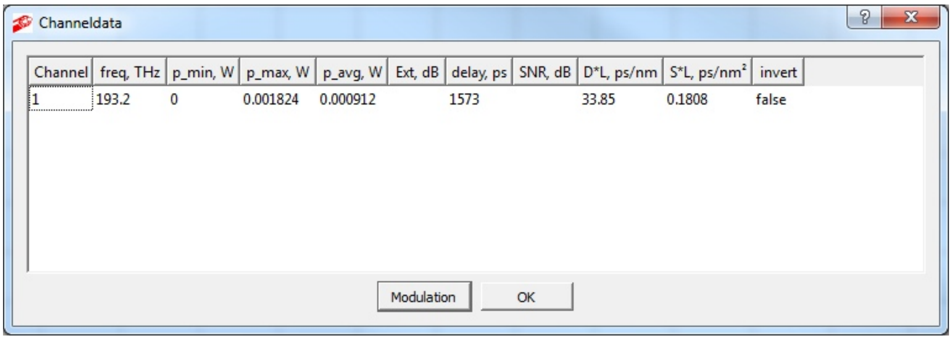
3.7.3 The Sampled Signal (Signal)

This signal structure is closely connected with the sampled signal run and - most importantly - contains the complex sampled signal information for the x- and (optionally) the y-polarization axis. the "MultiSignal" contains a signal structure for *each* of the *n* channels in the separated channels simulation mode. In the total field simulation mode, only one signal structure is present.

The structure can be further divided into:

- **unit:** The unit of the sampled signal. May be either "Watt", "sqrt(Watt)", "Volt" or "Ampere".
- **domain:** The domain of the signal. May be either "time" (default) or "frequency".
- **f_0:** The center frequency of the signal channel. This frequency is always equal to the center frequency of the built-in simulation parameters in total field simulation mode. In separated channels simulation mode it is equal to the frequency of the individual (separate) channel and will be set by the optical or electrical source of the designated WDM channel.
- **pol_planes:** The number of polarization planes in the signal. May be either 1 or 2 (if polarization effects are included in the simulation).
- **comp:** Contains the complex sampled signal vector for the x-polarization. When polarization effects are included in the simulation, the complex sampled signal vector for the y-polarization will also be included.

3.7.4 Accessing Channeldata and Sampled Signal

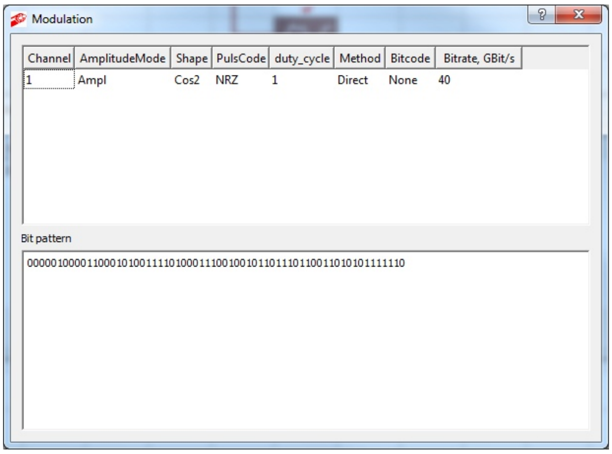


Channel	freq, THz	p_min, W	p_max, W	p_avg, W	Ext, dB	delay, ps	SNR, dB	D*L, ps/nm	S*L, ps/nm^2	invert
1	193.2	0	0.001824	0.000912		1573		33.85	0.1808	false

Figure 3.7.2. Channel Data Overview.

After a simulation run is finished, you can access both the channeldata and the sampled signal for each link in the simulation by right clicking on it (figure 3.6.2 on page 45). You can then select visualizers to display the sampled signal or take a detailed look at each component of the channeldata. More information can be found in section 3.6 and 3.17. If you select the entry "Channel data" from the menu, the contents of the parameterized signal will be displayed (figure 3.7.2).

This overview provides access to the parametric signal description at the current position in the network. Various stationary state properties are listed for each WDM channel: Carrier frequency, minimum and maximum power levels, extinction ratio, channel delay, signal-to-noise ratio (SNR), accumulated dispersion (dispersion · fiber length, $D \cdot L$), accumulated dispersion slope (slope · length, $S \cdot L$), whether the original bit sequence is currently inverted or not, and polarization elevation angle (θ). Clicking on the "Modulation" button will open another sub-window (figure 3.7.3).



Channel	AmplitudeMode	Shape	PulsCode	duty_cycle	Method	Bitcode	Bitrate, GBit/s
1	Ampl	Cos2	NRZ	1	Direct	None	40

Bit pattern

0000010000110001010011110100011100100101101110110011010101111110

Figure 3.7.3. Overview of the modulation structure.

This window lists information about the modulation format of each channel: Pulse shape, pulse code, duty cycle, bit code, bitrate, and bit pattern. The *AmplitudeMode* column shows if the pulse shape applies to the power (squared amplitude), amplitude, or square root of the amplitude of the signal. The *Method* column is reserved for future use (it will currently always be *Direct*). Please also refer to section 3.12 for a more detailed explanation about how modulation formats are implemented in PHOTOSS.

3.7.5 Signal Units

Currently, PHOTOSS can model electrical and optical signals. When using electrical signals, their center frequencies are considered to be zero as the electrical signal is described in the base band.

3.7.6 Summary

- Signals are also denoted as "MultiSignals"
- A component propagates one "MultiSignal" per link
- Each "MultiSignal" contains one "Channeldata" structure and n "Signal" structures where n denotes the number of channels (1 in Total Field mode and n in Separated Channels mode)
- The "Channeldata" contains the parameterized signal components and can be subdivided into further sub-components:
  - a noise vector carrying the analytical noise samples (also present when using numerical noise)

  - *n* “channel” structures containing the data for each signal channel. This structure can be subdivided in further substructures.

- The “Signal” structure contains the sampled signal components and can be subdivided as follows:
  - unit: signal unit of current channel (sqrt(Watt), Watt, Volt or Ampere)
  - domain: domain of signal (either time or frequency)
  - f_0: Center frequency of current channel. In total field mode, f_0 is identical to f_0 of the simulation parameters. In separated channels mode, f_0 is the center frequency of the current (separated) channel.
  - pol_planes: The number of polarization planes of the complex sampled signal.
  - comp: Matrix structure containing the complex signal samples for each sample in the time (or frequency) domain. Contains the signal information for the x- and y-polarization axis (if “include polarization effects” has been activated in the simulation parameters)
- PHOTOSS “Multisignals” can be saved and loaded using the File Saver and File Loader components

3.8 Simulation Band and Channels

3.8.1 Underlying Concept

As the PHOTOSS application aims to model the real physical behavior of optical transmission systems, certain assumptions have to be made. One concept is that the complex electrical field vector $\vec{E}(t)$ of an optical (or electrical) signal can be represented by a "sampled" signal (see also 3.7). In an ideal world with no limits on computational efficiency or memory constraints it would be possible to express a sampled signal over the whole spectrum in the frequency domain with infinitesimally small sampling intervals and - conversely - for infinite time in the time domain with an an unlimited sampling resolution.

In reality, however, simulating an infinitely densely sampled signal is not possible. Thus, a few restrictions have to be taken into consideration to define the properties of a sampled signal which *can* be simulated and is as close to the behavior of a real, physical signal as possible. These properties of a sampled signal define the PHOTOSS "simulation band" and are represented by built-in simulation parameters (see 3.3).

3.8.2 Sampled Signal Simulation Band

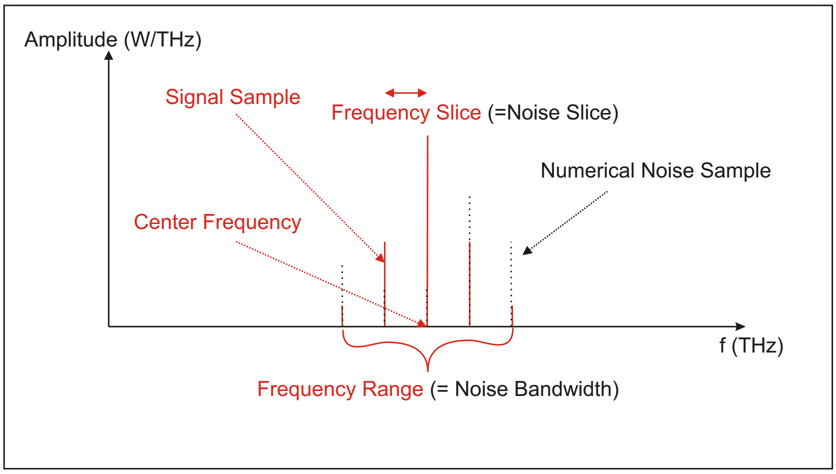


Figure 3.8.1. Sampled Signal and numerical noise representation.

PHOTOSS assumes that a sampled optical or electrical signal can be defined by supplying built-in simulation parameters. Figure 3.8.1 shows how these parameters make up a simulation band:

- **Center Frequency**: The simulation band is assumed to be centered around a certain frequency f_0. When using WDM systems with multiple signals at different carrier frequency, this center frequency should be close to or equal to the carrier frequency of your central WDM channel - e.g. 193.1 THz. In *Separated channels mode* the center frequency of a sampled channel vector is determined by its carrier frequency, thus the center frequency is disabled in the simulation parameters dialog.
- **Samples per Bit** (S_{Bits})**:** Defines how many sampling points are simulated for each bit. This number has to be set high enough to account for distortion effects like Intersymbol Interference (ISI) which influence a pulse shape on time scales shorter than the bit duration. Normally, sampling only twice per bit is not enough to account for these distortion effects. 16 to 128 samples per bit are usually a good place to start with. Due to the use of efficient Fast Fourier Transform (FFT) algorithms, N has to be a power of two.

- **Samples per block (N):** This parameter is automatically derived by setting the number of bits per block and samples per bit. The larger the number of samples per block, the higher the computational effort to process the simulation; however, multi block simulations might help to keep that effort small (see also 3.11). Due to the use of efficient Fast Fourier Transform (FFT) algorithms, N has to be a power of two.
- **Bits per block (N_{Bits}):** The number of bits per signal block. This value also has to be a power of two.
- **Number of blocks (B):** The number of blocks which are simulated after another. Each block uses the same simulation band.
- **Reference bitrate (f_{Bit}):** The reference bitrate in Gb/s is used to calculate exact sampling duration in the time domain and the frequency resolution in the frequency domain for a given setting of samples per bit and bits per block. The derived parameters (see below) are automatically calculated by entering a reference bitrate value. Typical bitrates are provided in the attached preset listbox of the simulation parameters dialog (however, arbitrary values may be typed into this field). Please note, if you desire to perform mixed bitrate simulations you have to enter at this point the bitrate of the lower bitrate channel.

⚠ **Note that the reference bitrate does *not* necessarily have to be equal to the bitrate(s) which you actually use in your signal sources. They have to satisfy the condition $bitrate_{PPG} >= bitrate_{reference}$, though. It is also possible to use different bitrates for different WDM channels as long as this condition is satisfied. When the bitrate of all channels is identical it is prudent to adjust the reference bitrate to this value.**

Once these parameters have been specified, PHOTOSS will automatically compute the so-called "derived simulation parameters". These parameters cannot be set directly:

- **Frequency range (F):** According to the given reference bitrate and given samples per bit, the frequency range can be computed using: $bitrate_{reference} \cdot samplesPerBit$. This parameter defines the width of the sampled signal in the frequency domain in THz.
- **Frequency slice (Δf):** This parameter defines the frequency resolution of the sampled signal in the frequency domain in GHz. It is calculated using $frequencyRange/samplesPerBlock$.
- **Block time (T):** The duration for a simulated block in the time domain in ps - calculated by $samplesPerBlock/frequencyRange$.
- **Sampling time (Δt):** The temporal resolution for a signal sample in the time domain in ps. Calculated using $1/frequencyRange$.
- **Number of Bits (B_{Bits}):** The total number of simulated bits. It is derived by evaluating $bitsPerBlock \cdot numberOfBlocks$.

When numerical noise modeling (see section 3.9) is used, all noise samples are automatically included in the sampled signal - noise and sampled signal band are identical.

3.8.3 Analytical Noise Simulation Band

PHOTOSS also offers the option to use analytical noise modeling instead of numerical noise modeling (see section 3.9). The main difference between these two approaches when considering the simulation band is that the analytical noise model needs an *additional* simulation band for the noise vector in the parameterized signal and this simulation band is normally *not* identical with the sampled signal simulation band as depicted in figure 3.8.2.

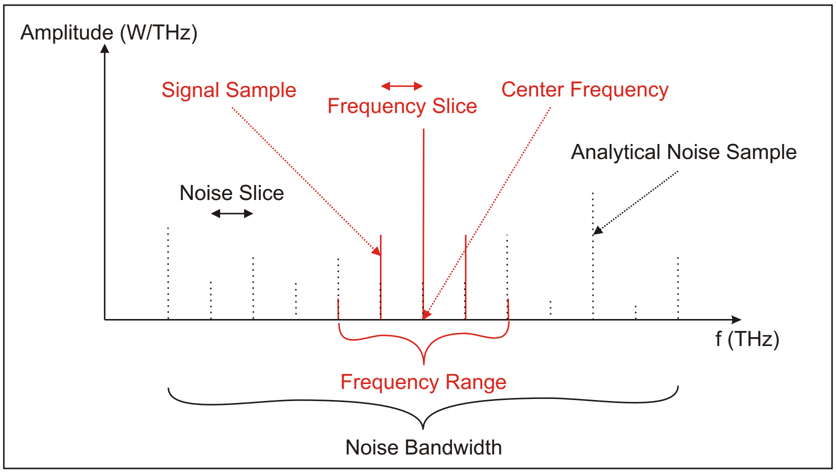


Figure 3.8.2. Analytical noise representation.

When analytical noise modeling is enabled, the sampled signal simulation band is supplemented with an additional noise simulation band - those two simulation bands are normally *not* identical.

The analytical noise band can be characterized by the following parameters:

- **Center frequency:** Center frequency around which the noise PSD vector is distributed. It should also be the carrier frequency of your center WDM channel and is always identical to the center frequency of the sampled signal representation.

- **Reference bandwidth:** The bandwidth in which noise is measured.

- **Noise bandwidth:** The total bandwidth of the noise PSD vector. The bandwidth is distributed around the *Center frequency*. The noise bandwidth should always be slightly larger than the frequency range of the sampled signal.

- **Noise samples:** The number of samples used in the noise PSD vector (determines the bandwidth of each noise sample, see *Noise slice*).

- **Noise slice:** The bandwidth of a single noise sample (= *Noise bandwidth* / *Noise samples*). The noise slice does not have to be equal to the frequency slice of the sampled signal.

- **Noise threshold:** Specifies the minimum power level of a WDM channel to be recognized as a valid signal. Channels are removed from the signal description, either if their power drops below this threshold or if it drops below the noise power at the channel's carrier frequency. In either case, the power of a removed channel is added to the noise vector.

Presently, there is only one analytical noise sample vector for both polarization axes. This does not effect the simulation of systems with PMD or with only one polarization axis. However, simulations with PDL or combined PMD and PDL effects can *not* be carried out with the analytical noise model - please switch to the numerical noise model instead. For more information on PDL please refer to the component PDL Element on page 273 and to the component PMD + PDL Emulator on page 261.

3.8.4 Total Field and Separated Channels Mode

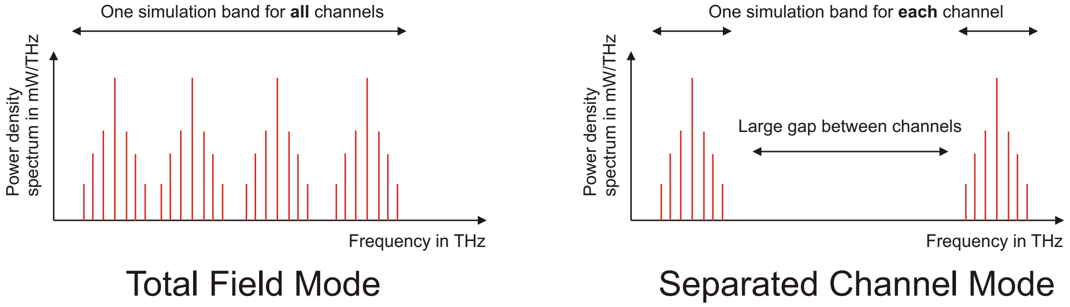

One simulation band for **all** channels
Power density spectrum in mW/THz
Frequency in THz
Total Field Mode
One simulation band for **each** channel
Large gap between channels
Power density spectrum in mW/THz
Frequency in THz
Separated Channel Mode

Figure 3.8.3. Total field vs. Separated Channels approach.

PHOTOSS offers two distinct modes for the simulation of WDM channels:

1. Total Field Mode (default)
2. Separated Channels Mode

The Total Field analysis method is the default when using PHOTOSS simulations. All WDM channels of an optical signal are stored within a single sampled signal simulation band. This allows for an easy integration of inter-channel effects in physical component models. In terms of the “signal” substructure of the MultiSignal (see 3.7), only one “signal” entry exists in the signal vector.

The Total Field mode should be used when inter-channel effects have to be calculated and the size of the simulation band for all WDM channels is not too large - e.g. there are no large gaps between the WDM channels in the simulation bands (left hand side of figure 3.8.3).

Sometimes, WDM channels are less densely packed and the gaps between them in the frequency domain become wider. Generally, the Total Field approach could still be used but it would not be a very economical way since large parts of the sampled signal have to be processed which actually do not contain any relevant signal data because the electrical field would be “zero” (aside from noise components when using numerical noise modeling). In these cases, the Separated Channels approach may be the better choice: Here, different WDM channels are stored in separate simulation bands and thus separate “channels” - multiple “signal” entries exist in the signal vector.

Additionally, the calculation of inter-channel effects can be switched on or off independently in the single mode fiber component for the Separated Channels mode. Thus, an isolated analysis of single (nonlinear) effects in optical fibers is possible.

3.8.5 Channels

PHOTOSS distinguishes between two types of “channels”:

1. channels in the parameterized signal
2. channels in the sampled signal

In the *parameterized* simulation run, the term “channel” denotes *something different*: Here, a channel is a set of individual, parameterized scalar information values but not complex sampled signals. Each channel has an individual denoted frequency, delay, power levels and modulation. The following components are part of a “ChannelStruct” in the parameterized signal (compare fig. 3.8.4):

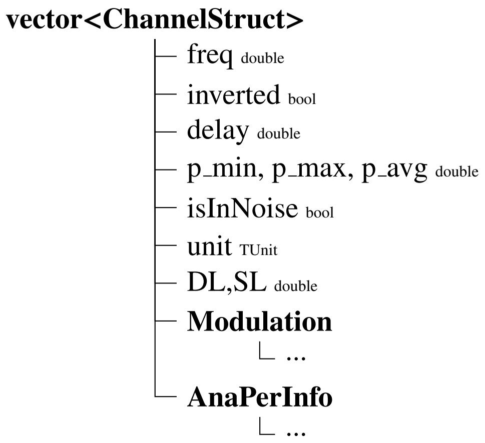


Figure 3.8.4. The parameterized ChannelStruct

- **freq:** the center frequency for each parameterized WDM channel. This frequency is identical to the center frequency of the used modulation source (e.g. the Pulse Generator) for the distinct channel
- **inverted**: Has the signal been inverted (in that case, p_{min} and p_{max} have been switched)?
- **delay:** delay of current channel in ps
- **p_min, p_max, p_avg:** minimum, maximum and average signal power of current channel
- **isInNoise**: Has the channel signal power fallen below the noise threshold? In this case, the channel is considered to be “noise only” and it is assumed that its signal information has been lost.
- **unit:** signal unit of current channel (sqrt(Watt), Watt, Volt or Ampere)
- **D·L and S ·L:** accumulated dispersion and dispersion slope of current channel
- **Modulation:** Structure containing data for the description of the modulation format (see also 3.12).
- **AnaPerInfo:** Structure containing data for the analytical performance evaluation and for analytical noise modeling. Is also present (but unused) when numerical noise is simulated (see also 3.9).

In the *sampled* signal run, the MultiSignal holds a vector of *n* sampled signals where each “signal” corresponds to a *channel* (a better term would be “simulation band”). In Total Field mode (see above), only one channel (one simulation band) is present, while several channels (simulation bands) may be present in the Separated Channels mode. So – in a nutshell – the term “channel” also denotes the number of simulation bands and thus sampled signals for the sampled simulation run.

3.8.6 Summary

- PHOTOSS distinguishes between parameterized channels and sampled channels
- A parameterized channel is a sub-component of a Multi Signal and the “channel data” and holds the following components (also see section 3.7):
  - Information regarding the analytical noise model
  - Information regarding the signal modulation
  - Information regarding the parameterized signal representation

- A sampled channel holds the sampled data for each distinct WDM channel
- The simulation bandwidth and the sampling resolution of the sampled channels are Simulation Parameters
- Electrical or optical signal behavior can only be modeled *inside* the Simulation bandwidth
- The appropriate choice for the corresponding Simulation Parameter is dependent on the type of system (e.g. multiple channels, bit rate, etc.)
- PHOTOSS offers a “Total Field” and “Separated Channels” bandwidth mode
- For simulation bandwidths with small gaps between WDM channels, the “Total Field” approach is appropriate
- For simulation bandwidths with large gaps between WDM channels, the “Separated Channels” approach is appropriate
- “Separated Channels” can also be used to switch on or off effects between distinct channels in the nonlinear single mode fiber.
- Each transmitter component will generate its own signal parameterized channel. In the “Separated Channels” model, it will also generate its own sampled channel.

3.9 Noise Models

3.9.1 Underlying Concept

To accurately model the behavior of an optical transmission system, noise has to be incorporated into the simulation. Transmission components may add noise to deterministic signals (such as ASE noise generated by optical amplifiers). The generation of noise represents a random, statistical process. Noise is normally assumed to be additive, white Gaussian noise ("AWGN"). Generally, PHOTOSS offers two different simulation modes for noise modeling:

1. Analytical Noise modeling
2. Numerical Noise modeling

You can select the mode of your choice by changing the simulation parameter "noisehandling", also known as "Noise" in the simulation parameter dialog (see section 3.3).

In a nutshell, the main difference between these two models is that the analytical noise model makes certain assumptions about the modulation format and uses information of optical and electrical filters in the simulation to derive information for noise modeling in a *separate* analytical noise vector. The numerical noise model, in contrast, adds noise components directly to the sampled signal and does not impose any restrictions on the modulation format. Thus, both models differ in terms of the usage of the simulation band. This can be understood by looking at figures 3.8.2 and 3.8.1 on page 52 and 50.

3.9.2 Analytical Noise Modeling

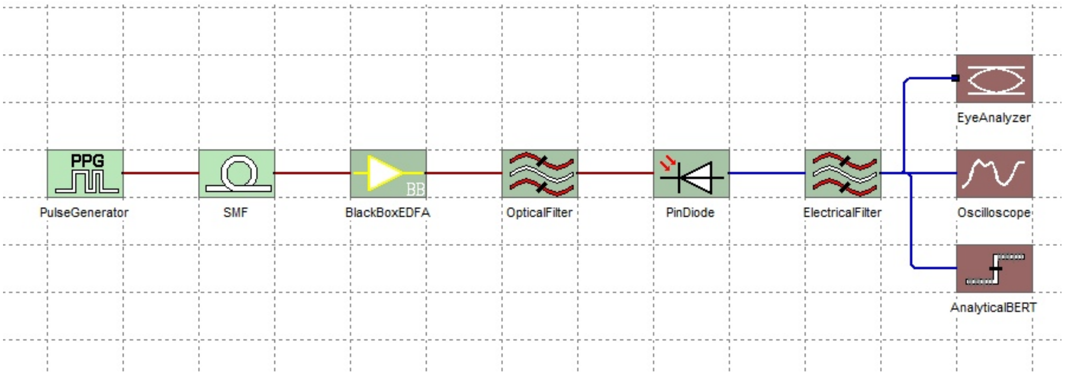


Figure 3.9.1. A typical setup to carry out BER and Q factor estimations for an NRZ-System (one span).

In the parameterized signal description, each WDM channel is characterized by its power (power of a binary mark and space), carrier frequency, bandwidth, and some information about the bit pulse geometry. Noise is propagated through the network using a coarsely sampled vector containing the broadband noise power spectral density (PSD). This approach enables a quick power budget analysis with an additional calculation e.g. of the signal to noise ratio or the extinction ratio.
Noise samples are distributed in the frequency interval *Noise bandwidth* with the spectral distance *Noise slice*. The noise vector contains n = *Noise samples* different noise samples. The default value for the *Noise bandwidth* is larger than the *Frequency range* of the sampled signal. Note, that the *Noise slice* and the *Frequency slice* are not necessarily the same in the analytical case. The analytical noise samples are stored in a different vector than the sampled signal vector.

The BER estimation at the receiver is then based on common assumptions such as *a Gaussian distribution of receiver noise*. Compared to numeric noise, fewer bits are required and a faster simulation is possible - normally, a total number of 512 to 4096 bits is usually enough to estimate BERs well below 10^3 as the analytical noise modeling approach does *not* require counting “real” bit errors.

The drawback of the analytical noise model is that the assumption of Gaussian distributed receiver noise does not hold when considering non-intensity modulated signals - e.g. for phase modulated formats such as DQPSK, DBPSK and 16-QAM. In these cases, analytical noise modeling should not be used and you should switch to numerical noise modeling.

⚠ **Analytical noise modeling can only be accurately used if intensity-modulated formats are considered. The AnalyticalBERT currently supports the modulation formats (N)RZ-OOK, Duobinary and CS-RZ. Other modulation formats have to be treated with numerical noise modeling.**

PHOTOSS offers the component “AnalyticalBERT” for BER and Q factor estimation of distorted, intensity-modulated signals. You can find out more about this component in section 7.8.1. Currently, the formats (N)RZ-On-Off-Keying, Duobinary and CS-RZ are supported. Aside from constraints on the modulation format, the analytical approach also makes the following assumptions:

- There should be at least one EDFA noise source in your system which actually adds noise to the signal and/or adapts the OSNR value of the signal.
- There should be at least one optical filter before the receiver.
- The optical signal is converted to the electrical domain by using a PIN diode component.
- There should be at least one electrical filter before the receiver.
- The noise components for both polarization axes are identical. However, this does not hold when PDL effects are considered (see page 52).

PHOTOSS internally uses the information about the EDFA component and filter structures in the simulation to derive the transmission function for the optical and electrical signal and then adjusts the separate sampled noise vector accordingly. If any of the above components is *not* included in your simulation, this may lead to inaccurate results at the receiver. The same applies when you want use custom MATLAB® components in PHOTOSS: You should not use your own signal sources, noise sources or receivers when selecting the analytical noise model as your components are most probably not designed to properly interact with the analytical model and process the necessary information. You can, however, always use your own MATLAB® components in the numerical noise mode. Components which do not influence the system noise should not pose a problem and can be used. A typical setup for BER and Q factor estimation in analytical noise mode is depicted in figure 3.9.1.

⚠ **The analytical noise model approach should not be used when you want to incorporate your own signal sources, noise sources or receivers (e.g. written in MATLAB®) in PHOTOSS.**

3.9.3 Numerical Noise Modeling

When considering numerical noise modeling, the numerical noise samples are automatically overlaid with the signal data in the sampled signal vector; there is no separation between the noise sample and signal sample vector. Thus, the *Frequency slice* and *Noise slice* as well as the *Frequency range* and *Noise Bandwidth* are identical.

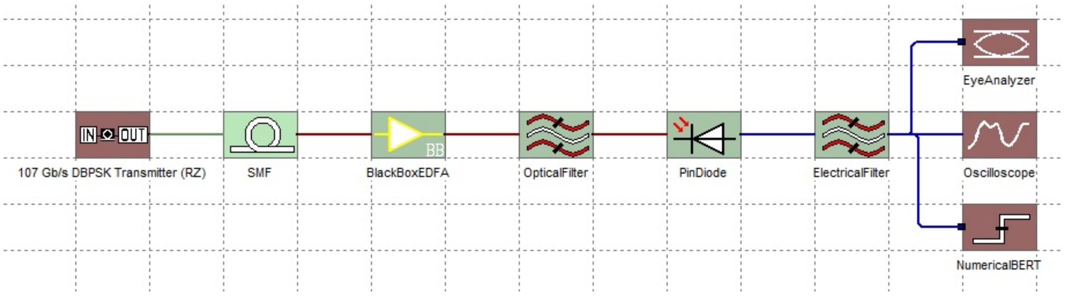


Figure 3.9.2. A typical setup to carry out BER estimations for a DBPSK system (one span).

In contrast to analytical noise modeling, the numerical noise model can be used with any modulation format and does not impose any limitation on the usage of optical or electrical filtering.

The drawback of the numerical noise model is that it usually requires more computational effort: Normally, one would be interested in adding linear and/or nonlinear distortion effects to the setup and in deriving observables like the bit error ratio (BER) in relation to the magnitude of the distortion effects. Such BER measurements can be carried out with the numerical noise model. However, it has to be made sure that the number of simulated bits is *sufficient* for deriving a reliable BER value with respect to the statistical nature of the noise components. Thus, the number of simulated bits must be large enough to predict the BER at a certain level. Consider this example:

- WDM setup with 10 channels (Total Field)
- each channel has 1024 bits per block
- each bit has 32 sampling points
- a single-block simulation is used

In order to make valid assumptions for BER values as low as 10^{-3}, statistically, at least one block with 1024 bits would have to be simulated to count one “real” bit error. One occurrence, however, hardly can be seen as a reliable representation of a statistical process. It would be much better to increase the number of bits per block or the number of blocks accordingly to simulate a total of at least 10,000 or even 100,000 bits to derive more reliable results for the BER value.

Currently, PHOTOSS offers the component “Numerical BERT” for the calculation of BER and Q factor values to estimate the signal quality. A typical BER measurement setup is depicted in figure 3.9.2. You can find out more about this component in section 7.8.4.

In both models, the *center frequency* of the sampled signal vector and the sampled noise vector are identical.

3.9.4 Summary

- PHOTOSS offers two noise models: The analytic and the numeric model
- The analytic noise model. . .
  - relies on an analytic approach that incorporates information of noise sources, optical and electrical filtering by regarding their (known) transfer functions
  - uses the analytical noise vector and “Analytical Performance Evaluation” structure in the signal for its noise calculations

  - allows for a fast calculation of the BER and Q factor by using the Analytical BERT and the "Analytical Performance Evaluation" structure
  - requires only a small number of simulated bits
  - works under certain assumptions that are only valid when considering intensity modulated signals (NRZ/RZ-OOK, Duobinary, CS-RZ)
  - does not work when phase modulated signals (e.g. DPSK, DQPSK, etc.) are considered
  - only works for a similar noise on *both* polarization axes. Modeling of polarization-dependent loss (PDL) effects is *not* possible

- The numerical noise model…
  - directly considers the noise components which are present in the sampled signal
  - allows for an accurate calculation of the BER and Q factor of the signal by counting "real" bit errors using the Numerical BERT
  - requires a large number of simulated bits (depending on the BER of interest)
  - works with intensity and phase modulated signals
  - works with signals with different noise levels on each polarization axis (e.g. with PDL effects)

3.10 Simulation Runs

3.10.1 Underlying Concept

When you want to execute a PHOTOSS simulation, we refer to this process as a “simulation run”. Normally, a simulation run consists of a fixed set of values for all component parameters and simulation parameters which can not be changed during the simulation run. Otherwise, the simulation is said to be a parameter variation and executing this simulation will result in the execution of multiple simulation runs (see below).

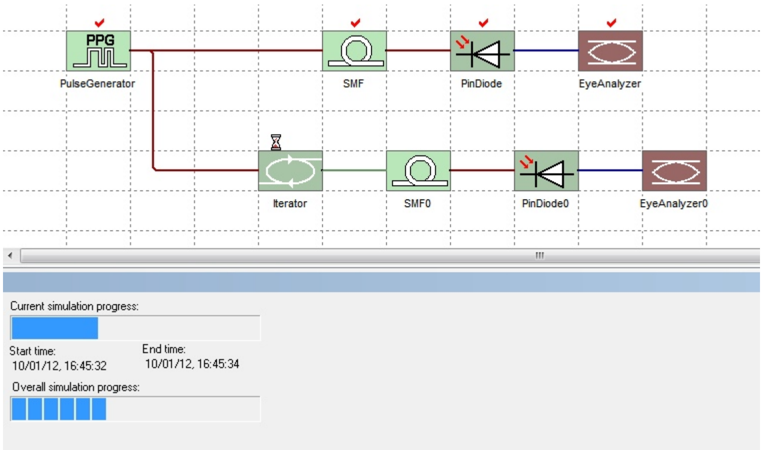
Each simulation run normally consists of two sub-runs which are described in more detail below:

1. the parameterized signal run and
2. the sampled signal run.

These two runs are part of the following steps which are executed each time a run is started:

1. All component parameters and simulation parameters for the current simulation run are checked. If their values are not valid, an error message (default) will be shown.
2. All component in-ports are checked for connection. If the in-port of a component is not properly connected, PHOTOSS will issue an error.
3. The PHOTOSS scheduler (see section 3.14) will determine the order in which the components will be processed.
4. The simulation state is changed from “idle” to a “processing” state. No changes can be made to parameters, component positions or connections on the grid.
5. The parameterized signal run will be initialized.
6. After the initialization the parameterized signal is propagated *once* through all components and networks.
7. Once the parameterized signal has been evaluated, the *sampled* signal run will be initiated.
8. After the initialization the sampled signal is propagated through all components and networks *once for each block*.
9. The progress bar will be updated to indicate how far the calculations have progressed.
10. Once the sampled signal has been evaluated, results (see section 3.16) are available and visualizer data (see section 3.17) can be displayed.
11. The simulation state is changed from “processing” to “finished”.
12. Results and visualizer data are available as long as the simulation remains in the “finished” state.

During a simulation run, the progress and the estimated time for the simulation will be displayed in the GUI (see figure 3.10.1). Some more complex simulations not only consist of one set of distinctive parameters for each component but they may contain multiple parameter sets. In these cases, a Parameter Variation (see section 3.13) can be executed. PHOTOSS will now automatically execute all simulation runs of the parameter variation after another. Steps 1 to 10 are repeated until all runs have been calculated and all results have been derived.



Current simulation progress:

Start time: 10/01/12, 16:45:32

End time: 10/01/12, 16:45:34

Overall simulation progress:

Figure 3.10.1. The progress of a running simulation is shown in the GUI. A red hook symbol is displayed above components which have already been calculated. A component which is currently being processed is designated by an hourglass symbol. Estimated end time and overall simulation progress are shown below the simulation grid.

3.10.2 Starting a Single Simulation Run

Before starting a simulation, make sure that

- the simulation parameters have been chosen,
- all network components have been placed on the grid and are interconnected accordingly,
- all component parameters have been set as required.
- the simulation has been saved. This can be done by clicking on the “File” and “Save as” menu entry in the GUI or pressing the hot key “CTRL + S”.

Start the simulation by pressing the ▷ toolbar button or by pressing “F5”. The Simulation toolbar provides a number of additional tools for monitoring and controlling the simulation (see section *Simulation Toolbar* on page 19). While the simulation is active, the current state is monitored in three windows named *output*, *warnings* and *errors*. They log various activities concerning the network components and the simulation control (also see section 3.23).

3.10.3 Starting a Parameter Variation

If a parameter variation is present in your simulation (this is the case when at least one simulation parameter has set its “Variation” flag to true), the PV toolbar button starts the parameter variation. The key combination “Ctrl + F5” also starts the parameter variation. Alternatively, you can open the simulation parameters overview dialog (figure 3.3.1 on page 27) and click on “Run Parameter Variation”.

3.10.4 Starting a Multiblock Simulation Run

Simulations may consist of multiple blocks (see section 3.11). All blocks will be calculated automatically for each simulation run. If the multi block simulation is also part of a parameter variation, all blocks will be processed once per variation run.

3.10.5 Starting Multiple Simulations

Sometimes you might want to not only run a single simulation (possibly also including a parameter variation) but to run a large number of simulations with different setups and/or parameter combinations. A few restrictions apply to this approach which should be kept in mind:

It is possible to open a multitude of simulation files at the same time. However, only one simulation can be executed in a single instance of PHOTOSS. Carrying out n simulations on the same computing machine requires n instances of PHOTOSS.

In order to automatically process large numbers of simulations consecutively in a single PHOTOSS instance, several tools may be very helpful: For example, you could opt to use PHOTOSS in the batch-processing mode (see section 4.3) and create a list of all the simulations that PHOTOSS should process. Alternatively you could also use PScript to create even more complex interactions between simulations - you could even use PScript to modify simulation setups depending on results obtained in earlier simulation runs, e.g. by adding or removing a system span, etc.

3.10.6 Clearing and Restarting a Simulation Run

If you want to start the simulation again and/or if you want to modify parameters and components after the simulation has reached the “finished” state, you have to “clear” the simulation by pressing the “F7” hot key or by clicking on the “Clear simulation” icon in the GUI. Once the simulation is “cleared”, the simulation state is reset to “idle” and all results and visualizer data will be deleted.

3.10.7 The Parameterized Signal Run

The parameterized signal run is carried out for every simulation run. During this run, the signal is comprised of the “channeldata” (see section 3.7) and will be propagated through the network and components. Before the signal is actually propagated through the network, the signal will be initialized in each component - some components use this phase of the simulation to carry out calculations which are done only once per simulation run, e.g. the calculation of PMD-effects for the PMD-path (see 3.22).

In general, the parameterized signal run is faster than the sampled signal run because no sampled signal representation is processed and the computational effort to calculate the “channeldata” is much lower. The difference in required simulation time for the parameterized and the sampled signal run may be several orders of magnitude - depending both on the number of simulated bits and the number of simulated blocks.

The main purpose of carrying out the parameterized simulation run is to get a quick estimate of the most relevant system parameters such as the signal average and maximum power levels or the accumulated dispersion. This run is not meant as a perfectly accurate simulation of the physical effects since it does not incorporate a sampled signal, but the information gained by carrying out the parameterized run can be very valuable nonetheless:

Consider a parameter variation of a very complex WDM transmission system with a lot of channels and twenty spans, 2^{18} bits shall be simulated in each of the 50 blocks. Nonlinear effects are included in the transmission fiber. Carrying out the parameterized run will probably only take a few seconds but processing the sampled run will be more likely to take several minutes or even hours. If - for any reason, e.g. a typing error - the parameters for the dispersion compensation of such a system would have been set in a non-intended manner, very much computational effort might be lost because the signal cannot be detected correctly at the receiver. In order to avoid such cases, it is possible to *only* carry out the parameterized signal run alone by deactivating the simulation parameter “Sampled” in the simulation parameters dialog (see section 3.3). It is very prudent to carry out a quick parameterized calculation of the signal to make sure that your signal power levels and dispersion parameters (etc.) are in order *before* beginning complex and time-consuming simulations of sampled signals.

After a parameterized simulation run has been carried out, all information of the parameterized signal (the “channeldata”) will be stored in each simulation link and can be accessed in the GUI by right-clicking on a link and selecting the “channeldata” option. This information will also be available after a sampled simulation run has been carried out for as long as the simulation has not been “cleared”.

3.10.8 The Sampled Signal Run

In this run, electrical and optical signals are discretized both in time and frequency. The sampled signal vectors are numerically propagated through the transmission system from their respective sources to the receiver components. This approach accounts for all relevant physical effects and time-transient phenomena.

As a default setting, a sampled signal run will be carried out in a simulation run as well. After the initialization, the *sampled signal* will be propagated through the network/components to calculate the physical effects which are specified by the component simulation models. A sampled signal run is carried out once for each block of the simulation, where the initialization of the signal is only carried out in the first block. You can find more information on the components of the sampled signal representation in section 3.7.

Simulation of a sampled signal is necessary to account for all physical effects which occur in an optical transmission system. The size of the sampled signal as well as the simulated polarization axes are solely determined by the settings of the simulation parameters which set the simulation band (section 3.3 and 3.8). A larger number of simulated bits or a larger number of sampling points results in a larger sampled signal and thus in more computational time being required for the sampled signal run.

In case of multiple signal channels in the separated channels simulation mode (see also 3.8), a sampled signal has to be computed for *each* individual channel. The larger the number of channels, the larger the computational effort to calculate all complex sampled signals.

3.10.9 Combined Simulations

“Combined Simulations” can be selected by activating the correspondent simulation parameter - they are available for single block simulations only. In contrast to the “normal” simulation mode, where the parameterized signal run is carried out for each component and afterwards the sampled signal run is carried out for each component, in this mode both signal runs are carried out directly after each other for each component of the network.

This setting can be used in combination with the “Clear Promptly” option (see section 3.24) in order to save memory for large single block simulations.

3.10.10 Options Available During a Simulation Run

Once a simulation run has entered its “processing” state, some restrictions apply to the options which are normally available to you. During a simulation run . . .

- . . . component and simulation parameters can not be changed
- . . . components can not be created, removed or linked

- . . . no changes can be made on the simulation grid
- . . . simulation files can not be modified
- . . . the random number generation can not be influenced
- . . . links and components which have already been processed can be accessed to view the "channel-data", visualizer information or simulation results. However, using this access during a simulation run is not recommended since it may crash the application and abort the simulation.

3.10.11 Summary

- Each simulation run is comprised of one specific set of Simulation and Component Parameters
- A Parameter Variation consists of several simulation runs with different sets of Simulation and/or Component Parameters
- During a simulation run, no changes of any kind can be made to simulation and component parameters or component positions/links
- During a simulation run, all components and links are calculated consecutively in a specific order determined by the PHOTOSS Scheduler
- After a simulation run, component results become available
- A simulation run may be comprised of either one or multiple blocks
- A simulation run is comprised of the following sub-stages:
  - Initialization of the parameterized signal run (once per simulation run)
  - Parameterized signal run (once per simulation run)
  - Initialization of the sampled signal run (once per simulation run)
  - Sampled signal run (*once per block!*)
- The information stored in components and the simulation links is available after a simulation run.
- This information can automatically be cleared by choosing the "clear promptly" option to save memory.

3.11 Multi Block Simulations

3.11.1 Underlying Concept

Normally, a simulation run (see 3.10) only incorporates the simulation of n bits per block and *one* signal block. Sometimes - especially when numerical noise modeling is used - the number of bits which have to be calculated in order to allow for measurements of very low bit error ratios tends to get very high. Thus, the number of bits n has to be increased and the sampled signal for each channel gets bigger. Depending on the RAM of your machine, a high number of bits per block may result in too high a demand for your system memory. This can be problematic when Fourier transformations have to be performed and large vectors / arrays have to be stored in memory.

In these cases, an alternative approach is to keep the number of bits per block low, but to use a *multiple* number of m blocks (the total number of simulated bits $n \cdot m$ will not change). Multiple block simulations require less memory than a single block simulation with the same number of bits. Each signal block will be processed consecutively after another. There is *no* interaction between consecutive blocks - inter-block-effects are *not* considered so make sure that the number of bits per block and the corresponding block duration are *long enough* for the representation of all relevant physical effects in your signal (e.g. the “memory length” when considering equalization, etc.).
Using the multi block approach is only necessary when using the numerical noise model. The analytical noise model does not require the computation of long bit sequences - even for low BER computations (see also 3.9).

Whenever you are using a multi block simulation, the following should be kept in mind:
Component and simulation parameters can not be changed between blocks in a multiple block simulation. However, some components act differently, depending on the way they are processed:
Some components which draw on the generation of random numbers to create numerical noise such as the EDFA or the PIN diode generate different random numbers (and thus: different noise) in each simulated block as a default. Normally, this is exactly what you would want when using numerical noise: Process a bit stream of more bits (in each block) while the noise in the system changes according to its statistical parameters from block to block. If - for any reasons - you want numerical noise effects to be constant throughout time you have to use single block simulations and a “deterministic” setting for the PHOTOSS random number generator (see 3.21).

Generally, there are two approaches for multi block simulations when considering the modulated signals:
The first approach is to simulate the *same* bit sequence in each block - each block will contain a different numerical noise representation but the bit sequence itself will be identical. This approach, however, can not be used when the length or your pseudo-random bit sequences (PRBS) needs to be longer than the number of bits per block.
The second approach is to create a PRBS which has the identical size as the number of bits (bits per block times number of blocks). It is also possible to create and use a longer PRBS but this may result in a warning when the sequence is processed. If you use PHOTOSS signal sources such as the Pulse Generator and you set up a PRBS which is shorter than the total simulated number of bits, the sequence will be automatically repeated in each block whenever necessary.

It should also be noted that PHOTOSS requires that the number of bits per block should be a power of two. This restraint is due to the fact that Fast Fourier Transforms (FFT) can be computed very much faster when the length of the sampled signal vector is a power of two. If - for any reasons - your customized PRBS is *not* a power of two you should use padding or zero bits to fill up your sequence.

3.11.2 Summary

- Multiple block simulations are useful when a large number of bits needs to be simulated and require less memory then a single block simulation with the same number of bits
- PRBS-Sequences (see 3.12) are always calculated for the whole number of bits (bits per block times number of blocks) per default to achieve the maximum sequence length - this may result in a repetition of certain parts of the PRBS if the PRBS length is smaller than the whole number of bits
- The following components *always change* during a multiple block simulation from block to block:
  - The numerical noise added to the signal (e.g. by the EDFA component)
  - Other components which draw on Random Number Generation (e.g. optimizers which are adapted in each block)
- The following components remain *constant* during a multiple block simulation from block to block:
  - The PMD-Path Jones Matrix; the same amount of PMD or PDL is added to the sampled signal in each block
  - All contents of the parameterized signal
  - All component parameters and simulation parameters

3.12 Modulation

3.12.1 Underlying Concept

One of the key objectives one strives to achieve in optical transmission systems is to transmit a modulated data stream in the optical domain at the transmitter site and to detect the signal at the receiver site - hopefully without a large number of misinterpreted data bits or so-called “bit errors”. In PHOTOSS, we group all considerations and ideas regarding the creation of a data stream in the “modulation” concept.

In general, the modulation concept incorporates various modulation formats to modulate binary data (zero and one bits) into a *symbol*. A symbol may hold the information of one or even multiple bits, depending on the type of modulation format. Data can be modulated in the electrical as well as in the optical domain. Once the binary source data has been modulated to the desired format, a complex optical (or electrical) signal containing the modulated symbols is obtained which can be transmitted through your simulation setup. Once the signal has been detected at a receiver component - e.g. the Analytical or Numerical Bit Error Rate Tester, it has to be *demodulated*, since these components process the *bit* pattern - *not* the symbol pattern.

Although modulation formats like NRZ-ASK (Non-Return to Zero Amplitude Shift-Keying) or DPSK (Differential Phase Shift-Keying) are very simple and have been implemented in many real optical transmission systems, recent trends indicate that higher order modulation formats which encode more than one bit into one complex symbol have attracted a lot of attention and are already implemented in current state-of-the-art systems. A few examples for higher modulation formats are Differential Quadrature Phase Shift-Keying (DQPSK) or Quadrature Amplitude Modulation (QAM) formats. Additionally, multiplexing techniques like Polarization-Multiplexed QPSK (Polmux-QPSK) are also receiving a lot of attention.

3.12.2 Restrictions to the Modulation Structure

Due to the new techniques and changing modulation formats listed above, the demands for a simulation tool for optical transmission systems change as well. As a result, the PHOTOSS modulation concept has undergone several improvements and changes. This will most likely hold true for upcoming PHOTOSS releases in the near future - so be sure to re-read this section of the manual to be up-to-date on which modulation formats are supported in PHOTOSS. The modulation structure of the parameterized signal (see also section 3.7) will most certainly be modified in future releases to ensure maximum usability even for complex higher order modulation formats.

In the current version of PHOTOSS, some restrictions apply to the parameterized signal and the modulation structure: The parameterized signal structure and especially the channelData structure were originally designed to incorporate intensity- or phase modulated symbol streams containing *one bit per symbol*. This is also the reason why the terms reference “symbol rate” / “bit rate” and “symbol pattern” / “bit pattern” are sometimes used interchangeably for each other which is *not* the case for higher modulation formats. Currently, the parameterized signal structure only can hold binary symbol information (e.g. a zero and a one *symbol*).

Still, higher order modulation formats like DQPSK *can* be implemented and used in PHOTOSS but the detection of these formats at the receiver must be achieved by spreading the detected *sampled* symbol stream with more than one bit per symbol into several symbol/bit streams with only one bit per symbol. Otherwise, the receiver components (see below) are not able to handle the input signal correctly. Additionally, the correct bit sequence for each symbol stream must be loaded into the parameterized signal before the receiver from a text file.

3.12.3 Implementing Setups with Higher Order Modulation Formats

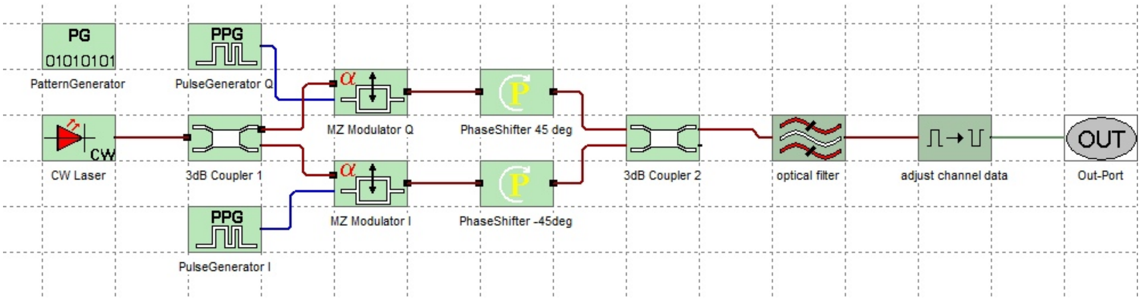
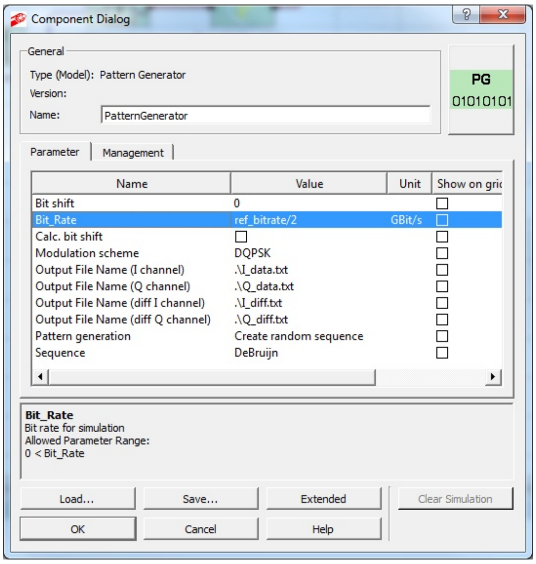


Figure 3.12.1. Simplified setup for a (D)QPSK transmission system.

How higher modulation formats can be successfully used in PHOTOSS setups can be understood best from looking at a simplified (D)QPSK transmission system as depicted in figure 3.12.1. Here, a Pattern Generator (see also section 7.11.5) is used to generate a QPSK and DQPSK symbol stream. Its parameters are depicted in figure 3.12.2. Note that the parameter “Bit_Rate” of the pattern generator actually corresponds to a *baud rate* when considering higher order modulation formats! If - for example - a reference bit rate of 40 Gbit/s has been set in the simulation parameters, the bit rate of the pattern generator (aka the baud rate) should be set to *half* that value since two bits are transmitted per QPSK symbol. You will also notice that the pattern generator does not have any output ports. This is due to the fact that the parameterized signal can not contain patterns with more than one bit per symbol. Thus, the I and Q channels have to be saved to separate files which can be loaded at the receiver to specify the corresponding branch.



General
Type (Model): Pattern Generator
Version:
Name: PatternGenerator

Name	Value	Unit	Show on gric
Bit shift	0		□
Bit_Rate	ref_bitrate/2	GBit/s	□
Calc. bit shift	□		□
Modulation scheme	DQPSK		□
Output File Name (I channel)	.\I_data.txt		□
Output File Name (Q channel)	.\Q_data.txt		□
Output File Name (diff I channel)	.\I_diff.txt		□
Output File Name (diff Q channel)	.\Q_diff.txt		□
Pattern generation	Create random sequence		□
Sequence	DeBruijn		□

Bit_Rate
Bit rate for simulation
Allowed Parameter Range:
0 < Bit_Rate

Figure 3.12.2. Parameters for the Pattern Generation.

At the receiver (figure 3.12.3), the complex incoming symbol stream with two bits per symbol is divided into two symbol streams with one bit per symbol (the I and Q channel). This is accomplished by using a differential receiver structure. The most important part is that the channel data must be adjusted by using the pattern modifier components: The upper branch of the receiver detects the Q and the lower branch detects the I channel (this is indicated by the settings of the calculator components). Thus, the pattern modifiers use the settings depicted in figure 3.12.4 to write the correct bit stream to each channel.

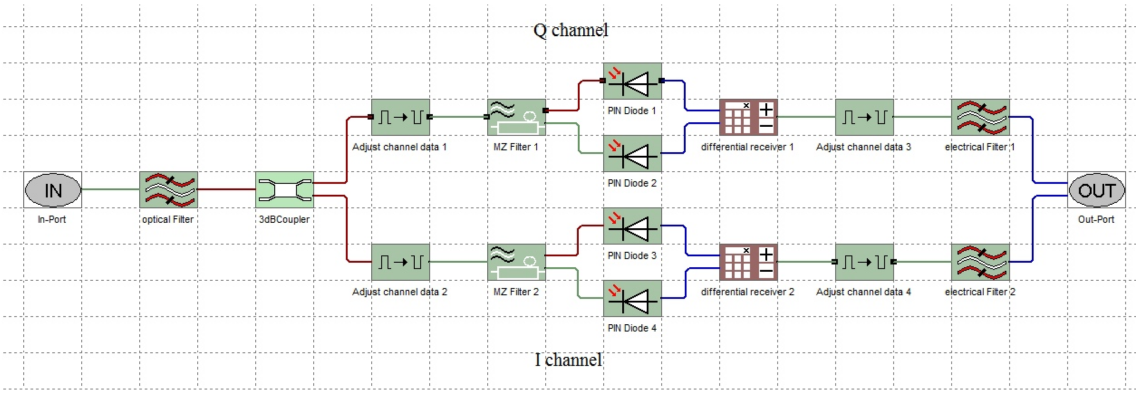
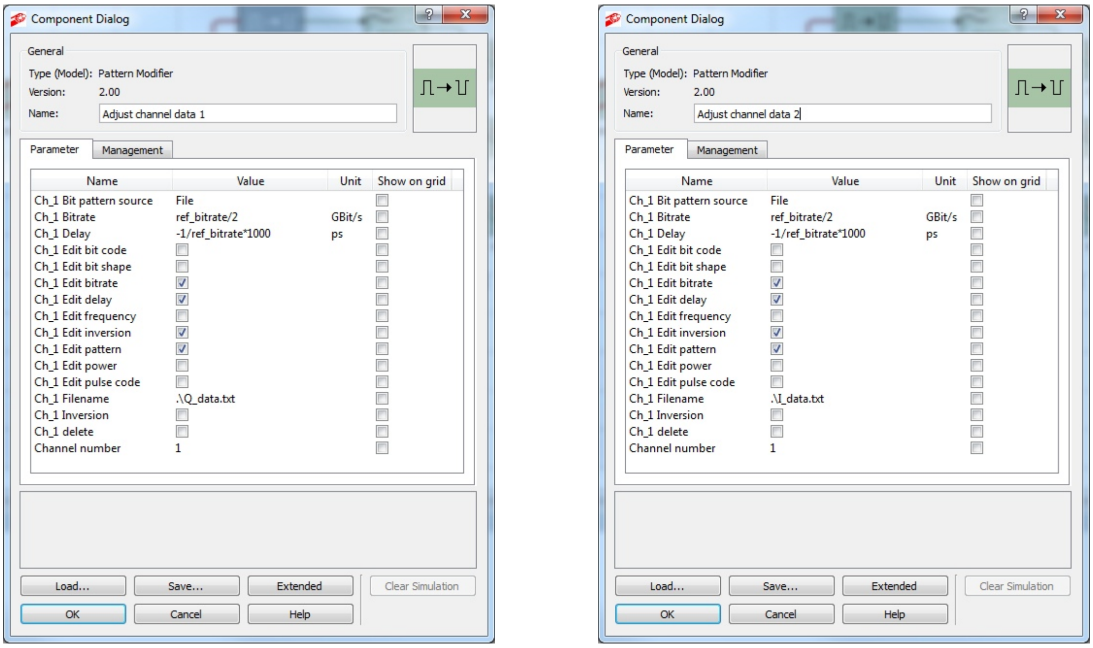


Figure 3.12.3. Simplified receiver for a (D)QPSK transmission system.



Component Dialog

General

Type (Model): Pattern Modifier
Version: 2.00
Name: Adjust channel data 1

Parameter | Management

Name	Value	Unit	Show on grid
Ch_1 Bit pattern source	File		□
Ch_1 Bitrate	ref_bitrate/2	GBit/s	□
Ch_1 Delay	-1/ref_bitrate*1000	ps	□
Ch_1 Edit bit code	□		□
Ch_1 Edit bit shape	□		□
Ch_1 Edit bitrate	☑		□
Ch_1 Edit delay	☑		□
Ch_1 Edit frequency	□		□
Ch_1 Edit inversion	☑		□
Ch_1 Edit pattern	☑		□
Ch_1 Edit power	□		□
Ch_1 Edit pulse code	□		□
Ch_1 Filename	.\Q_data.txt		□
Ch_1 Inversion	□		□
Ch_1 delete	□		□
Channel number	1		□

Component Dialog

General

Type (Model): Pattern Modifier
Version: 2.00
Name: Adjust channel data 2

Parameter | Management

Name	Value	Unit	Show on grid
Ch_1 Bit pattern source	File		□
Ch_1 Bitrate	ref_bitrate/2	GBit/s	□
Ch_1 Delay	-1/ref_bitrate*1000	ps	□
Ch_1 Edit bit code	□		□
Ch_1 Edit bit shape	□		□
Ch_1 Edit bitrate	☑		□
Ch_1 Edit delay	☑		□
Ch_1 Edit frequency	□		□
Ch_1 Edit inversion	☑		□
Ch_1 Edit pattern	☑		□
Ch_1 Edit power	□		□
Ch_1 Edit pulse code	□		□
Ch_1 Filename	.\I_data.txt		□
Ch_1 Inversion	□		□
Ch_1 delete	□		□
Channel number	1		□

Figure 3.12.4. Left: Settings for the pattern modifier for the Q branch. Right: Settings for the I branch.

3.12.4 Components Depending on Input-Modulation

Many of the PHOTOSS components work on the complex electrical or optical signal itself and they do not require any additional information regarding its modulation format. Some components, however, *are* dependent on the input modulation format. This means that they need additional information which is given in the modulation structure of the parameterized part of the MultiSignal (see below) to decode the incoming symbol stream in order to process the incoming complex signals correctly. Additionally, not every component which depends on the input modulation format is able to process *all* incoming formats. If a signal with a non-supported modulation format is input into a component, an error will be thrown to indicate this. The following components of PHOTOSS are dependent on the modulation format (given in brackets):

- Analytical BERT (ASK)
- Bitpolarimeter (ASK and DPSK)
- Constant-Modulus Equalizer (DPSK)

- Decision-Directed Equalizer (DPSK)
- DPSK Decoder (DPSK)
- Eye Analyzer (ASK and DPSK)
- Numerical BERT (ASK and DPSK)
- PMD Compensator (ASK)
- Single Pulse Analyzer (ASK)
- TDM-Mux (ASK and DPSK)

All other PHOTOSS intermediate or receiver components should be compatible to any given modulation format. You should also note that you can - of course - define complex signals containing your own modulation formats by writing your own transmitter and receiver components. This feature is explained in much more detail in section 4.7 on page 120.

3.12.5 The Modulation Structure

The parameterized part of the PHOTOSS MultiSignal (see section 3.7) contains the so-called “modulation” structure for each individual signal channel. The following information is included in the structure:

- Channel: Channel for which the modulation is specified.
- AmplitudeMode: Characterizes the pulse form. Can be *SQRT_AMPL*, *AMPL* and *PWR*.
- Shape: Characterizes the pulse shape. Can be *REC*, *COS2*, *SIN*, *SOLITON*, *GAUSSIAN*, *TRIANGLE*, *SAW_UP* or *SAW_DOWN*
- PulsCode: Characterizes the pulse code. Can be *RZ*, *NRZ*, *ORZ*
- duty_cycle: Specifies the duty cycle of the modulation format.
- Method: Specifies the pulse generation mode. Can be *DIRECT*, *FILTER*.
- Bitcode: Specifies the bit coding. Can be *NONE*, *DUOBINARY*, *AMI*, *PHASESWITCH*, *AMPLITUDESWITCH*
- Bitrate: Specifies the channel bit rate.
- Bit pattern: Holds the (binary) bit pattern for each channel. Complex symbols patterns are not supported in the parameterized signal, yet.

3.12.6 Summary

- Modulation formats can be characterized in PHOTOSS by the modulation structure in the parameterized signal
- This structure was designed for symbol patterns with one bit per symbol
- More than one bit per symbol can currently not be displayed in the parameterized signal
- Sampled signals with more than one bit per symbol are allowed

3.13 Parameter Variation

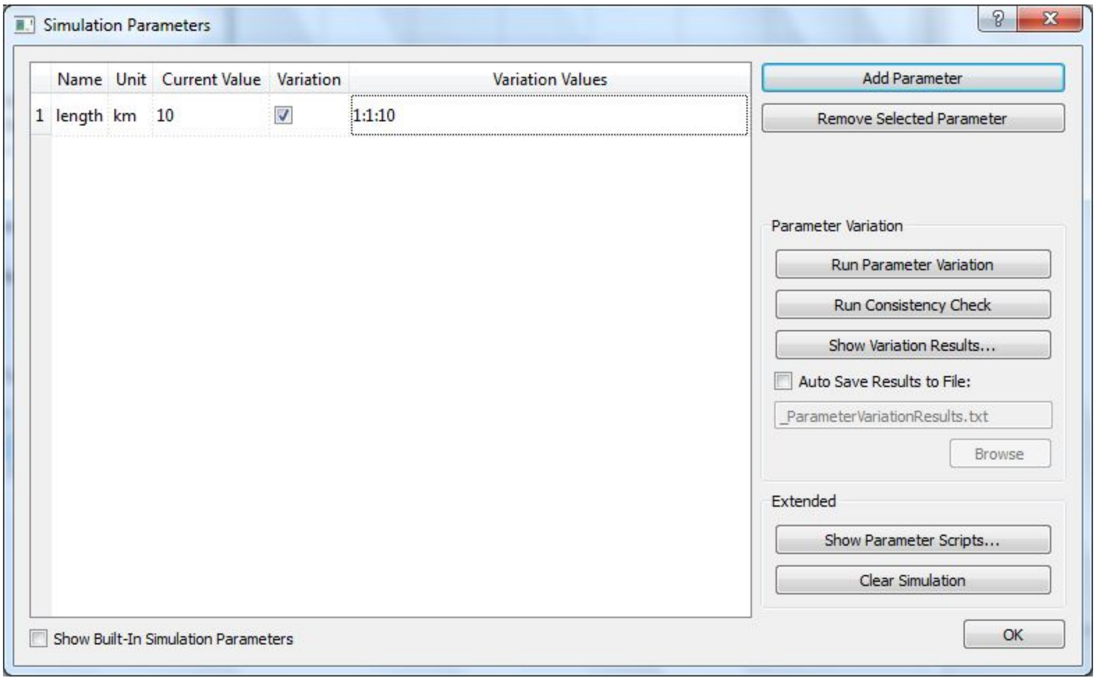
3.13.1 Underlying Concept

The parameter variation functionality provides a convenient way to perform multiple simulation runs with varying parameter settings for the same network. A typical application is e.g. the optimization of specific parameters and the analysis of their impact on the transmission performance. As PHOTOSS offers multi-dimensional variation, multiple parameters may be varied within a single variation process in order to investigate their mutual dependencies and find globally optimized parameter sets.

The basic steps of a parameter variation procedure are:

1. Select all **simulation parameters** to be varied and specify their respective **values**.
2. Select one or more simulation **results** of interest.
3. **Start** a parameter variation. A separate simulation run will be carried out for all combinations of the specified parameter values. The selected results will be collected and stored for each simulation.
4. **Analyze** the collected results and determine the optimal settings.

Depending on the number of parameters to be varied and the number of different values chosen for each parameter, a large number of simulation runs may be carried out in step 3. As the simulation runs are automatically carried out one after another with no need for manual user actions, time-consuming optimization runs may easily be carried out overnight or during weekends.



	Name	Unit	Current Value	Variation	Variation Values
1	length	km	10	☑	1:1:10

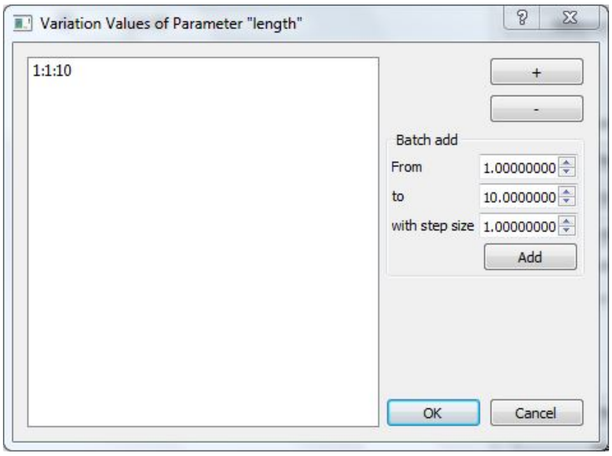
□ Auto Save Results to File: _ParameterVariationResults.txt

□ Show Built-In Simulation Parameters

Figure 3.13.1. Simulation Parameters Dialog: Add and modify the parameters you want to use for your variation.

3.13.2 Setting up and starting a parameter variation

To set up a parameter variation, first open the “Simulation Parameters” dialog in the GUI (see figure 3.13.1). Now, you can either use the built-in simulation parameters for your variation or you may create your own parameters by clicking “Add Parameter”. You can now assign a custom name, unit and current value to your parameter. The name must not contain any of +/-, ∗, \, /, ˆ, ( ), [ ] { } and must not start with a number. If you want to also use this parameter for a parameter variation, simply check the “variation” checkbox and double click on the “Variation Values” column to set up your variation values (see fig. 3.13.2).



Variation Values of Parameter "length"

1:1:10

Batch add

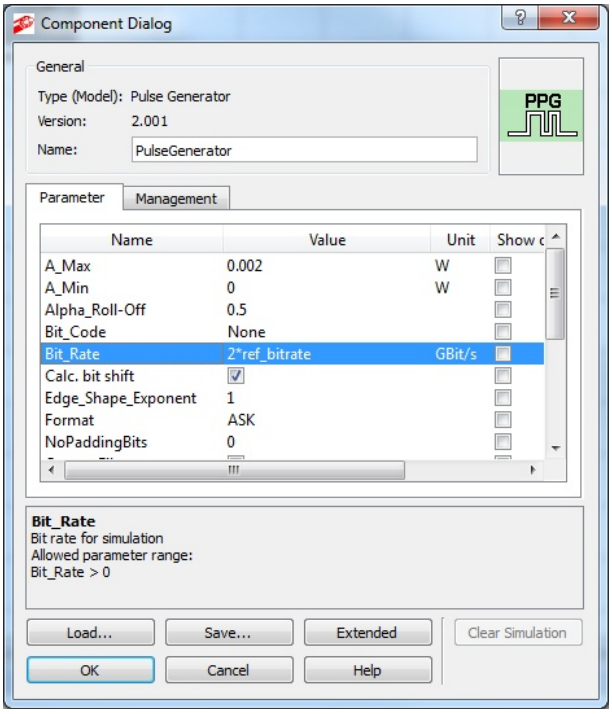
From 1.00000000

to 10.0000000

with step size 1.00000000

Figure 3.13.2. Setting up variation values for a parameter variation.

You can either define each value for the variation manually by pressing the “+” button and editing each value or generically by using the “Batch add” functionality on the right by supplying a minimum value, a step size and a maximum value. You can also enter ranges by double-clicking the variation value in the simulation parameters dialog and typing them in the format *minimum value*:*step size*:*maximum value*.



Component Dialog

General

Type (Model): Pulse Generator

Version: 2.001

Name: PulseGenerator

Name	Value	Unit	Show c
A_Max	0.002	W	□
A_Min	0	W	□
Alpha_Roll-Off	0.5		□
Bit_Code	None		□
Bit_Rate	2*ref_bitrate	GBit/s	□
Calc. bit shift	☑		□
Edge_Shape_Exponent	1		□
Format	ASK		□
NoPaddingBits	0		□

Bit_Rate
Bit rate for simulation
Allowed parameter range:
Bit_Rate > 0

Figure 3.13.3. The simulation parameter “ref_bitrate” is used as placeholder in a formula for the parameter “bitrate” in the Pulse Generator.

Once you have set up all parameters and their ranges for your variation, you also might want to use your custom and/or built-in simulation parameters as placeholders in various components of your simulation. To do so, simply open the component you are interested in and insert the name of the simulation parameter as a placeholder for your chosen component parameter or insert the name in a formula for a component parameter. During the variation runs, the component parameter will now be automatically set to the current value of your simulation parameter (see fig. 3.13.3).

After setting up all parameters for the variation, you may also want to execute a consistency check in advance before starting the variation. Simply click on “Run Consistency Check” (see fig. 3.13.1) - any invalid set of parameters will be shown and has to be removed from the variation set.

Now you may want to choose your component result(s). Open all components you are interested in and activate their results that you want PHOTOSS to compute. Your parameter variation is now ready for execution - click on “Run Parameter Variation” to start it. Once the parameter variation has finished and all simulation runs are completed, you may also take a look at all results by clicking on “Show Variation Results”.

3.13.3 Creating a Monte Carlo Parameter Variation

Some physical phenomena in optical transmission systems (e.g. Polarization Mode Dispersion) are of a statistical nature. In a real system they would change constantly due to certain factors of the environment - for example temperature, strain on the fiber, etc. In order to predict statistic effects, it may be necessary to run the very same simulation with identical parameter sets multiple times.

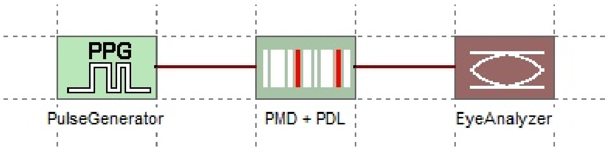
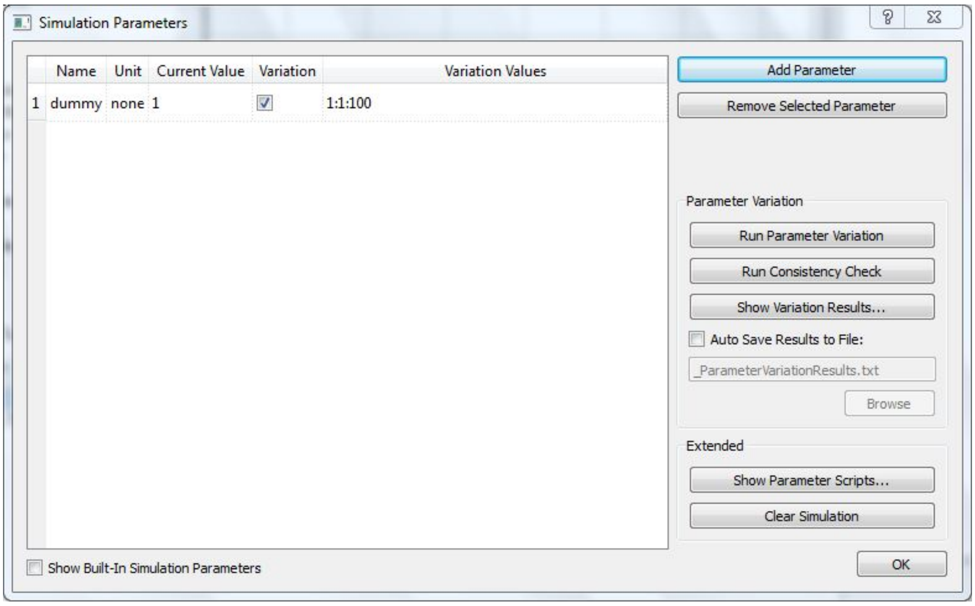


Figure 3.13.4. A simple setup for a Monte Carlo parameter variation

For example let us consider a simple simulation which is comprised only of a Pulse Generator, a PMD and PDL Emulator and an Eye Analyzer using only default values (see fig. 3.13.4). The random generator in the simulation parameters has been set to “Random” mode. Each time the simulation is executed, the PMD and PDL Emulator will come up with a new set of Birefringent and PDL elements and thus, will add a different kind of distortion to the signal. As a result, the eye analyzer will calculate a different value for the optical eye opening after each simulation run.



	Name	Unit	Current Value	Variation	Variation Values
1	dummy	none	1	☑	1:1:100

□ Auto Save Results to File: _ParameterVariationResults.txt

□ Show Built-In Simulation Parameters

Figure 3.13.5. A dummy parameter for Monte Carlo parameter variations

In order to assess the statistical quality of the eye opening result, the simulation shown above needs to be carried out many times - the ideal solution is to create a parameter variation to carry out this task. The only difference to a "normal" parameter variation described above is that we obviously do not need any placeholder for the components of the simulation. You only need to create a "dummy" parameter and assign generic values to it. If - for example - you want the simulation to be carried out 100 times, simply add a dummy parameter and assign the generic variation values $min = 1$, $step = 1$, $max = 100$ (see fig. 3.13.5).

Just keep in mind to activate the "Variation" check box and to set the random generator setting to "Random" or "Once Deterministic" for your simulations and you are ready to start your variation. You should also keep in mind that the numerical noise (etc.) will *also* change for each variation run and each block in each variation run.

3.13.4 Viewing Variation Results

There are several ways of assessing the results of your parameter variations. Please note that PHOTOSS will save those only for values which were marked for evaluation before the simulation run was started. You can do this by activating the corresponding "Value" checkbox in the "Results" tab in the Component Dialog. For every selected value a file will be created. During the parameter variation, each simulation run's results are appended to their corresponding files. Please note that clearing the simulation will not delete these files, so you can start separate variations and eventually evaluate the gathered results as a whole without having to recover any files in between. Using this method of data presentation however you have to take care of associating the results to their corresponding varied values yourself.

PHOTOSS can take care of this for you. With the "Auto Save Results to File" checkbox activated, a file containing all results marked for monitoring during the parameter variation will be created. This file only contains the results of the last variation that has been executed.

A more focussed way of analyzing and evaluating your data can be achieved in the "Parameter Variation Results" dialog which can be opened via the "Simulation Parameters" window. After selecting all the varied parameters and component results by clicking on them while holding down the *Control (Ctrl) key* you can either export them to a text file in order to analyze them with another program of your choice by clicking the “to ASCII File” button or directly plot them in MATLAB® using the “to MATLAB®” button.

3.13.5 Summary

- A parameter variation is specified in the simulation parameters dialog by setting the “Variation Values” for the designated parameter
- A parameter variation may apply to both custom and built-in simulation parameters
- Components may use simulation parameters as placeholders for their component parameters
- A parameter variation may consist of n varied parameters and may thus be n-dimensional
- A parameter variation (usually) consists of multiple simulation runs
- Each set of parameters comprises *one* simulation run of the parameter variation
- More complex parameter variations may be created using PScript (see below)
- Component results have to be marked for evalution before starting a parameter variation

3.14 Scheduling

3.14.1 Underlying Concept

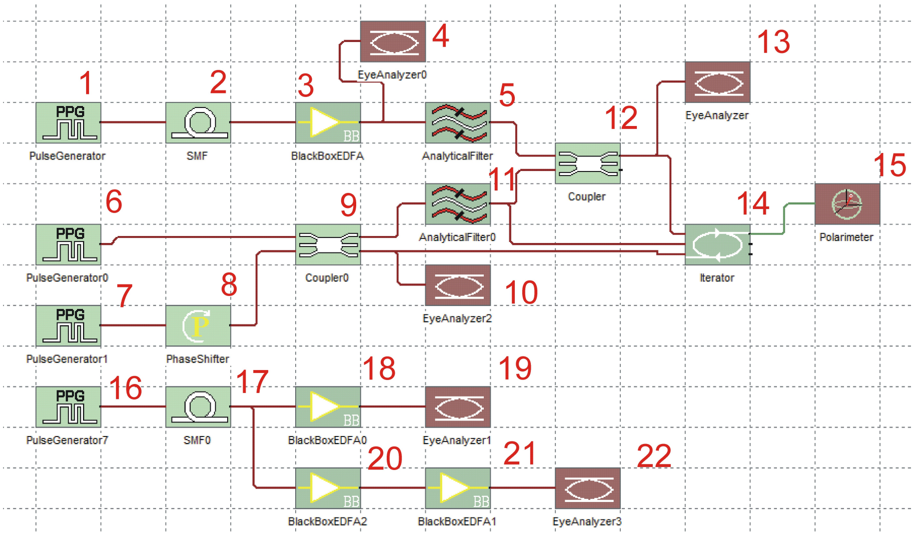


Figure 3.14.1. A simulation can consist of several “branches” that may or may not be connected with each other. Red numbers indicate the order in which components are sequenced.

As PHOTOSS components are connected together on the PHOTOSS grid, information is exchanged between them through “links”. Since not all components need necessarily be connected together, it is also possible to have several “branches” in a single simulation file such as depicted in figure 3.14.1. The PHOTOSS Scheduler will determine the order in which components of a simulation have to be processed. Its methods of determining the order are too complex to be described here in full detail but generally, the order is derived deterministically, hence the same simulation will always be scheduled identically if no components are added, removed or re-linked. Several parameters influence the scheduling algorithm and a few of them may be controlled by the user. Normally, it is best to let PHOTOSS worry about sequencing the components but if you specifically need to influence the sequence, you should first ask yourself if you could find a less complex workaround solution like splitting your simulation by using “File Loaders” and “File Savers”.

The most important feature of the scheduler is that signals can only be propagated from the left to the right side of the simulation, meaning that a signal always originates from a source, is propagated through numerous components and finally reaches a receiver component. Signal “back flow” or “back propagation” of signals outside the scope of a single component is not supported. The only exception from this rule are the iterator components which allow for a limited “loop mode” trough their components.

The following rules influence the scheduling:

- Components, which have no in- or out-ports (such as the “Pattern Generator”) are always processed first.
- Source components which have no in- but at least one out-port are processed next.
- Source components are scheduled in the same order they have been set up with.
- Multiplexer components (e.g. the “Coupler”) can only be processed once all components on the left side have been processed.
- If possible, shorter paths are processed before longer paths.

If these rules are observed, the components of simulation in figure 3.14.1 are sequenced in the order shown in the picture.

3.14.2 Summary

- PHOTOSS carries out the sequencing of its components and simulation links automatically
- The sequence is determined once per simulation run and can not be set up manually
- The sequence is identical for all blocks of a simulation run
- Transmitter components are always sequenced before receiver components
- Signal propagation is carried out consecutively from left to right
- Signals usually propagate through a component only once per block with the exception of an iterator where the signal passes the component once for each iteration
- Signal loops or signal “back flow”/back propagation from the right to the left side is not supported

3.15 Convolutions

3.15.1 Underlying Concept

Various components like FIR or IIR filters require a convolution of the optical or electrical signal in the time domain. In principle, there are two ways to convolute the signal: The first way is a *linear convolution* which simply extends a sampled signal whenever the memory length for the convolution exceeds the sampled signal length. Linear convolutions are not supported from PHOTOSS version 5.0 or newer. The second way is to use a *cyclic convolution*:

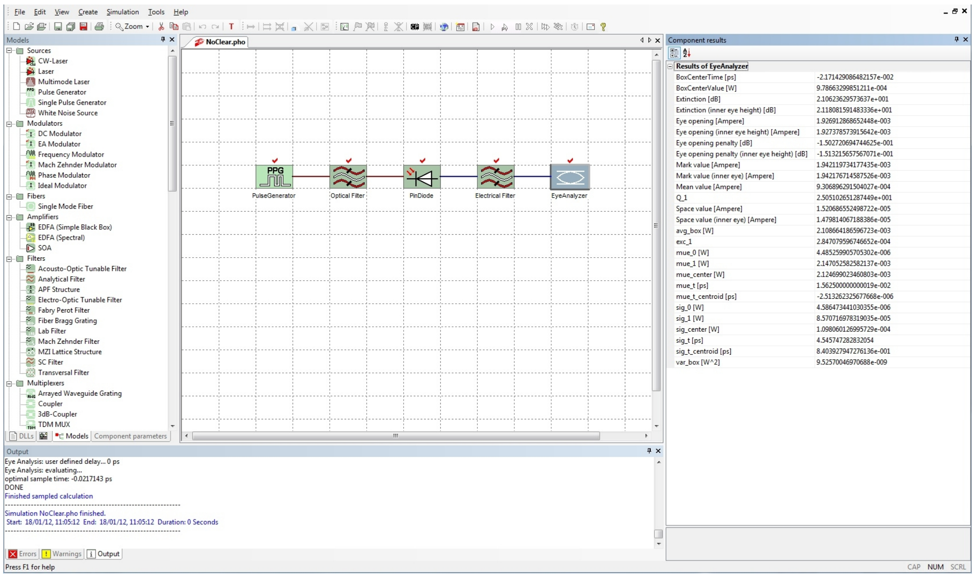
Cyclic convolutions assume all signal blocks to be periodically continued in both directions. Each block is calculated separately without any mutual dependencies. At the receiver, the results are accumulated and averaged over all received signal blocks where required. Cyclic convolutions are fast and memory-efficient, but may yield spurious results due to phase discontinuities and artificial wrap-around effects at the block boundaries. Numerical noise may further exhibit non-physical auto-correlation as it is also assumed periodic over the block duration. PHOTOSS implements advanced cyclic convolution algorithms in order to reduce the above effects where possible, but keep in mind that the origin of the spurious results is the method itself and in general cannot be fully suppressed.

3.15.2 Summary

- PHOTOSS uses a cyclic convolution mode to convolute optical or electrical signals in the time domain.
- Cyclic convolutions may cause artificial wrap-around effects for large convolution memory lengths. Increasing the number of samples / bits per block may help to remedy this problem.

3.16 Results

3.16.1 Underlying Concept



Results of EyeAnalyzer	
BoxCenterTime [ps]	-2.171429086482157e-002
BoxCenterValue [W]	9.786632998512 11e-004
Extinction [dB]	2.10623629573637e+001
Extinction (inner eye height) [dB]	2.118081591483336e+001
Eye opening [Ampere]	1.926912868652448e-003
Eye opening (inner eye height) [Ampere]	1.927378573915642e-003
Eye opening penalty [dB]	-1.502720694744625e-001
Eye opening penalty (inner eye height) [dB]	-1.513215657567071e-001
Mark value [Ampere]	1.942119734177435e-003
Mark value (inner eye) [Ampere]	1.942176714587526e-003
Mean value [Ampere]	9.306896291504027e-004
Q_1	2.505102651287449e+001
Space value [Ampere]	1.520686552498722e-005
Space value (inner eye) [Ampere]	1.479814067188386e-005
avg_box [W]	2.108664186596723e-003
exc_1	2.847079596746652e-004
mue_0 [W]	4.485259905705302e-006
mue_1 [W]	2.147052582582137e-003
mue_center [W]	2.124699023460803e-003
mue_t [ps]	1.562500000000019e-002
mue_t_centroid [ps]	-2.513262325677668e-006
sig_0 [W]	4.586473441030355e-006
sig_1 [W]	8.570716978319035e-005
sig_center [W]	1.098060126995729e-004
sig_t [ps]	4.545747282832054
sig_t_centroid [ps]	8.403927947276136e-001
var_box [W^2]	9.52570046970688e-009

```
Eye Analysis: user defined delay... 0 ps
Eye Analysis: evaluating...
optimal sample time: -0.0217143 ps
DONE
Finished sampled calculation
------------------------------------------------------------
Simulation NoClear.pho finished.
 Start:  18/01/12, 11:05:12  End:  18/01/12, 11:05:12  Duration: 0 Seconds
------------------------------------------------------------
```

Figure 3.16.1. The results of a component are available after the simulation and are shown on the component results page.

When simulating optical transmission systems, one is usually interested in certain observables - e.g. the bit error ratio (BER) of the system, the accumulated dispersion or the required optical signal-to-noise-ratio (OSNR) for a given BER. In the PHOTOSS application, we denote observables as "results". Results are defined by many of the PHOTOSS components and become available once a simulation run is finished and the component(s) have been calculated.

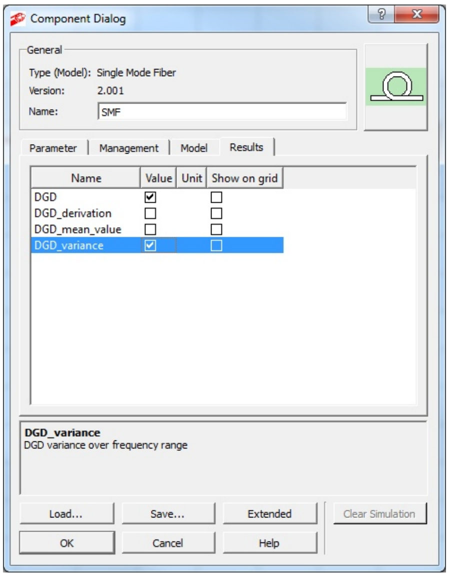
Most of the component results can be shown in the GUI and are displayed on the component results page (see figure 3.16.1). The result page contains the name, unit and computed value for each result of the selected component.

You will notice that the results which are shown on the component results page are usually of the data types integer or double. In some cases, components also offer the calculation of output results which are comprised of a vector or matrix of these data types. These results are not displayed in the GUI but they can be written to so-called "result file" text files. One example would be using the option "Auto Save" of the component "Oscilloscope", etc.

Sometimes, the result value is actually not an integer or double data type, but one of the expressions "1.#QNAN", "1.#INF" or "-1.#IND". This may have several reasons:

- The simulation has not yet finished or has been cleared. The result has not been computed, yet, or has been erased.
- "Clear Promptly" has been activated for the selected component (see 3.24) and all results have been erased.
- The component could not compute a valid integer or double value for the desired result. This can be the case if, for example, signal distortions become so large that an eye opening at the receiver could not be calculated or a numerical optimizer could not converge to a stable solution, etc.

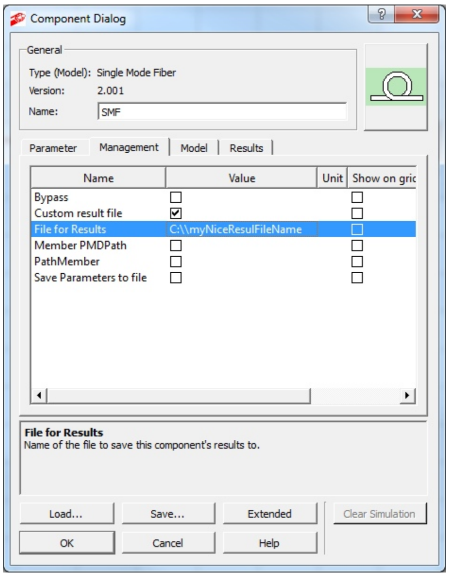
3.16.2 Selecting Results and Result Files



Type (Model): Single Mode Fiber
Version: 2.001
Name: SMF

Name	Value	Unit	Show on grid
DGD	☑		□
DGD_derivation	□		□
DGD_mean_value	□		□
DGD_variance	☑		□

DGD_variance
DGD variance over frequency range



Type (Model): Single Mode Fiber
Version: 2.001
Name: SMF

Name	Value	Unit	Show on gric
Bypass	□		□
Custom result file	☑		□
File for Results	C:\\myNiceResulFileName		□
Member PMDPath	□		□
PathMember	□		□
Save Parameters to file	□		□

File for Results
Name of the file to save this component's results to.

Figure 3.16.2. Left: Results can be saved to individual result files be selecting them in the component properties dialog. The results “DGD” and “DGD_variance” will be saved to a file. Unselected results will still be calculated and are available in the GUI. Right: A custom result file prefix and path can be specified.

Usually, results are not automatically written to result files. However, you can decide to create a (separate) result file for each component result by opening the component parameters dialog, switching to the “Results” tab and selecting the result which you want to save to a file (see left side of figure 3.16.2). Unselected results will still be calculated and are shown in the GUI.

As a default, result files will be written to an automatically generated result file. Its file name will be composed by using the prefix of the component (custom) name, the underscore “_” and the suffix of the result name. Thus, the result “DGD” of the component “Single Mode Fiber” with the name ”mySMF” would produce the result file “mySMF_DGD”. The file will be saved in the same directory where the simulation file resides.

It is also possible to exchange the automatically generated prefix of the component name and to save the result file in any other directory by activating the parameter “Custom result file” on the “Management” page in the component dialog and entering a Custom file for results (see right side of figure 3.16.2). Now, the result file will be saved under the given path (and with the given prefix) and retain its normal suffixes of the underscore and the result name.

If the result file for a component result already exists, PHOTOSS will automatically add the newly computed result to the next line of the result file - result files will *not* be overwritten. Thus, running the same simulation n times would result in n result entries in the result file(s).

3.16.3 Graphical Visualization of Results

Some components as well as the simulation links offer the ability to view graphical representations of vector- or matrix-style results like e.g. an oscilloscope view of the sampled signal in the time domain or a spectrum analyzer. These components use Visualizers for this functionality - please see section 3.17 for further details on this topic.

3.16.4 Summary

- Results are defined by PHOTOSS components
- Results are available after a simulation run is finished
- Clearing a simulation also resets the calculated result values
- Results are usually scalar values of the type `double` or `integer`
- Some components allow for the calculation of vector type or matrix type results. These results can only be saved to text format files and will not be shown in the GUI.
- All component results can be written to a text format file automatically after the simulation has finished
- Each component may specify its individual result text file
- Parameter Variations can save the results for each individual simulation run separately to individual files or a global result file

3.17 Visualizers

3.17.1 Underlying Concept

Visualization instruments the graphical display (or “plotting”) of signals, simulation results, or specific properties of network components. Most visualizers are included in PHOTOSS components. After a simulation run has finished and results become available, visualizer data is also available and can be inspected. Clearing a simulation also resets the visualizer data.

PHOTOSS comes with the following built-in visualization components (please refer to the Component Manual for details on the components and their respective parameters):

- **Oscilloscope:** Displays time-transient signals.
- **Spectrum Analyzer:** Displays signal spectra.
- **Eye Analyzer:** Displays eye pattern diagrams and performs eye analysis calculations.
- **Numerical BERT:** Performs an estimation of the bit error rate (BER) based on numerical algorithms.
- **Analytical BERT:** Performs an estimation of the bit error rate (BER) using analytical and semi-analytical approximations.
- **Single Pulse Analyzer:** Delivers various information from the analysis of a single bit pulse.
- **Parametric signal Analyzer:** Provides information on the parametric signal analysis.
- **Simulation Links:** Provides the same information as an oscilloscope or spectrum analyzer.

In addition to the above visualization components, any network component may provide one or more graphs showing certain component-specific properties or calculation results. These graphs are displayed via the same visualization interface as used for the above visualization components.

3.17.2 Accessing Visualization Graphs

Graphs are accessed by right-clicking a network component or right-clicking a connection link between two components. In either case, a pop-up menu appears that lists all available graphs (possibly along with some non-graphical results). Just select a menu entry and a corresponding display window will open.

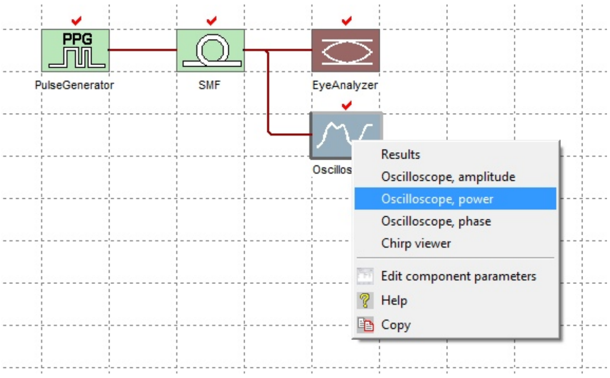
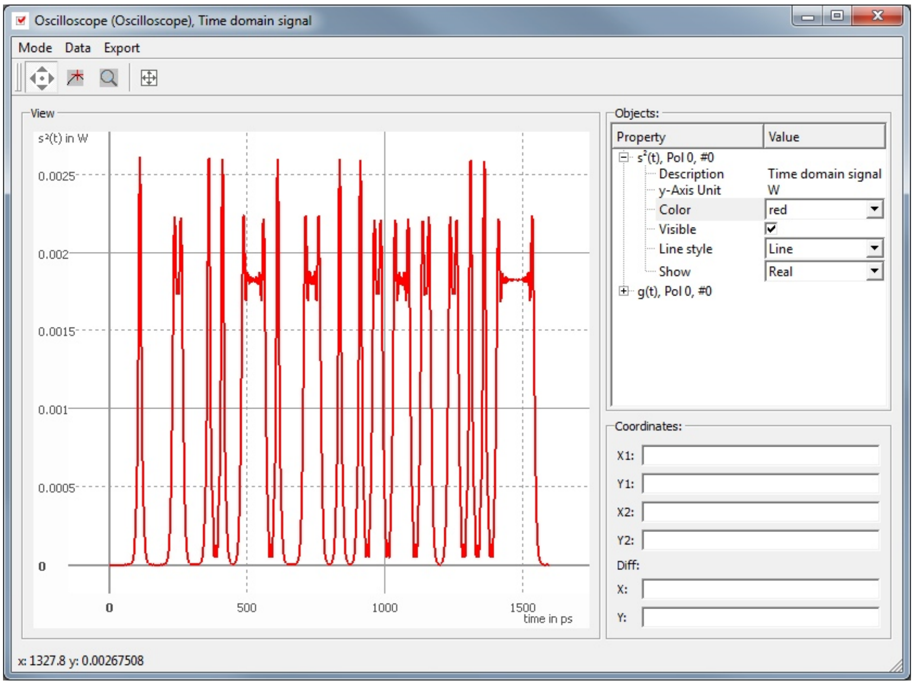


Figure 3.17.1. Right clicking on a component which includes visualizer(s) opens the Visualizer Context Menu to plot information.

The following remarks apply to all visualizers:

- No graphs or other results are available before the simulation has been started.
- While a simulation is running, only those graphs or other results are available which are provided by components that have already been processed.
- As some graphs or results are only calculated after they have actually been requested by the user to reduce the simulation time; it may take some time before they are shown.

3.17.3 Visualizer Controls



Property	Value
⊟ s²(t), Pol 0, #0	
Description	Time domain signal
y-Axis Unit	W
Color	red
Visible	☑
Line style	Line
Show	Real
⊞ g(t), Pol 0, #0	

X1:

Y1:

X2:

Y2:

Diff:

X:

Y:

x: 1327.8 y: 0.00267508

Figure 3.17.2. Visualizer Output for the Oscilloscope

Consider a typical visualizer window as depicted in figure 3.17.2. This window supports a number of operations that are accessible via the following toolbar buttons (or the corresponding menu entries). The buttons left to the separator correspond to three different user interface modes. There is always one mode active at a time. The following picture shows the toolbar while in *navigation mode*.

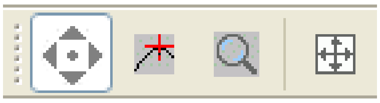


Figure 3.17.3. Navigation Toolbar

- **Navigate.** While in this mode dragging the view window moves the shown area and scrolling the mouse wheel zooms in and out.
- **Marker.** In this mode two markers can be set by clicking left or right. Left click sets the red marker, right click the green marker. The exact coordinates of these markers and their difference are shown in the Coordinates window.

- **Zoom in.** When active, clicking and dragging with the left mouse button will mark a rectangle to enlarge. After you release the mouse button, the marked area will be displayed in the full window space. After using this mode the graph window switches automatically to navigation mode.

- **Zoom out.** Show the entire graph.

The underlying data can be exported to MATLAB®, to a text file and to a number of image formats via the corresponding menu item.

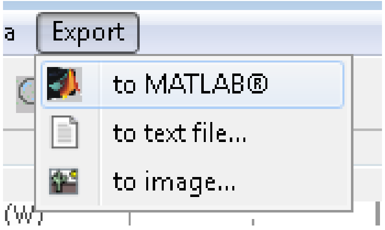
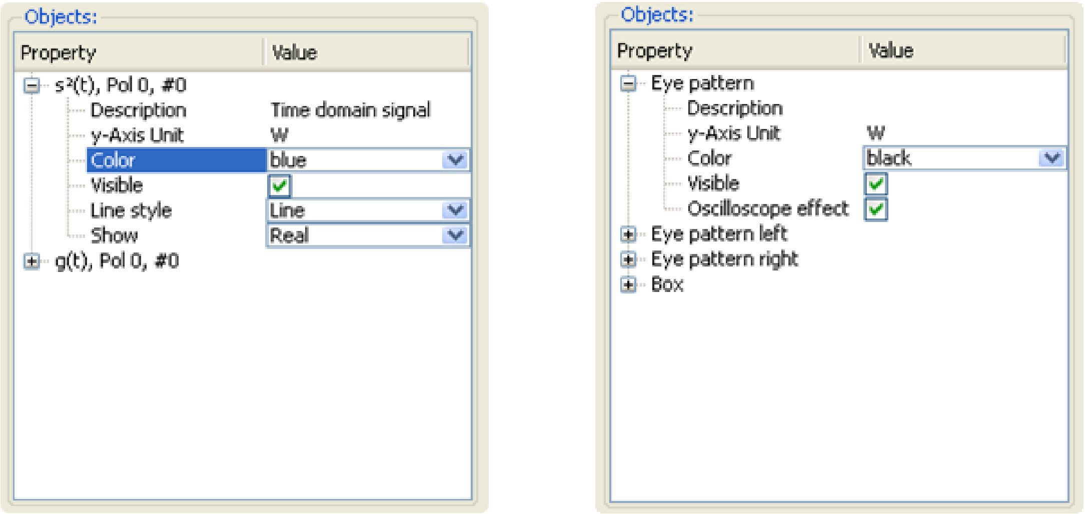


Figure 3.17.4. Export Menu

- **MATLAB®** opens a new MATLAB® context, transmits the data to it and creates a figure showing the same area as seen in the View window.

- **Text file** writes every visible data object to a text file containing one column for X values and one column for Y values separated by tabulator.

- **Image** writes a screen shot of the View window to a file. Several image file formats a supported (BMP, JPEG, PNG, PPM, TIFF, XBM, XPM).



Objects:

Property	Value
⊟ s²(t), Pol 0, #0	
Description	Time domain signal
y-Axis Unit	W
Color	blue
Visible	☑
Line style	Line
Show	Real
⊞ g(t), Pol 0, #0	

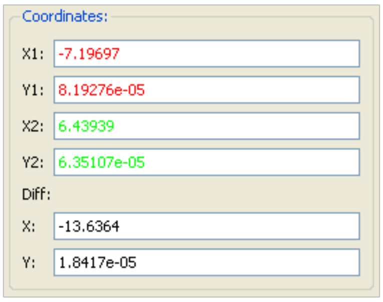
Objects:

Property	Value
⊟ Eye pattern	
Description	
y-Axis Unit	W
Color	black
Visible	☑
Oscilloscope effect	☑
⊞ Eye pattern left	
⊞ Eye pattern right	
⊞ Box	

Figure 3.17.5. Objects Window

Each graphical object that is displayed has its own item within the *Objects window* (see the right side of figure 3.17.2 - figure 3.17.5 shows different settings for this part of the main window). These items allow showing certain aspects of the object and modifying their appearance.

The sub items *Description* and *y-Axis* Unit display information about the corresponding data object. *Color*, *Visible*, *Line style* and *Show* are user editable properties. *Oscilloscope effect* is another editable property only available for certain objects, as for example the *Eye pattern*. When set to *true* the color of a pixel depends on the number of lines crossing that pixel.



Coordinates:

X1: -7.19697

Y1: 8.19276e-05

X2: 6.43939

Y2: 6.35107e-05

Diff:

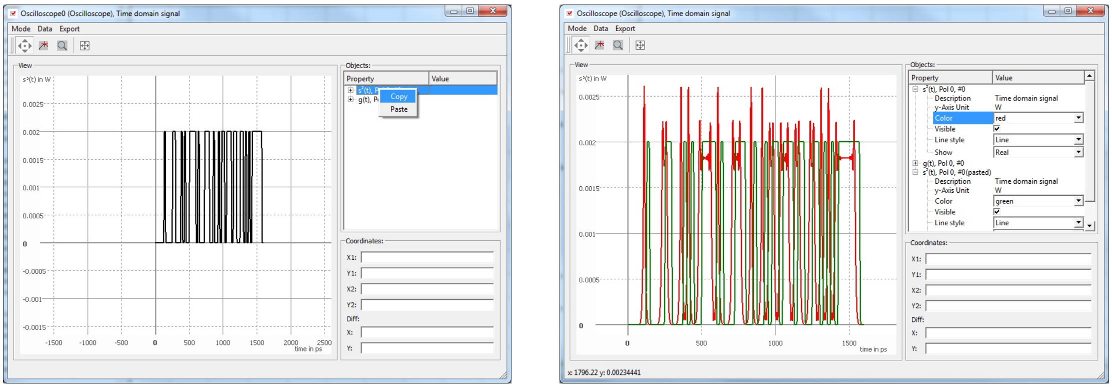
X: -13.6364

Y: 1.8417e-05

Figure 3.17.6. Coordinates Area

The *Coordinates* sub window shows the position of the two markers and the X and Y difference between their positions (see figure 3.17.6).

3.17.4 Copying Visualizer Data



Objects:

Property	Value
s²(t), Pol 0, #0	
Description	Time domain signal
y-Axis Unit	W
Color	red
Visible	☑
Line style	Line
Show	Real
g(t), Pol 0, #0	
s²(t), Pol 0, #0(pasted)	
Description	Time domain signal
y-Axis Unit	W
Color	green
Visible	☑
Line style	Line

Figure 3.17.7. Left: Visualizer data can be copied by right-clicking on the desired object and selecting *"Copy"*. Right: Data of two visualizers: An undistorted signal (green) and a signal which has been distorted by dispersion (red).

It is also possible to copy data from one visualizer graph to another - e.g. for comparing oscilloscope data before and after adding a signal distortion, etc. Simply select the data you want to copy by clicking on it in the objects dialog. You can either select the command "*Data* ⇒ *Copy*" from the menu or by right-clicking on the data in the objects dialog and selecting *"Copy"* (left of figure 3.17.7).
Insert the data by opening any other visualizer and select the command "*Data* ⇒ *Paste*" from the menu or right-click on any data in the objects dialog and select *"Paste"*. The data will now be added to the visualizer and its property will get the suffix "(pasted)" to indicate the new data (right of figure 3.17.7).

3.17.5 Summary

- Visualizers can be used to produce graphs and diagrams to illustrate vector or matrix style component results (e.g. an eye diagram or a plot of the signal amplitude versus signal time)

- Visualizers are always directly contained in a component or in a simulation link
- Visualizers can be opened as soon as the signal for the current component or simulation link has been calculated
- Visualizer data is usually calculated in the scope of the current block (Exception: the eye diagram is constructed over all blocks of a multiple block simulation)
- Each component has its own visualizer
- For comparison of visualizer data, the data may be copied between visualizers
- Clearing a simulation will also erase the visualizer data
- Normally, visualizer data will only be calculated / drawn on request by the user to save computational time

3.18 Networks

3.18.1 Underlying Concept

Normally, a PHOTOSS component is considered to be a black box with a fixed set of parameters, in- and out-ports and connections to other components (see section 3.4). More complex simulations naturally tend to contain a larger number of components. Once a certain degree of complexity is reached, it may be beneficial to group some of those components together and encapsulate them in a "Network". The "Network" is a special component that may be opened and may contain *other* components and links inside itself.

In general, a network component may be configured to hold any number of components inside and allow for any number of in-ports or out-ports. It is even possible to place a network (with components) *inside* another network, thus creating a "sub-network" in a lower hierarchy level, etc. The main advantage using networks is that they can also be copied easily (each component is copied along with it).

3.18.2 The Network Hierarchy

Throughout this manual you will sometimes come across the term "main network" or "root network". In these cases, the uppermost level on the simulation grid is addressed. Even when you have *not* used any network component, all components on the highest hierarchy are considered to be part of the "main network". This expression is, however, only theoretically motivated and does not influence the behavior of PHOTOSS.

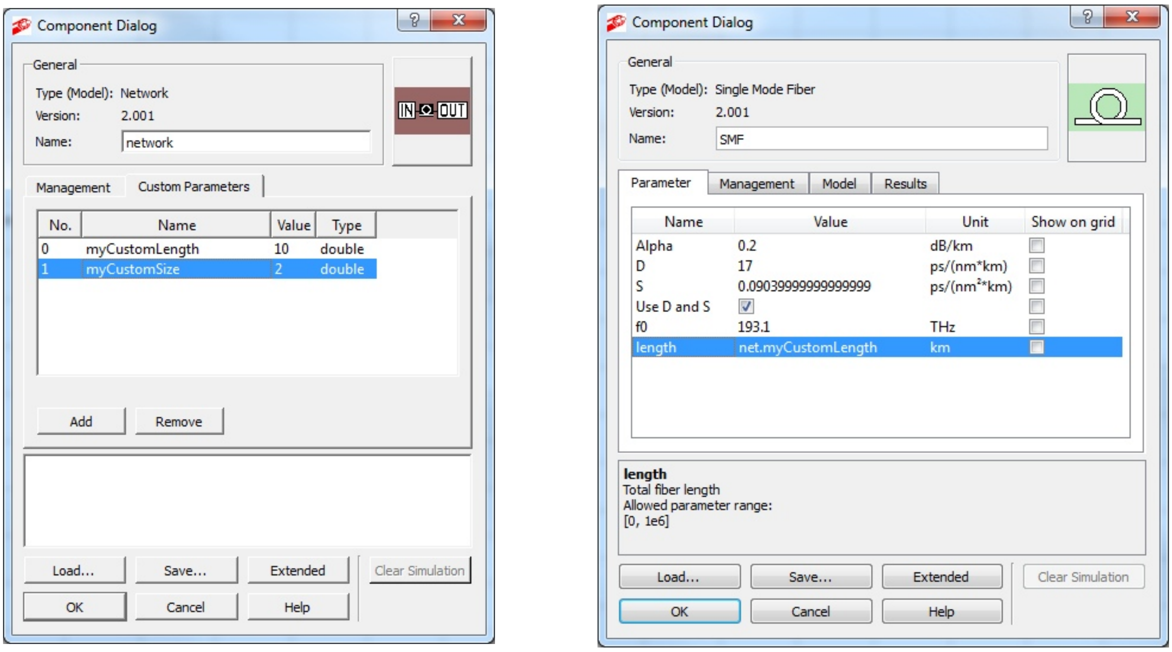
3.18.3 Opening Networks

To open a network, simply right-click on it and call up the context menu entry "Open network or iterator". Alternatively, you can use the hot key "F8" when a network has been selected or you can also use the "Open Network" GUI button to do so. In each case, a new tab will be opened in the GUI which displays the content of the selected network. To go back to your "root network" simply close the tab or switch back to the simulation tab. Of course, it is also possible to open multiple network tabs at the same time.

3.18.4 Creating and Using Custom Network Parameters

Another feature of the network which can be used to effectively reduce the complexity of a simulation is that you can define your own set of local parameters. These parameters work very much like custom simulation parameters (see section 3.3) but they are only accessible locally: Only components which reside *inside* the network component with the defined parameters may access these local variables. This includes all components which reside in sub-networks even further down in the network hierarchy, so local variables are propagated "down the hierarchy".

To create your own custom network parameters, click on the component dialog of the network component and switch to the "Custom Parameters" tab (see left of fig. 3.18.1). Click "Add" to generate a new custom parameter and assign a value and custom name to it. You can now go to any component inside your network and open its component dialog (see right of fig. 3.18.1). If you want to access the custom parameter of the network, use the prefix `net.` and add your custom parameter name. You can also access custom parameters of networks "higher up the hierarchy" by cascading the `net.` prefix. If, for example, you want to address a custom parameter named `mySize` of a network two levels above you would use the expression `net.net.mySize`.



Type (Model): Network
Version: 2.001
Name: network

No.	Name	Value	Type
0	myCustomLength	10	double
1	myCustomSize	2	double

Type (Model): Single Mode Fiber
Version: 2.001
Name: SMF

Name	Value	Unit	Show on grid
Alpha	0.2	dB/km	□
D	17	ps/(nm*km)	□
S	0.09039999999999999	ps/(nm^2*km)	□
Use D and S	☑		□
f0	193.1	THz	□
length	net.myCustomLength	km	□

length
Total fiber length
Allowed parameter range:
[0, 1e6]

Figure 3.18.1. Left: Custom Parameters dialog of a network component. Right: Using a custom parameter in a component inside the network

3.18.5 Summary

- Networks may hold any number of components, even further iterators or networks
- Networks may define local parameters which appear global to all components inside the network
- Addressing the local variables from sub-components can be done by using the prefix `net.`
- Addressing the local variables from sub-components on even lower levels can be done by cascading the prefix `net.`

3.19 Iterators

3.19.1 Underlying Concept

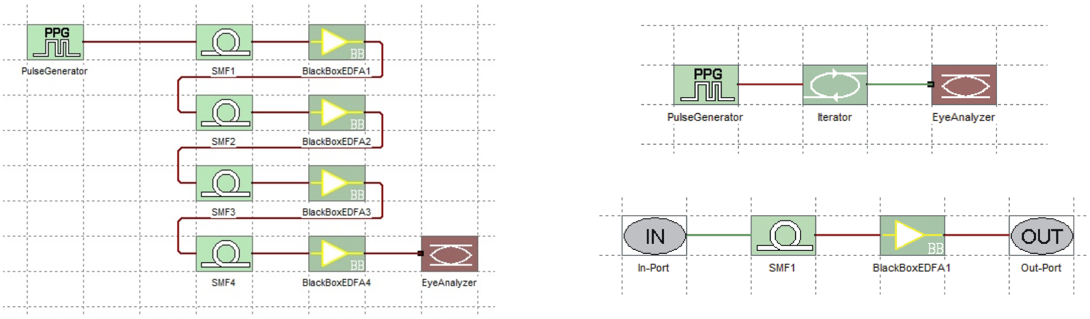


Figure 3.19.1. Left: Setup with multiple spans without using an iterator. Upper right: Same setup with using a horizontal iterator of size four. Lower right: Contents of iterator.

The "Iterator" component has all the features of a network component (see section 3.18). It also has some additional features: Normally, when a simulation is run, the signal is propagated through each component only once (or once per block). Now consider a system setup (left of fig. 3.19.1) in which certain sequences of components with identical parameter settings occur very often, for example a single mode fiber and amplifier span in a long-haul transmission system. The larger the number of spans, the greater the number of components. As simulations with a large number of components may get confusing, the iterator offers the ability to only put one span on the grid inside the iterator component (as it acts like a network) but to also propagate the signal through them as often as desired. The setup on the right of fig. 3.19.1 is equivalent to the setup of the left side of the figure by using a horizontal iterator of size five.

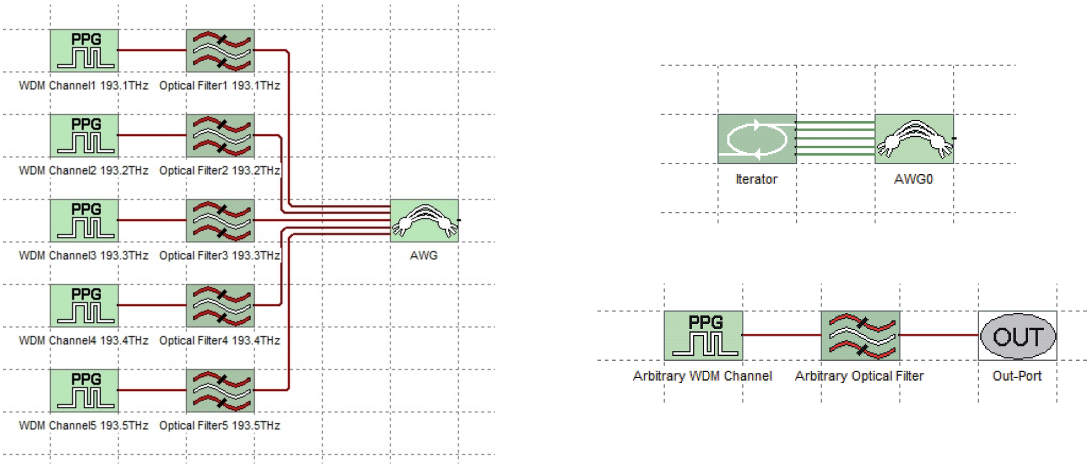


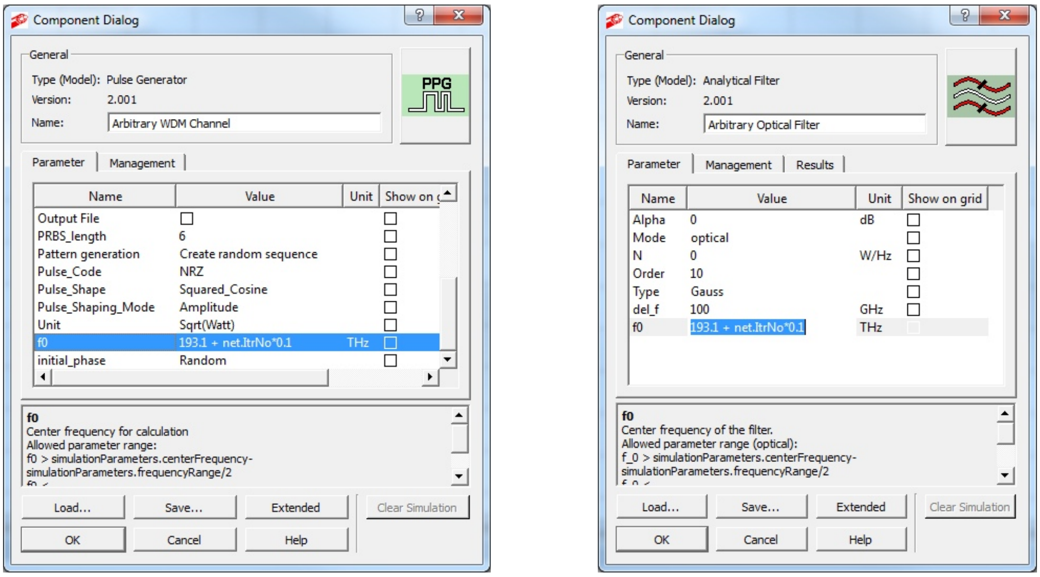
Figure 3.19.2. Left: Setup with multiple sources without using a vertical iterator. Upper right: Same setup with using a vertical iterator of size five. Lower right: Contents of iterator.

A *horizontal* iterator describes a component which is run once "from left to right" per chosen iteration number or "size" of the iterator. Choosing a *vertical* works quite differently: In this case, the signal is propagated through the components of the iterator only *once* but "size" sets of components are executed *in parallel*. This difference can be understood by looking at figure 3.19.2: On the left side, a WDM system setup with five signal sources and filters is shown where each signal source and filter operates on a certain, fixed wavelength. All five sources are multiplexed together using an "Arrayed Waveguide

Grating" (AWG). The right hand side shows an equivalent setup using only one arbitrary signal source and filter inside a vertical iterator of size five (equivalent to five out-ports) and zero in-ports.

You will notice, however, that you still have to define the (arbitrary) center frequencies for the source and the filter for each vertical "iteration" in the iterator. This can be done by writing a formula which incorporates the current "iteration number" for each out-port, e.g. you would want the first out-port of the iterator to supply a signal centered at 193.1 THz, the second one at 193.2 THz and so on. In order to do this, a the formula depicted in figure 3.19.3 can be used for the center frequency parameters of the source and the filter. As you can see, this arbitrary generation of frequencies only works if the frequencies are *evenly* spaced - if this is not the case, the vertical iterator can not be used to simplify your setup.

Please be advised that the iteration number `net.ItrNo` is counted beginning from *zero*.



General

Type (Model): Pulse Generator
Version: 2.001
Name: Arbitrary WDM Channel

Parameter | Management

Name	Value	Unit	Show on g
Output File	□		□
PRBS_length	6		□
Pattern generation	Create random sequence		□
Pulse_Code	NRZ		□
Pulse_Shape	Squared_Cosine		□
Pulse_Shaping_Mode	Amplitude		□
Unit	Sqrt(Watt)		□
f0	193.1 + net.ItrNo*0.1	THz	□
initial_phase	Random		□

f0
Center frequency for calculation
Allowed parameter range:
f0 > simulationParameters.centerFrequency-simulationParameters.frequencyRange/2

General

Type (Model): Analytical Filter
Version: 2.001
Name: Arbitrary Optical Filter

Parameter | Management | Results

Name	Value	Unit	Show on grid
Alpha	0	dB	□
Mode	optical		□
N	0	W/Hz	□
Order	10		□
Type	Gauss		□
del_f	100	GHz	□
f0	193.1 + net.ItrNo*0.1	THz	

f0
Center frequency of the filter.
Allowed parameter range (optical):
f_0 > simulationParameters.centerFrequency-simulationParameters.frequencyRange/2

Figure 3.19.3. Formula to generate evenly spaced center frequencies for each vertical iteration for the source component (left) and the filter (right).

Just like network components, iterators may also be placed inside iterators for creating even more complex setups. Access to the local variables of the iterator works the same way by using the `net.` prefix. The variables `net.ItrSize` and `net.ItrNo` are always present. Further information about the iterator can be found in section 7.12.3.

You should also be aware that a large number of iterations or a large number of components inside an iterator may pose serious demands on your machine's memory. Too complex simulations might lead to an "Out of memory" error which will be displayed in the PHOTOSS console. In these cases it might be beneficial to use the "File Loader" and "File Saver" components to further divide your simulation.

3.19.2 Summary

- Iterators offer all the features of a network component
- Iterators can be divided in horizontal and vertical iterators
- Iterators may hold any number of components, even further iterators or networks
- The content of horizontal iterators is scheduled to be calculated consecutively

- The content of vertical iterators is scheduled to be calculated in parallel
- Iterators are capable of storing component results for each iteration
- Iterators may define local parameters which appear global to all components inside the iterator
- Addressing the local variables from sub-components can be done by using the prefix `net.`
- The following local variables are always present inside an iterator: `net.ItrSize` and `net.ItrNo`; the iteration number is counted beginning from zero for the first iteration.

3.20 Interfaces and Import/Export

3.20.1 Underlying Concept

As explained above in section 3.1, a simulation holds the *basic* information to run a PHOTOSS simulation: The components (and their links) as well as the simulation parameters. However, in some cases it may be helpful to create numerical simulations which are considered in a broader context than just one single simulation file. There are several use cases where a single simulation file will be augmented by additional files; this section will list a few of these use cases and explain the benefits which can be gained by using the interfaces PHOTOSS will offer you to import and export data from/to a simulation.

3.20.2 Saving and Loading Signals to/from Files

Consider a generic simulation consisting of a signal source, a channel (e.g. a span containing a single mode fiber and an amplifier) and a receiver structure. It may be of interest to use a designated source and channel but to compare different receiver setups (e.g. by using different components or algorithms). The most simple approach would be to copy the simulation file and to exchange the receiver component(s) for each simulation and run both simulations separately to compare the results. This approach of course is possible, but it may be cumbersome and error prone - especially when statistical simulations are considered.

A better way would be to put a “File Saver” component (see section 7.12.2) at the end of the last span and to save the whole signal information to a file. Now, you can create two separate simulation files (one for each designated receiver structure) and to use a “File Loader” (see section 7.12.1) to reload the signal into the simulation file. Now, the same input signal for both receivers only had to be computed once and the signal file can even be stored and copied if you should decide to use a third receiver structure at a later time.

3.20.3 Modifying Signals by using MATLAB®

In some cases, a user may want to create his or her own components which can be used with PHOTOSS. A very simple way to achieve this aim is to write MATLAB® code and to create a module which can be inserted in the MATLAB® interface to interact with PHOTOSS. This approach will be described in more detail in section 4.7.

Another approach may be to not create a new MATLAB® component to interact with PHOTOSS, but to *directly* access and process a signal file which has been created by using a “File Saver” (see above). In this case, MATLAB® can also be used and it is very easy to import the signal to the MATLAB® workspace by saving the PHOTOSS signal in *.mat* format which is also supported by the “File Saver”.

3.20.4 Modifying Signals by using the Command Line Interface

Another way to modify PHOTOSS signals is to use the “Command Line Interface” (see section 7.11.2). It offers a functionality similar to a “File Saver” and some additional features: You can specify any command line call to an application of your choice which will be called once the component has been reached while running the simulation. The external program can then modify the signal file and after its calculation has finished, the simulation will resume.

3.20.5 Importing and Exporting Vectors and Matrices

Several components in PHOTOSS allow for the import or export of specific data. For example you might want to supply your own split step positions for processing a nonlinear fiber component or import your own bit sequence in a source component such as the “Pulse Generator”. In these cases, the data is always imported (or exported) by using .txt files.

In the most simple case, the text file only contains a vector of double or integer values - one value in each line. This format is used e.g. to import/export bit sequences. Some components use a more complex set of parameters or results - for example the equalizer components or PMD Path components such as the “Transmission Matrix Analyzer”. In these cases, a more sophisticated format can be used to supply all parameters in one single file. If you are not sure on which parameters have to be included in the file for the specified component, it is always a good idea to let the component write its parameters to a file first to get an overview of the required structure.

To define a vector in your input file, you should create it like this:

```
NameOfVectorToBeImported
Number Of Rows = n
% Any comment you might wish to add
Value_1
Value_2
Value_3
...
...
...
Value_n
```

The name of the vector which should be imported must correspond to the name which is required by PHOTOSS, according information, how the vector must be named can also be found by looking at the tooltips of the according input parameter in the component dialog. “Number of Rows” specifies the number of data entries the vector should have, the following n values will be read from the file to fill the vector.

An input matrix can be created in a very similar way:

```
NameOfMatrixToBeImported
Number Of Rows = n
Number Of Columns = m
% Any comment you might wish to add
Value_for_row1_column1  Value_for_row1_colum2 ...
Value_for_row2_column1  Value_for_row2_colum2 ...
Value_for_row3_column1  Value_for_row3_colum2 ...
...
...
...
Value_for_row_n_column1 Value_for_row_n_colum2  ...
```

In this case, m represents the number of columns of the matrix. Any combination of “*n* rows and *m* columns” is allowed but the number of columns may not vary over one matrix (vector<vector> equivalents are not allowed).

3.20.6 Summary

- Simulation files can be loaded, modified and saved in the *.pho* format
- PHOTOSS signals can be saved using a File Saver and loaded using a File Loader component

- MATLAB® interface components can be utilized to create custom components and to directly modify the PHOTOSS signal
- The Command Line Interface can also be utilized to modify the signal by using text format files
- Several components require text format input files or offer saving their parameters or results to text files
- Importing vector and matrix data is also possible and supported by several PHOTOSS components

3.21 Random Number Generation

3.21.1 Underlying Concept

Modeling of optical transmission systems often incorporates theories and models which draw on statistical processes to explain physical phenomena. Prominent examples are analytical or numerical noise models (see section 3.9) or polarization effects. In order to emulate these processes, PHOTOSS is equipped with a generator that can create random numbers according to a chosen statistical distribution. Aside from the separate PScript random generator (you can find more details about it in the PScript manual), a random generator always applies to the scope of a *single simulation* and its settings are determined by the simulation parameters.

The random generator offers a variety of statistical distributions which are used internally in components. It is also possible to access the random generator externally, for example to derive values for a component parameter randomly or to incorporate random numbers in a formula for a component parameter. Access to these functions can be gained by opening the component parameter editor and clicking on "Overview" (see fig. 3.5). This overview includes a full list of all available distribution functions.

Three basic operational modes of the random generator can be chosen in the simulation parameter dialog:

1. **Random number generator initializations - deterministic:** All random generators are initialized by the same deterministic seed. A simulation involving random processes will yield exactly the same results each time it is run. In practice, this option provides an easy means of making results exactly reproducible. Different blocks of a single simulation will generate different, but deterministic random numbers to ensure that multi block simulations can still be executed and generate a different numerical noise representation for each block. After all blocks of a simulation run have been completed, the generator will be reset so all runs of a parameter variation will receive the *same* set of random variables. The seed of the simulation can be changed manually, e.g. for the purpose of creating different, yet deterministic simulation setups.

2. **Random number generator initializations - once deterministic:** This mode is intended for reproducible Monte Carlo simulations and works identical to the deterministic setting save for one exception: All random generators are initialized with a deterministic seed, but the random generator seed is *not* reset for each run of a parameter variation. The seed of the simulation can be changed manually, e.g. for the purpose of creating different, yet deterministic simulation setups.

3. **Random number generator initializations - statistical:** Random generator seeds are automatically generated from pseudo-random values such as the system time or CPU clock ticks. If this setting is chosen, random processes will yield different results each time the simulation is run and there is no correlation between the random numbers of multiple simulation runs. In this mode, the seed will always be chosen automatically and can not be set by the user.

The seed for the random generator can be changed by adjusting the corresponding simulation parameters "random_generator" and "randomGeneratorSeed".

3.21.2 Summary

- PHOTOSS has a built-in Random Number Generator

- The generator has three basic working modes which influence the determination of the generator's seed.

- The “Deterministic” mode sets the very same seed each time a simulation run is executed (different blocks will be assigned different but deterministic random numbers during this run in a multi block simulation)
- The “Once Deterministic” mode may be used to create deterministic Parameter Variations (Monte Carlo). The seed will be reset to a deterministic value for each simulation run of the Parameter Variation
- The “Statistical” mode will randomly determine a different seed each time a simulation is run and represents the Standard Monte Carlo approach
- The deterministic seed of the generator is hard-coded but can be changed using PScript

3.22 PMD Path

3.22.1 Underlying Concept

In optical transmission systems, the transmission span often consists of a single or multi mode fiber. Ideally, the cross-section fiber should be perfectly symmetrical. In reality, however, the fiber geometry suffers from slight deviations from the ideal symmetry due to fiber bending and mechanical vibration or temperature changes in the environment. In these cases, the fiber is set to be a birefringent material and statistical physical effects like polarization mode dispersion (PMD) or polarization-dependent loss (PDL) may occur which may distort the signal.

PHOTOSS offers a wide variety of components which can emulate PMD effects such as the single mode fiber with an integrated wave plate model (see section 7.3.1), the PMD and PDL emulator (see 7.7.5) and many more. When considering these effects, one is often not only interested in the signal distortions but also in the statistics of PMD and PDL which create these distortions.

PMD and PDL effects can be fully described when the *Jones Matrix* of the optical transmission path is known - the matrix allows for the derivation of interesting observables such as the differential group delay (DGD) or second order polarization effects such as polarization-dependent chromatic dispersion (PCD) or depolarization (DEP). These observables can be obtained by putting the receiver component “Transmission Matrix Analyzer” (see section 7.7.10) in your transmission system.

Generally, this component behaves like all other PHOTOSS components and offers the observables in the form of results once a simulation run has been completed. There is, however, one difference:

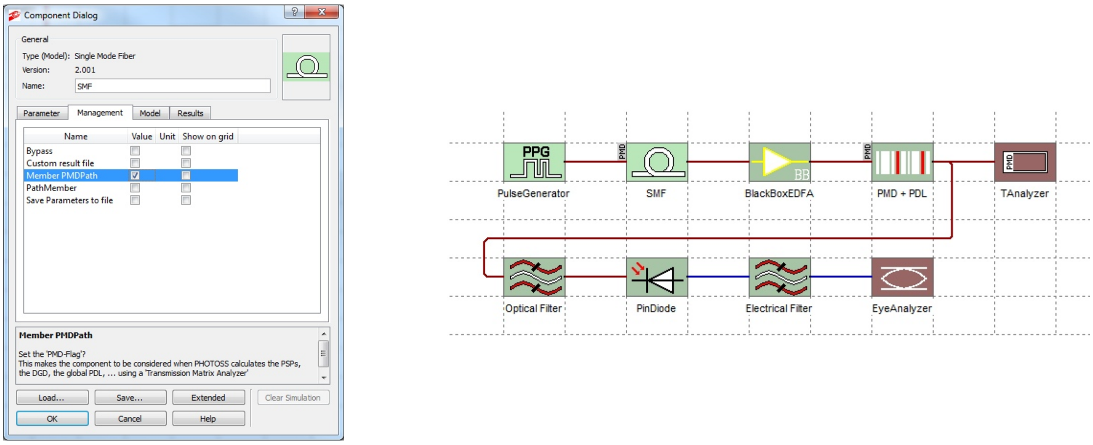
While normally, all relevant simulation data is propagated through the components with the MultiSignal through the “links” in the system (see 3.6), this is *not* the case with the Jones Matrix. Instead, this information resides on a different path level - the so-called “PMD Path”. All components which can influence the Jones Matrix of an incoming signal, can be members of the “PMD Path” and each component member of the path adjusts the Jones Matrix accordingly once it is calculated during the parameterized signal run.

3.22.2 Limitations of the PMD Path

The PMD path construct can only be used to setup *one* PMD path per simulation setup - depending on which components are members of the path. This path does not necessarily have to be identical to the path that the MultiSignal will take through the component links. Instead, it is only dependent on the components which are its members and on the *order* in which the components are processed. Normally, this order would be from left to right through the components but as explained in section 3.14 the order is determined by the PHOTOSS scheduler and is itself dependent on a number of various factors.
As a result, only one “Transmission Matrix Analyzer” may be used per simulation to avoid confusion or mixing of different Jones Matrices in the same system.

3.22.3 Joining or Leaving the PMD Path

Components can only join the PMD Path, if the corresponding built-in simulation parameter “Include polarization effects” (see 3.3.2) has been activated. You can then open a component which supports the path membership and switch to the “Management” tab. Now check or uncheck the parameter “Member PMDPath” to join or leave the PMD Path. A small PMD flag icon will appear on the upper left corner of the component image (see figure 3.22.1) to denote its membership. You can also click on the “Hoist the PMD flag” and “Lower the PMD flag” icons in the GUI or use the hot keys “CTRL + F9” and “CTRL+F10” to do so.



Component Dialog

General

Type (Model): Single Mode Fiber

Version: 2.001

Name: SMF

Name	Value	Unit	Show on grid
Bypass	□		□
Custom result file	□		□
Member PMDPath	☑		□
PathMember	□		□
Save Parameters to file	□		□

Member PMDPath

Set the 'PMD-Flag'?
This makes the component to be considered when PHOTOSS calculates the PSPs, the DGD, the global PDL, ... using a 'Transmission Matrix Analyzer'

Figure 3.22.1. Left: Activate or deactivate PMD path membership by switching to the management tab and selecting “Member PMDPath”. Right: In this setup, the components “Single Mode Fiber” and “PMD and PDL Emulator” are members of the PMD Path. A “Transmission Matrix Analyzer” is used to calculate the Jones Matrix of the transmission system.

Currently, the following components support PMD Path membership:

- Single Mode Fiber
- Birefringent Element
- PDL Element
- PMD + PDL Emulator
- PMD Emulator
- Polarization Controller (in “Adjust SOP-Mode”)
- The Transmission Matrix Analyzer is *always* part of the PMD Path.

3.22.4 Carrying out quick PMD Path Calculations

Since all PMD calculations are carried out in the parameterized signal run (see section 3.10) and the simulation of a sampled signal is not necessary to obtain information about the Jones Matrix of the optical transmission system, you can greatly speed up the simulation by de-selecting the simulation parameter “Signal representation - Sampled” (see 3.3.2). This mode is not meant for calculation of PMD and signal interactions but solely for determining the systems’ Jones Matrix and the underlying PMD statistics in e.g. Monte Carlo simulations with many simulation runs. If you want to also create and observe PMD-induced signal distortions in the sampled signal, the parameter “Signal representation - Sampled” has to be activated.

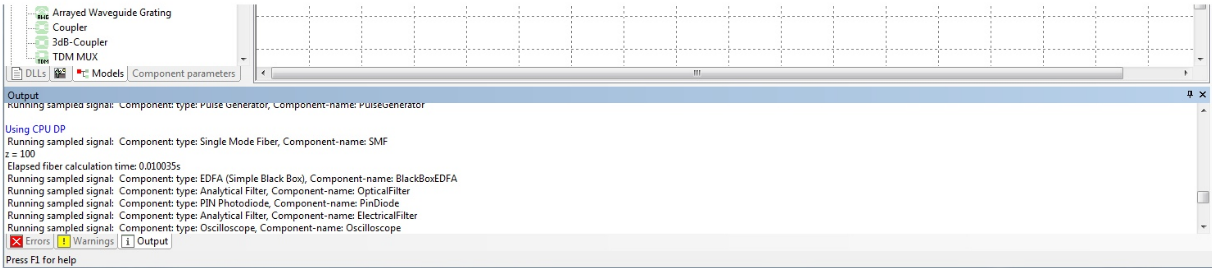
3.22.5 Summary

- The PMD path is used to describe the corresponding Jones Matrix of a transmission system to calculate PMD effects
- A PMD path is comprised of one or more components which have set their PMD path membership to “true”
- After a simulation, the component Transmission Matrix Analyzer can be used to evaluate the resulting Jones Matrix of the path

- The path has no visible representation (e.g. via links) in the GUI but components which are path members have a “PMD Path” flag
- Components are incorporated in the path in the same order as they are calculated by the scheduler
- There may only be one Transmission Matrix Analyzer per simulation; consequently, only one PMD Path should be used to avoid confusion
- Quick statistical PMD calculations can be carried out without having to simulate a sampled signal

3.23 Logs and Consoles

3.23.1 Underlying Concept



```
Running sampled signal:  Component: type: Pulse Generator, Component-name: PulseGenerator

Using CPU DP
 Running sampled signal:  Component: type: Single Mode Fiber, Component-name: SMF
z = 100
 Elapsed fiber calculation time: 0.010035s
 Running sampled signal:  Component: type: EDFA (Simple Black Box), Component-name: BlackBoxEDFA
 Running sampled signal:  Component: type: Analytical Filter, Component-name: OpticalFilter
 Running sampled signal:  Component: type: PIN Photodiode, Component-name: PinDiode
 Running sampled signal:  Component: type: Analytical Filter, Component-name: ElectricalFilter
 Running sampled signal:  Component: type: Oscilloscope, Component-name: Oscilloscope
```

Figure 3.23.1. Logout window

Simulations of optical transmission systems sometimes tend to get very complex and it is difficult to get an overview about all the things that have to be kept in mind when designing these systems. Physical distortion effects may tend to cause the signal quality to degrade - sometimes even stronger than expected by the user who sets up a simulation. PHOTOSS has been designed to give you feedback to the overall simulation status to help you keep track of all the important information in your simulation. Thus, all PHOTOSS components and the simulation network itself create log files (per default) which enable you to track down potential problems or causes of errors in your simulation setup.

The PHOTOSS main window contains a log section (see figure 3.23.1) which is divided in three sections:

- Output: Shows the standard output of the PHOTOSS log. Each component which is processed can add helpful information to this log regarding its calculation and status.
- Warnings: Some component and simulation parameter settings may cause a component to issue a warning. This tab displays the warning message for each component. It is always prudent to closely check whether the warning is actually a hint to a potential error in the simulation setup or a false parameter setting or if the behavior is both desired and intended by the user.
- Errors: Severe problems which occurred before or during a simulation run are displayed in this log tab. If an error occurs during a simulation run, the simulation is immediately stopped.

All three components of the logout information can also be saved to a log file (default). You can also choose to save only specific information of one or more categories to the log file by adjusting the PHOTOSS option (see section 2.9). Since log files tend to get very large (several gigabytes) for complex simulations with many components and large parameter variations, it is often a good way to run a simulation once (no variation) to check whether any errors, warnings or problems might arise by using the log information. Once you are certain that the setup works to your liking, the log can be deactivated to save time and hard disk space.

3.23.2 Summary

- PHOTOSS offers the creation of a log file (default) to keep track of the simulation progress and status
- logs are categorized in three sections: Output, Warnings and Errors
- logout information can also be seen in the main window of the GUI

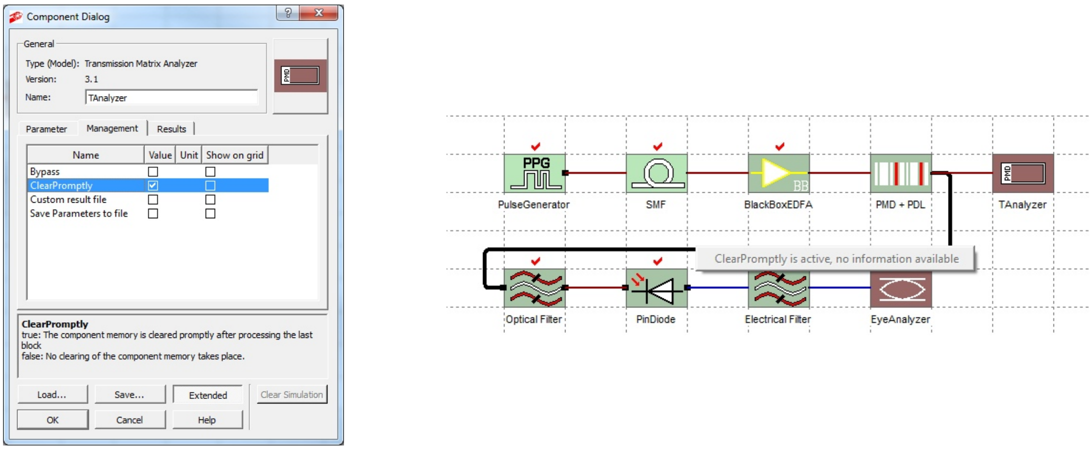
- logs of large and complex simulations tend to get very big - consider deactivating the writing of log files for these simulations

3.24 Memory Management

3.24.1 Underlying Concept

The simulation files for complex optical transmission systems tend to contain a large number of components and simulation "links" between them. In the normal simulation mode, PHOTOSS has to allocate the memory for each simulation link and component for the whole duration of the simulation. Thus, the larger the simulation band and the larger the number of links and components, the greater the demand on the machine's memory. When simulating complex WDM systems with a lot of channels and large iterator components, several gigabytes of system memory may be needed to allocate the memory. Fortunately, there are several ways to reduce the needed memory for complex simulations:

The first approach is to use multi block simulations (see section 3.11) to reduce the number of bits per block and thus the memory required by each sampled signal channel.



Component Dialog

General

Type (Model): Transmission Matrix Analyzer

Version: 3.1

Name: TAnalyzer

Parameter | Management | Results

Name	Value	Unit	Show on grid
Bypass	□		□
ClearPromptly	☑		□
Custom result file	□		□
Save Parameters to file	□		□

ClearPromptly
true: The component memory is cleared promptly after processing the last block
false: No clearing of the component memory takes place.

ClearPromptly is active, no information available

Figure 3.24.1. Left: Activate or deactivate "Clear Promptly" by switching to the management tab, clicking on "Extended" and selecting "Clear Promptly". Right: In this setup, the components "PMD + PDL Emulator" and "Eye Analyzer" have activated clear promptly. Right clicking on a link will not show information of the channel data since the channel has been erased to save memory.

The second approach is to use the "Clear Promptly" functionality of PHOTOSS. All PHOTOSS components may activate or deactivate (default) their "Clear Promptly" status. This can be done either by clicking on the corresponding icons in the GUI when a component has been selected or by opening the component parameter dialog, switching to the "Management" tab, selecting "Extended" and changing the "Clear Promptly" parameter (see figure 3.24.1).

There is also a way to quickly activate "Clear Promptly" for *all* components in the simulation - open the PHOTOSS options dialog (see 2.9) and select the option "Force Clear Promptly in simulations". This will also override the component settings for this parameter.

When clear promptly is activated for a component, the following will happen once it has been processed:

1. Component Results will not be kept in memory.
2. All internal component data (vectors, matrices, etc.) will be erased from memory.
3. All outgoing links including the MultiSignal (parameterized and sampled signal information) will be erased from memory once the have reached the next component(s) on the grid.
4. All Visualizer Data of the component will be erased from memory.

Please be advised, that the parameterized signal will only be erased from the memory if the "Combined Simulation" mode has been activated in the built-in simulation parameters.

Activating the "Clear Promptly" option can save more than forty percent of memory in one-block simulations with a large number of samples per block and/or a high number of iterations in an iterator component. Although "Clear Promptly" can also be activated for multiple block simulations, it will only have effect in the scope of each single block of the simulation (e.g. for a high number of iterations in an iterator component) since all simulation links will be automatically cleared once a new block is being calculated.

3.24.2 Summary

- Very complex simulations tend to consume a lot of machine memory
- Memory requirements can be reduced by using multiple block simulations
- Memory requirements for single block simulations and simulations with iterator components can be greatly reduced by using the "Clear Promptly" functionality

3.25 GPU Processing

3.25.1 Underlying Concept

When considering complex optical transmission systems, the calculation of single mode fiber components often proves to take the most computational time of the simulation. This holds true especially, when nonlinear effects are enabled and a large number of split steps and/or a high input power has been selected. In these cases, simulations may often take several hours, days or even weeks. Needless to say that incorporating such complex simulations in large parameter variations would take even more time.

To greatly reduce the computational time necessary to process the single mode fiber component, PHOTOSS offers the usage of graphical processing units (GPUs). While current quad-core CPUs (e.g. Intel® Core i7 965 XE) achieve a peak performance of more than 70 GFlop/s, graphics cards (e.g. NVIDIA® GeForce® GTX 275 with 240 stream processors) already provide more than 1 TFlop/s. The reason is that the hardware of current GPUs allows for a large number of parallelized computations.

Some GPU devices are specifically designed to carry out complex numerical simulations. Currently, PHOTOSS supports the usage of NVIDIA™ GPU devices which have CUDA 1.3 compatibility (or greater). A list of all CUDA 1.3 compatible GPUs can be found on the web page of the vendor NVIDIA: http://developer.nvidia.com/cuda-gpus

If you have installed an NVIDIA® CUDA compatible graphics card(s) with compute capability 1.3 into your computer, you can select to use it for the calculation of the single mode fiber component(s) by using the appropriate PHOTOSS options (see chapter 2.9 on page 23). You can choose the option *Use GPU for fiber calculation*. Furthermore, you can choose from a list of graphics cards, which one to use for PHOTOSS calculation, if multiple graphics cards are installed in your computer. If *Use GPU for fiber calculation* is activated, PHOTOSS automatically uses the graphics card for time consuming simulation of the single mode fiber component. Depending on your setup and the selected number of samples per block in the *simulation parameters* a speed-up of more than a factor of 10 up to 180 compared to CPU-based simulations can be expected with comparable accuracy of the results [1].

3.25.2 Accuracy

A GPU device normally offers hardware for single and double precision processing of floating point numbers. While simulations with a single precision-setting in the PHOTOSS options dialog may be faster, the double precision-setting offers a higher accuracy. This accuracy may very well effect the simulation results and observables you are interested in so make sure to select the appropriate option for your simulation setup. If you are unsure which option is suitable for your setup, the following compromise may be of help:

It is possible to use the single precision setting for a less accurate but quick sweep of your parameter space. After you have identified the parameter settings which are most relevant for your interest, you can switch to double precision to start an accurate calculation for this sub-parameter set. In literature, such an approach has been called “Stratified Monte-Carlo Sampling”, see [2].

3.25.3 Limitations and Solutions

Although a high performance gain can be achieved when using GPU devices, some limitations are imposed: Most GPUs offer less working memory (“RAM”) than most state of the art desktop PCs. Very often, the GPU’s RAM is limited to 1 or 2 Gigabytes of memory. If your simulations are very complex and contain a large number of simulated samples per block, this memory may not be sufficient to carry out numerical simulations of the single mode fiber on the GPU as the *whole* sampled signal needs to be stored in the GPU memory. If your memory is not large enough, the GPU calculation will be aborted and an “Out of Memory”-error will be displayed in the PHOTOSS log. The best way to counteract this problem is to use fewer simulated samples per block and to use multiple block simulations (see 3.11) instead to keep the required RAM *per block run* well below the RAM of the GPU.

Another problem might arise when you want to execute several instances of PHOTOSS on the same machine: Currently, it is not possible to share a GPU between several PHOTOSS instances. It is, however, possible to use several PHOTOSS instances and several different GPUs at the same time by selecting a specific GPU device for each PHOTOSS instance.

In any case it should be noted that GPUs usually require a lot of power for their computations and can produce a lot of excess heat, especially when they are operating at full capacity. Make sure that your GPU fans are working correctly to avoid overheating effects or even possible damage to your hardware.

3.25.4 Summary

- GPUs can be used to greatly decrease the computational time necessary to calculate non-linear and linear effects of the single mode fiber component
- GPU calculations can be carried out with single precision (faster) or double precision (more accurate).
- Each PHOTOSS instance can use its own GPU device if multiple GPUs have been installed on the machine
- The memory of the GPU imposes certain limits on the maximal sampled signal size

References

[1] S. Pachnicke, A. Chachaj, M. Helf, P. Krummrich: *Fast Parallel Simulation of Fiber Optical Communication Systems Accelerated by a Graphics Processing Unit*, IEEE ICTON Conference, (2010)

[2] S. Pachnicke, A. Chachaj, C. Remmersmann, P. Krummrich: *Fast Parallelized Simulation of 112 Gb/s CP-QPSK Transmission System using Stratified Monte-Carlo Sampling*, Optical Fiber Communications Conference (OFC), Los Angeles, March 2011

3.26 PScript

3.26.1 Underlying Concept

PScript is an abbreviation for PHOTOSS Script. It is a script engine based on the ECMA-Standard. PScript can be used to embed a PHOTOSS simulation in an algorithmic context. Thus, it enables the user to control the PHOTOSS environment, using predefined, automated scripts which reside “outside” of the PHOTOSS simulation. PScript can help to drastically reduce the amount of “manual” work needed to create flexible, straightforward, and yet complex parameter variations.

You can access the PScript console by clicking on the PScript Icon or clicking on “Tools => PScript Console”. The default hot key for opening the PScript console is `CTRL+ALT+P`. Since PScript offers a broad range of features and functionality, a separate PScript manual is available with your PHOTOSS installation. Please refer to this document for a detailed explanation of all PScript features, functions and general information.

3.26.2 Summary

- PScript is a PHOTOSS scripting language
- It can be used to easily create complex parameter variations or to algorithmically create, modify or run simulations
- PScript offers full interaction of both the PHOTOSS as well as a MATLAB® workspace environment
- A complete list of all PScript functionality is available in the separate PScript manual.

4 Additional Features

4.1 Path Analysis

Overview

A path analysis lets you monitor how the transmission performance develops when a signal travels along a pre-defined *path*. A path may be composed of multiple subsequent network components. Instead of just evaluating the transmission performance in front of or behind certain components, a path analysis is capable of tracking performance measures even within widespread components such as optical fibers. You may for example analyze how the signal quality degrades or improves along a fiber in order to determine its optimum length.

Defining an Optical Path

To define the optical path to observe, select one or more components and press the ⚑ toolbar button (or the *Simulation/Set Path* menu entry) to add them to the path. As shown in the example picture, components which are path members are displayed with a small flag on top.

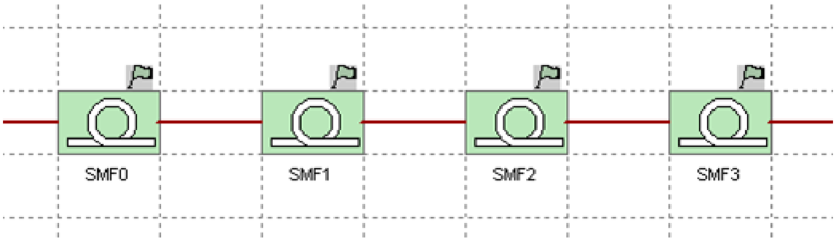


Figure 4.1.1. Path Analysis Setup

You may add further components to the path in the same way, or use the ☒ toolbar button (or the *Simulation/Erase Path* menu entry) to remove them from the path. You may also specify whether a component belongs to the path or not in its respective component parameter dialog: Open the *Management* page and set the *PathMember* list entry to *true* or *false*.

Notes:

1. Make sure to only add components to the path that actually pass **optical** signals.

2. Only fibers are considered with their actual length, while all other optical components are considered to be of zero length.

3. The spatial steps where the performance is evaluated within a fiber are set via the *StepSizePE* parameter on the *Management* page of the components' parameter dialogs. You may choose different step sizes for different components in the same path.

4. Make sure to only add components to the path that belong to the same continuous signal flow. Path members must be sequenced in a way that the same signal passes through them. You may leave out one or more intermediate components, but you may not e. g. add components from behind different outputs of a splitter device to the same path.

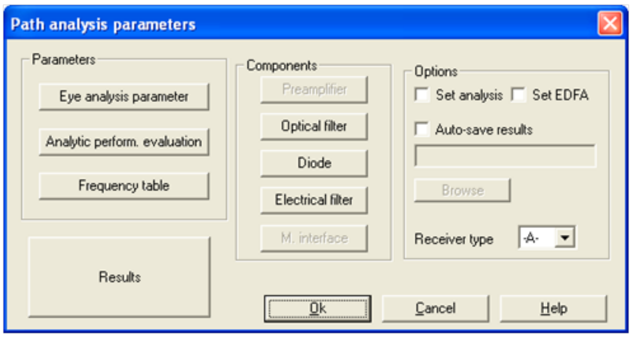


Figure 4.1.2. Path Analysis Parameters Dialog

Preparing a Path Analysis

Open the *Path analysis parameters* dialog with the toolbar button or *Simulation/Path analysis parameters* menu entry. Check *Set analysis* to activate the path analysis. The *Frequency table* button serves to select which WDM channels of the optical signal are to be included in the analysis. Please refer to section *Frequency Table* for details. The *Auto-save results* option will record all results calculated during the path analysis to a file called *<name of .pho file>_PathAnalysisResults.txt*. The results file is stored in the same directory as the *.pho* file.

Receiver type lets you select one of four receiver types (A, B, C, D) which are described in the next sections. The component parameters of the respective receiver components can be edited in the *Components* box. All receivers may optionally include an EDFA preamplifier (*Set EDFA* option). Receivers of type B, C and D include a programmable MATLAB® Interface component. In order to use these receiver types, you must have MATLAB® installed on your computer. Please refer to corresponding section in the component manual for details.

Receiver Types

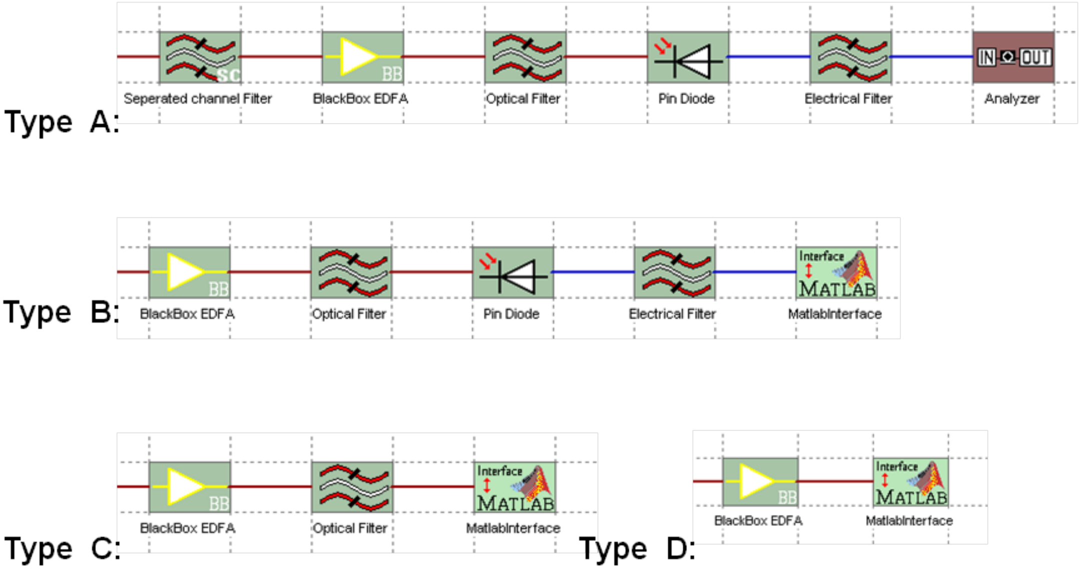


Figure 4.1.3. Receiver Types in PHOTOSS

Figure 4.1.3 depicts the receiver structures are available for a path analysis. Use the frequency table to select which frequencies to observe and which analyzers to include in type A receivers (see section *Frequency Table* for details). When using the *Separated channels* simulation method, type A receivers include additional SC Filter components with their respective *Frequency* parameters set to the selected WDM channels (as depicted above).

Example Setup

When a path analysis is carried out along a fiber, the selected receiver is fed with the optical signal behind every spatial step within the fiber. Use the *StepSizePE* parameter in the fiber's component parameter dialog (*Management* page) to adjust the step size.

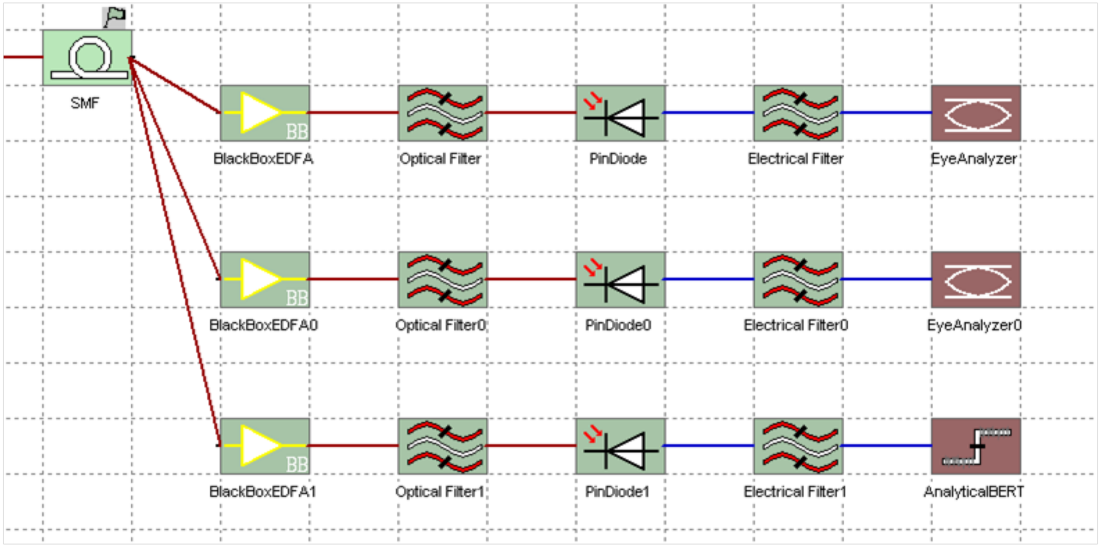


Figure 4.1.4. Type A receiver example with three frequencies. Note that the setup includes one fiber component whose inner calculation steps are depicted as separate icons. This is for clarity only, not what you would actually see on screen.

Type A receivers may include the following components:

1. An optional EDFA preamplifier is used if you check the *Set EDFA* option. Press the Preamplifier button to open a component parameter dialog and specify the EDFA settings.

2. An optical band-pass filter separates a single WDM channel from the spectrum. Press the *Optical filter* button to fine-tune its parameters. The center frequency will be automatically adjusted to the frequencies selected in the frequency table (see section *Frequency Table*).

3. A pin photodiode is used for direct detection of the optical signal. The *Diode* button will open a corresponding component parameter dialog.

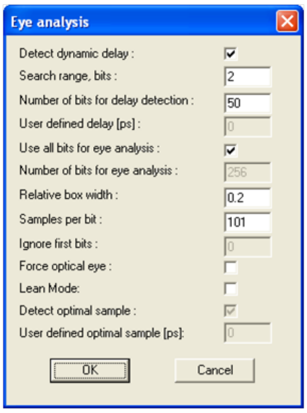
4. An electrical low-pass filter resembles the limited receiver bandwidth and provides additional pulse forming. Parameters are set via the *Electrical filter* button.

5. Analyzer components such as eye pattern analyzer or analytical BERT evaluate the received signal. It depends on the analysis methods selected in the frequency table which analyzers are used and which results are calculated (see section *Frequency Table* on page 112).

Receiver Tuning

The receivers need to be tuned to the frequencies of interest. For type A receivers, this is done by the frequency table (see section *Frequency Table* on page 112). All other receiver types involve a freely programmable MATLAB® Interface component that must be manually tuned within the user defined MATLAB® functions. Please refer to section MATLAB® Interface (7.11.4) in the Component Manual for details. The remaining receiver components can be tuned using the *PathFrequency* variable that contains the corresponding frequency of interest.

Performance Evaluation Settings



Detect dynamic delay : ☑
Search range, bits : 2
Number of bits for delay detection : 50
User defined delay [ps] : 0
Use all bits for eye analysis : ☑
Number of bits for eye analysis : 256
Relative box width : 0.2
Samples per bit : 101
Ignore first bits : 0
Force optical eye : ☐
Lean Mode: ☐
Detect optimal sample : ☑
User defined optimal sample [ps]: 0

Figure 4.1.5. Eye Analysis Parameters Dialog

Press the *Analytic Performance Evaluation - Eye analysis parameter* button to open the *Eye - Analysis* dialog. Here you have access to the same parameter set that is used in the component parameter dialog of the *Eye Analyzer* component (please refer to the *PHOTOSS Component Manual* for details):

- **Detect dynamic delay:** The eye analysis is able to perform an automatic estimation of the dynamic signal delay caused by certain transmission elements (e. g. components based on rate equation models or nonlinear algorithms) if this option is checked.

- **Search range, bits:** The dynamic delay search will not exceed the specified number of bit durations.

- **Number of bits for delay detection:** The dynamic delay search is based on maximizing the cross-correlation between the received signal and an idealized reference signal. The correlation will be calculated using the specified number of bits.

- **User defined delay:** If the user does not want the automated delay detection, he must determine this delay here. The delay is measured in ps.

- **Use all bits for eye analysis:** If checked, PHOTOSS will use all simulated bits to perform the eye analysis.

- **Number of bits for eye analysis:** If option *Use all bits for eye analysis* is not checked, the user may determine the number of bits that is used by the eye analysis.

- **Relative box width:** The eye analysis involves fitting a rectangular box into the inner eye opening so that the height of the rectangle is maximized for a given minimum width. The relative box width is expressed as a fraction of the bit duration and effectively represents the detection jitter of the receiver. The center of the resulting rectangle represents the optimum detection time (x axis) and optimum decision threshold between "zeros" and "ones" (y axis).

- **Samples per bit:** In order to ensure an integer number of samples per bit for the eye analysis, the received signal is resampled using the specified number of samples per bit.
- **Ignore first bits:** You may choose to exclude a number of bits from the eye analysis. This may be useful to suppress transient initialization artifacts before the system has reached a steady state when you are working with linear convolutions. This parameter has no effect if cyclic convolutions are chosen (see section *Simulation Parameters* on page 27).
- **Force optical eye:** If the incoming signal is an optical one, the eye analysis will not use the norm but the absolute of the signal, if this option is used.
- **Lean mode:** If a lot of bits are used (>100.000) the eye analysis may be very memory consuming. If only the optimal sample within a bit is of importance, the user may check this option. Due to the limited data stored in this mode, not all results of the eye analysis can be calculated.
- **Detect optimal sample:** If checked the eye analysis will determine the optimal sample time within the bit automatically. To do that, it will search for the maximum eye height.
- **User defined optimal sample:** If the option *Detect optimal sample* is not checked the user must determine the optimum sample time in ps. A negative value will place the optimal sample in the left part of the bit and a positive value in the right part of the bit.

User defined threshold: If 'Manual threshold determination' is checked, the user has to define the threshold.

Please refer to the descriptions of the *Eye Analyzer* and *Bit Error Rate Tester (analytical)* components in the PHOTOSS Component Manual for further details e. g. regarding the *Include noise* options.

Accessing the Results

After the path analysis simulation has been carried out, press the Results button in the *Edit Paths* dialog to open the *Global variables* dialog. As this is basically the same dialog as used for parameter variation simulations, you may refer to section *Parameter Variation* for details on its functionality.

On the *Parameter Results* page, you now find Frequency and Position as varied parameters, where *Frequency* represents the selected WDM channels, and *Position* is the position along the specified optical path. The *Component Results* column lists the results that have been calculated along the path. It depends on the chosen analysis methods which exact results are available. The methods may be specified for each selected WDM channel in the *Frequency table* dialog. Please see section *Frequency Table* on page 112 for details.

4.1.1 Summary

- The Path Analysis can be used to gain access to results *inside* a Single Mode Fiber component, e.g. the attenuation as a function of the fiber length
- Results are always considered as a function of the fiber length
- Various (built-in) receiver structures are available for an analysis of the complex signal of the fiber after each chosen length sub-step
- Each (built-in) receiver structure may be modified by the user before the simulation run
- The Path Analysis may also be used during a parameter variation

The *Analytic perform. evaluation* button opens the *Analytic performance evaluation* dialog with the following settings.
Calculate performance: Select one or more scenarios which “zero” and “one” values to use for performance evaluation.

- *All detected lines*: The average values of all detected “zeros” and “ones” at the optimal detection time are used.
- *Box lines*: The top and bottom values of the rectangular box are used.
- *All detected lines (Jitter included)*: Like *All detected lines*, but “zero” and “one” values are also averaged within the jitter area around the optimal detection time.
- *Worst case lines*: The inner eye opening bounds (highest detected “zero” and lowest detected “one”) at the optimal detection time are used.

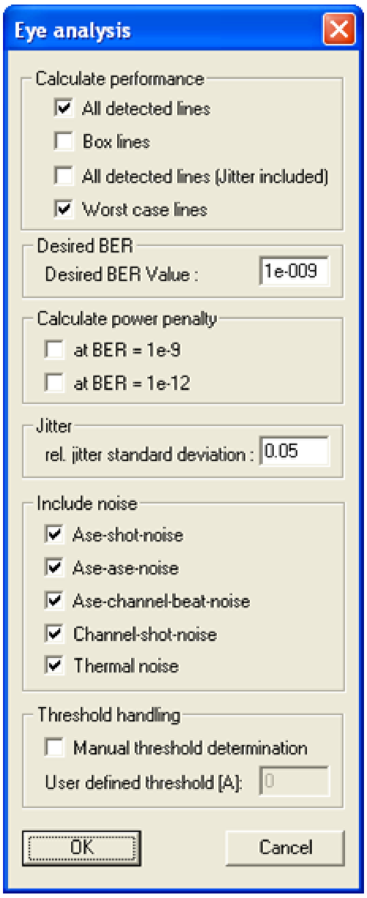
Desired BER: You may determine a BER for which the analytical BERT will calculate the required OSNR value.

Calculate power penalty: You may optionally also calculate the power penalty for a bit error rate (BER) of 10-9 or 10-12.

Jitter: You may specify a relative jitter as a fraction of the bit duration. Jitter is only regarded with the *All detected lines (Jitter included)* option.

Manual threshold determination: The user may choose between two options:

- automatic optimal threshold determination by PHOTOSS
- manual threshold determination



Calculate performance

☑ All detected lines

□ Box lines

□ All detected lines (Jitter included)

☑ Worst case lines

Desired BER

Desired BER Value : 1e-009

Calculate power penalty

□ at BER = 1e-9

□ at BER = 1e-12

Jitter

rel. jitter standard deviation : 0.05

Include noise

☑ Ase-shot-noise

☑ Ase-ase-noise

☑ Ase-channel-beat-noise

☑ Channel-shot-noise

☑ Thermal noise

Threshold handling

□ Manual threshold determination

User defined threshold [A]: 0

Figure 4.1.6. Eye Analysis Parameters Window

4.2 Frequency Table

Overview

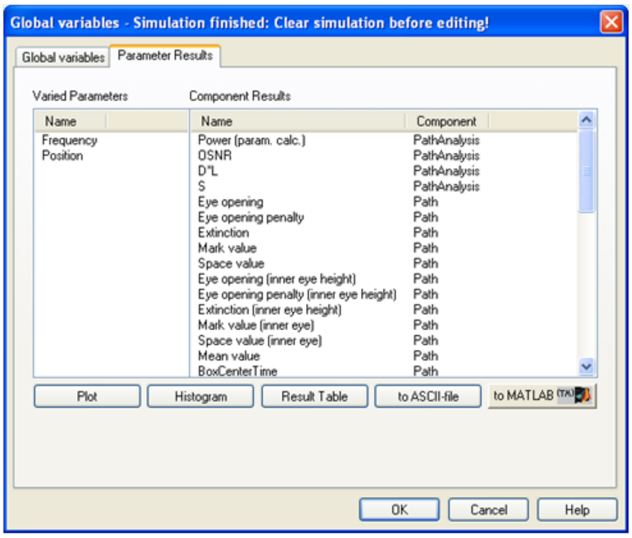
The frequency table is an auxiliary tool for selecting which frequencies to analyze during a path analysis (see section *Path Analysis*). As this particularly aims at the selection of WDM channels, the frequencies are denoted as *channels* in this respect.

Note that the selected frequencies (“*channels*”) need **not** necessarily relate to existing WDM channel carriers, so that pure sideband analyses are possible as well.

Details

Open the frequency table by pressing the *Frequency table* button from within the *Edit paths* dialog (see section *Path Analysis* on page 107).

In addition to adding or deleting channels manually, you may specify a number of channels to be equally spaced around a given center frequency with a given channel spacing.



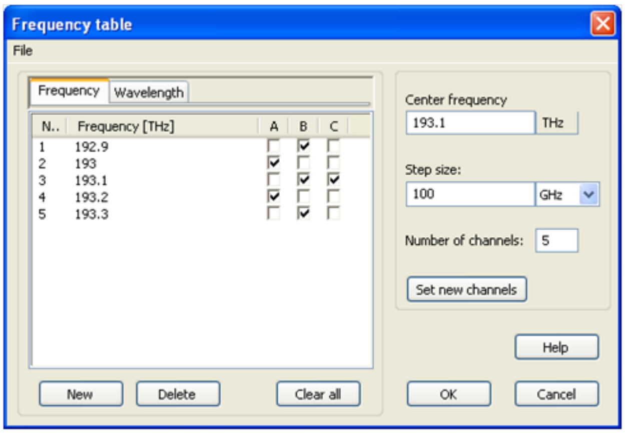
Varied Parameters

Name
Frequency
Position

Component Results

Name	Component
Power (param. calc.)	PathAnalysis
OSNR	PathAnalysis
D*L	PathAnalysis
S	PathAnalysis
Eye opening	Path
Eye opening penalty	Path
Extinction	Path
Mark value	Path
Space value	Path
Eye opening (inner eye height)	Path
Eye opening penalty (inner eye height)	Path
Extinction (inner eye height)	Path
Mark value (inner eye)	Path
Space value (inner eye)	Path
Mean value	Path
BoxCenterTime	Path

Figure 4.1.7. Eye Analysis Parameters Dialog



N..	Frequency [THz]	A	B	C
1	192.9	☐	☑	☐
2	193	☑	☐	☐
3	193.1	☐	☑	☑
4	193.2	☑	☐	☐
5	193.3	☐	☑	☐

Center frequency: 193.1 THz

Step size: 100 GHz

Number of channels: 5

Figure 4.2.1. Frequency Table

Channels may be defined both in terms of wavelength or frequency. You may switch back and forth between wavelength and frequency domain via the *Frequency* and *Wavelength* tabs.

In the File menu, you can save (*export file*) or load (*import file*) the actual frequency and analysis settings. Three analysis methods may be selected for each channel (denoted as A, B, and C), which will provide different sets of available result after the path analysis.

⚠ **Note that the analysis methods only apply to type A receivers (see section Path Analysis on page 107).**

Method A: CW Signal Analysis

Available results are *Power*, *OSNR*, *D*L*, *S* (same as with method C), plus:

- **max. Value:** The maximum electrical current behind the receiver (unit: A).
- **min. Value:** The minimum electrical current behind the receiver (unit: A).

- **avg. Value:** The average electrical current behind the receiver (unit: A).
- **Std:** The standard deviation of the electrical current behind the receiver (unit: A).
- **norm. Std.:** The normalized standard deviation of the electrical receiver current (no unit).
- **mx:** The normalized amplitude of the electrical current behind the receiver (no unit).

The above results correspond to the respective component results offered by the *Oscilloscope* component. Please refer to the *PHOTOSS Component Manual* for further details.

Method B: Full Pattern Analysis

Available results are *Power*, *OSNR*, *D*L*, *S* (same as with method C), plus:

- Eye opening, Eye opening penalty, Extinction, Mark value, Space value, Eye opening (inner eye height), Eye opening penalty (inner eye height), Extinction (inner eye height), Mark value (inner eye), Space value (inner eye), Mean value, BoxCenter Time, BoxCenter Value, mue_t, sig_t, mue_t_centroid, sig_t_centroid, mue_1, Q_1, exc_1, sig_1, sig_center, mue_center, var_box, avg_box, mue_0, sig_0

The above results correspond to the respective component results offered by the *Eye Analyzer* component. Please refer to the *PHOTOSS Component Manual* for further details.

- All lines (Box lines, Worst case lines, Jitter included), Ber
- All lines (Box lines, Worst case lines, Jitter included), Q
- All lines (Box lines, Worst case lines, Jitter included), power4berm9
- All lines (Box lines, Worst case lines, Jitter included), power4berm12

The above results correspond to the respective component results offered by the *Bit Error Rate Tester (analytical)* component. Please refer to the *PHOTOSS Component Manual* for further details.

Note that there are four possible cases for each result (*All lines*, *Box lines*, *Worst case lines*, *Jitter included*). These correspond to the four *Calculate performance* options in the *Analytic performance evaluation* dialog (see section *Path Analysis* on page 107).

Method C: Power Analysis

This method is based on the parametric signal analysis. It involves the following results:

- **Power:** The optical power before the reference receiver. Only first order properties are taken into account (such as fiber loss or amplifier gain). The power is given in dBm.
- **OSNR:** The optical signal-to-noise ratio (SNR) before the reference receiver. The noise power is approximated by integrating the broadband noise vector over the signal bandwidth that is defined in the simulation parameters. The OSNR is given in dB.
- **D*L:** The accumulated dispersion (linear material dispersion $D \cdot$ fiber length L) in ps/nm.
- **S:** accumulated dispersion slope (dispersion slope · fiber length) in ps/nm2.
- **power:** The electrical power behind the receiver given in the unit W @ 50 Ohm, where an electrical resistance of 50 Ω is assumed.

4.3 Command Line Batch Mode

Introduction

There are two command line batch modes; one for multiple simulations and for a single simulation.

PHOTOSS may require the usage of administrator privileges when run in batch mode - especially when run with a job distribution tool or with MATLAB®. It may be necessary to open the "CMD" itself with administrator privileges to "migrate" these privileges to the application.

Running multiple simulations from the command line

The following steps are required to run multiple simulations:

1. **Create a list file**
   Write the names of all desired *.pho files in a list file. The list file is a plain ACSII text file with any filename (e. g. *mySims.txt*) and the following syntax:

   ```
   sim1.pho
   sim2.pho
   special\simSpA.pho
   c:\simI.pho
   ```

   You may use relative or absolute paths. The files *sim1.pho* and *sim2.pho* are either located in the same directory as the list file or in the *My Documents folder* within the user profile (typically *C:\Documents and Settings\<Username> My Documents*). *simSpA.pho* is in the special subfolder of one of the above. The last line specifies an absolute location on drive C.

2. **Start PHOTOSS from the command line**
   After creating the list file, open a command line window and navigate to the directory that contains *PHOTOSS.exe* (typically *C:\Program Files\Lenge\PHOTOSS x.xx*). Assuming that the list file *mySims.txt* is located in *D:\myDir*, type

   photoss -b D:\myDir\mySims.txt

   (no absolute path is required if the list file is located in the same directory as *PHOTOSS.exe* or in the *My Documents* directory). PHOTOSS will open and run each file on the list in the specified order. When all simulations are finished, PHOTOSS will terminate automatically. The text that would normally be displayed in the *Simulation Output* window is saved to the folder that contains the list file. If *mySims.txt* is the list file, the output file is named *mySims_batchlog.txt*.

Running a single simulation from the command line

Open a command line window and navigate to the directory that contains *PHOTOSS.exe* (typically *C:\Program Files\Lenge\PHOTOSS x.xx*). Assuming that the simulation file is *D:\myDir\sim1.pho*, type *photoss -r d:\myDir\sim1.pho*, the simulation output will be saved to:
D:\myDir\sim1.pho_batchlog.txt.

Minimizing PHOTOSS, when started from the command line

If PHOTOSS is started several times from a batch-file, the user may be disturbed by the PHOTOSS window, which is opening every time a new PHOTOSS instance is opening. This is why the parameter *–minimized* may be used to start PHOTOSS in a minimized form.

Command Line Parameters

There are a number of additional parameters, which can be specified, when calling *PHOTOSS.exe* from the command line. The GPU acceleration option may be altered by the parameter *–GPU=*. 'none' disables the GPU calculation of the fiber. 'default' will activate the GPU acceleration. If you have installed multiple GPUs into your PC, the parameter *–GPU=* may be used to specify the desired GPU. You can either give a number ('1', '2', ...) of the desired GPU similar to the enumaration in the Tools⇒Options dialog. Furthermore, 'auto' may be used to automatically select a GPU, which is currently not used by another *PHOTOSS.exe* instance. If the automatic selection of the GPU is desired, a specific GPU must not be selected in the Tools⇒Options dialog.

With the parameter *-p* a global variable may be changed. If e. g. a global variable with the name *power* exists in the simulation file you want to execute, it may be set to the value 12 by the parameter *-p power=12*.

4.4 Avoiding start/close overhead of PHOTOSS

Introduction

If PHOTOSS is called by a script to execute a single simulation (e.g. *photoss -r mySim.pho*), PHOTOSS starts, opens the PHO file, executes the simulation and is closed again. Starting and closing of PHOTOSS takes a couple of seconds. If multiple short simulations are executed, the starting and closing process becomes a significant factor.

Passing Simulation files between two instances of PHOTOSS

The solution to this problem is a new command line *parameter -c* (short for '*client*'). If PHOTOSS is started with this option, it starts and searches for another instance of PHOTOSS (the server). If it finds an active instance of PHOTOSS, it sends its simulations to that instance and shuts down again. The start and shutdown of the client takes significantly less time than starting a full PHOTOSS application.

To use this feature, the following steps are required:

1. Start an instance of PHOTOSS (e.g. by double-clicking *PHOTOSS.exe*, or by calling PHOTOSS from the command line). This instance of PHOTOSS will act as a server.

2. Now one may start a client from the command line. It does not matter, whether one starts just one simulation (using option -r) or a list of simulations (using option b):

   `photoss -c -r mySim.pho` or `photoss -c -b myList.txt`

3. Now the server instance will execute the simulations. If the server is already executing another simulation, the simulations sent by the client are queued and executed afterwards.

4.5 Overriding Simulation Parameter values

Introduction

When PHOTOSS is started with a simulation file given as command line parameter it is possible to set the value of global variables used in this simulation. This is useful for controlling a parameter variation from the outside of PHOTOSS, for example by script.

Setting parameter values from the command line

As described above PHOTOSS can be used with the command line parameters “-r” or “-b” to instantly load and run given simulation files. If one of these parameters is given an additional parameter “-p” is also accepted. “-p” must be followed by a string like “<variable>=<value>” where “<variable>” is the name of a global variable used in the given simulations and “<value>” is a floating point number. If multiple parameters are set via the command line, the option “-p” is required for every parameter. In any case, the global variable has to be set to type “Variation” in the “Edit global Variable”-dialog before.

Example:

```
photoss -p power=12.5 -r sim.pho
```

If the given simulation defines a global variable named “*power*” its value will be set to 12.5 and the simulation will be run. If there is no such global variable defined parameter “-p” has no effect.
Note that the parameter’s variation values will be overwritten and the simulation file will be saved.

4.6 Loading Component DLLs

Additional PHOTOSS component models may be loaded from external dynamic link library (DLL) files. During startup, PHOTOSS will automatically search its *component_dlls* subdirectory for suitable DLL files with additional component models and try to load them.

If the *component_dlls* subdirectory is empty or does not exist (this is typically the case when PHOTOSS is being run under Condor®), PHOTOSS tries to load the library *dev_kit.dll* from the directory where `PHOTOSS.exe` is located.

4.7 Creating Components

4.7.1 Underlying Concept

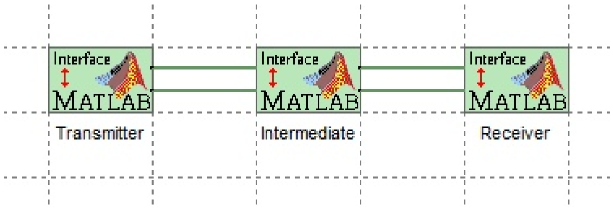
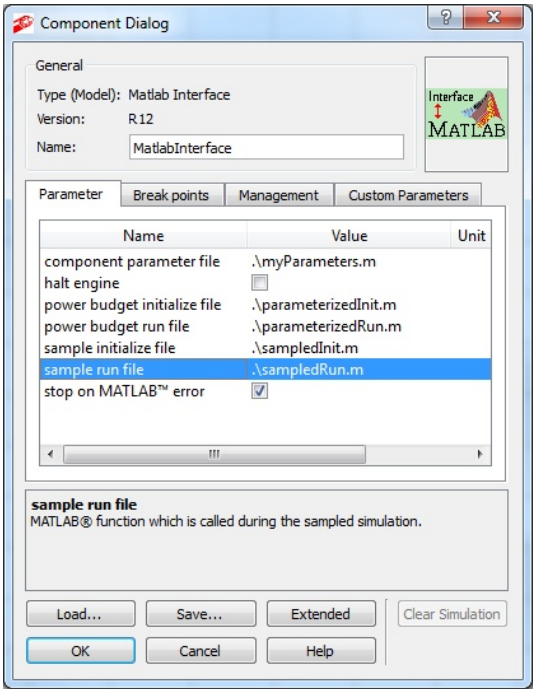


Figure 4.7.1. MATLAB® components can be implemented as transmitters (left), intermediate components (middle) and receivers (right).

For various reasons, it might be interesting for you to create your own components and use the with the PHOTOSS application. Generally, this can be easily accomplished when using the MATLAB® interface. More details of the interface can be found in section 7.11.4 on page 347. In the following subsections we will illustrate how you can design and implement your own transmitter, intermediate and receiver MATLAB® components. Please note that it is also possible to write your own C or C++ components and use them with the command line interface (see 7.11.2), but this topic will not be addressed further as it is much more complex.

4.7.2 Component Runs



General

Type (Model): Matlab Interface

Version: R12

Name: MatlabInterface

Name	Value	Unit
component parameter file	.\myParameters.m	
halt engine	□	
power budget initialize file	.\parameterizedInit.m	
power budget run file	.\parameterizedRun.m	
sample initialize file	.\sampledInit.m	
sample run file	.\sampledRun.m	
stop on MATLAB™ error	☑	

sample run file
MATLAB® function which is called during the sampled simulation.

Figure 4.7.2. A specific MATLAB® file can be supplied for each type of simulation run. Supplying a MATLAB® file is optional.

These remarks apply to all types of MATLAB® components and should always be kept in mind. Just like the simulation runs described in 3.10, the MATLAB® component offers to supply a MATLAB®

script for each of individual simulation run (or “component run”). The individual MATLAB® files can be specified for the corresponding component parameters - see figure 4.7.2. Supplying a MATLAB® file is optional - if no file has been specified for one of the simulation runs, the MATLAB® interface will only pass the incoming signal(s) to its out-port(s) and will not change the content of the signal. If, however, you intend to implement a more complex component with i.e. multiplexing-features and an unequal number of in- and out-ports you should provide a MATLAB® file for each simulation run to ensure that the signal mapping from in- to out-port is always carried out as you intended.

The following component runs can be addressed individually:

- *component parameter file:* this file should contain code that should be carried out only *once* per block and prior to the calculation of the parameterized signal.
- *power budget initialize file:* this file will be evaluated once the component is processed during the initialization of the parameterized run (once per simulation run)
- *power budget run file:* this file will be evaluated once the component is processed during the parameterized run (once per simulation run)
- *sample initialize file:* this file will be evaluated once the component is processed during the initialization of the sampled run (once per simulation run)
- *sample run file:* this file will be processed *each time* a sampled run is executed (once per block per simulation run).

Note, that “simulation run” refers to a single set of parameters in a parameter variation. A parameter variation of one parameter with ten distinct values will result in *ten* simulation runs (while each simulation run may also have multiple blocks!).

A key requirement for creating your own MATLAB® components is to understand how the PHOTOSS signal and multi signal model is implemented. Please refer to chapters 3.7, 3.9 and 3.8 before trying to implement your components.

4.7.3 Creating an Intermediate Component

This subsection will deal with the creation of *intermediate* components - i.e. components which are not used as transmitters or receivers but between them. The requirements for these type of component are as follows - they should be able to:

- have any number of in-ports
- have at least one out-port
- pass any number of incoming signals to any out-port(s)
- modify any incoming signal
- work in both the electrical and/or optical domain

As we will illustrate, a MATLAB® component can accomplish all these tasks easily because the “MultiSignal” structure is already present at the in-port(s) of the MATLAB® component and in most of the cases it only has to be modified. Please open the PHOTOSS Example *MATLAB Intermediate Component* in your PHOTOSS example directory to see the implementation and usage of the MATLAB® interface. All MATLAB® files can be found in the same directory as well. After the execution of the example simulation you should be able to view a constant signal power and amplitude output in the oscilloscopes after the MATLAB® interface. You will also notice that the result *Power (sampled calc.)* as well as *avg. Value*, *max. Value* and *min. Value* of the oscilloscope are set to “2.0”. This is due to the fact that the complex signal value of *each* of the two polarization is set to “1.0” and the oscilloscope calculates the *norm* of the complex signal.

4.7.4 Creating a Receiver Component

Creating a receiver Component is even more easy than creating an intermediate component (see above), since you normally do not need to pass a “MultiSignal” to an out-port of the component - in most of the cases your receiver terminates an optical or electrical transmission line. Very often, receiver components have the task to calculate observables like the BER or the required OSNR at a fixed BER from the data of an incoming sampled signal. The principle of creating such a component is almost completely identical with the one described above. A receiver component should:

- have any number of in-ports
- normally have no out-ports
- normally not pass any incoming signal to any out-port(s)
- modify an incoming signal only for internal usage
- work in both the electrical and/or optical domain

The PHOTOSS example entitled *MATLAB Receiver Component* will demonstrate how a very simple oscilloscope receiver can be realized by a MATLAB® component. Running the designated simulation file should automatically produce plots of the complex optical and electrical signal. The simulation has been implemented in separated channels mode to illustrate that plots for each distinct simulation band/channel can be shown. You may also notice that the optical filter at the end of the system lets only the first sampled signal channel at 193.1 THz pass.

4.7.5 Creating a Transmitter Component

A signal source or transmitter component can be used to implement your own custom modulation format or modulation source in PHOTOSS. From a practical point of view, writing such a component in MATLAB® is slightly more difficult: The reason is, that you have to create all parameterized and sampled signal informations *from scratch*, i.e. you do not use any template or blue-print incoming signal which you only have to modify as would be the case when using intermediate components.

This can be accomplished by using some predefined MATLAB® functions. However, there are some certain limitations you should always keep in mind when creating your own signal sources (please refer to section 3.7 for more information on the PHOTOSS signal structure!). A custom transmitter can *not* be used in combination with the following PHOTOSS receiver components:

- Analytical BERT
- Eye Analyzer
- Numerical BERT
- Single Pulse Analyzer

The reason for this is that the components listed above make certain assumptions about the analytical noise model and - more importantly - the modulation of the signal. They have not been designed to handle user-defined modulation formats. Normally, this should not pose a serious problem as a new type of modulation format will also most probably require a new type of receiver which can also be implemented as a MATLAB® component.
All other PHOTOSS components should be able to process any modulated signal since they were designed to operate on the complex signal itself and they do not rely on secondary information about the modulation or analytical performance evaluation substructure of the signal.

In any case, *all* MultiSignal structure subcomponents *must* be initialized properly in order for PHOTOSS to “understand” your newly generated signal. The easiest way to observe restrictions at once is to view the PHOTOSS example *MATLAB Source Component* and start from there. This example will show you how a MATLAB® transmitter can be implemented and it will also demonstrate how to properly initialize all MultiSignal values. The MATLAB® component will generate a very simple NRZ-ASK signal which can also be viewed in a PHOTOSS oscilloscope.

Please be aware that you can simply copy the MATLAB® script file “initParameterizedMultiSignal.m” and use it to create your own paramezerized transmitter signals. If any changes are made to the signal model in future PHOTOSS versions you only have to replace this function in your setups and you do not have to touch your own code.

⚠ **You should *not* try to modify the modulation or ana_per_info substructures of the parameterized signal unless you are absolutely certain what you want to do and how changes can be implemented safely. Improper settings may cause the application to crash, so remember to always save your work before experimenting.**

4.7.6 Behavior in Multiple Block Simulations

In some cases, you may want to accumulate data over multiple blocks when using a multiple block simulation (see 3.11). You will note that only small code adaptations are necessary to do this: During the sampled signal run - and thus during each block - the MATLAB® workspace will always hold the variable *current_block_no* which will tell you the number of the currently processed block. The *total* number of blocks can always be accessed by calling up the variable *simulation_parameter.Number_Blocks*.

4.7.7 Summary

- The easiest way of creating user-defined components is using the MATLAB® interface
- Three distinct types MATLAB® components can be implemented: Transmitters, Receivers and intermediate components.
- Transmitter components are used to generate custom complex, sampled signals. A generic initialization function is included in the PHOTOSS example directory.
- Intermediate components modify all incoming MultiSignals.
- Receiver components are used for signal detection and computation of observables / results.
- A user-generated signal can *not* be detected with PHOTOSS receiver components (except for the oscilloscope). You may have to implement your own receivers.
- A user-generated signal *can* be processed in almost every PHOTOSS intermediate component, e.g. the single mode fiber, etc. If a custom signal is not supported by the designated component, an error will be thrown to indicate this.
- MATLAB® components can supply functions for each simulation run (parameterized and sampled)
- Detailed examples for all user-defined components are included in the PHOTOSS example directory of your installation

5 Example Setup

5.1 40 Gbit/s Transmission Walkthrough

For a step-by-step example walkthrough, we consider a small 40 Gbit/s transmission system as depicted below. It consists of an optical pulse generator, a single mode fiber, a PIN receiver diode, and an analytical filter to resemble the limited bandwidth of the receiver. The received signal is displayed by an eye pattern analyzer.

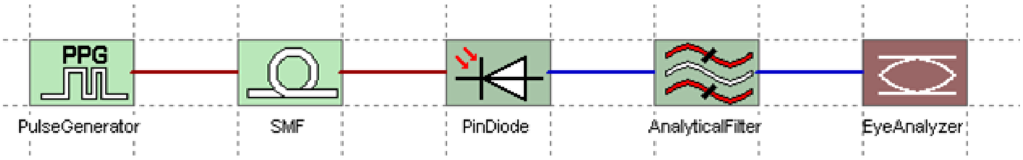
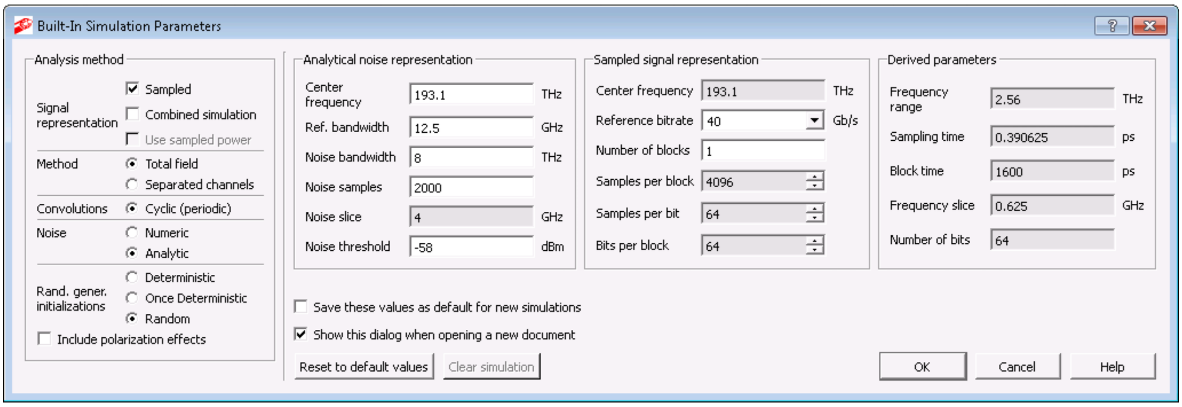


Figure 5.1.1. Example Setup

1. Create a new project by pressing the toolbar button (or select *New/NewSimulation* in the File menu).

2. Define the simulation parameters. If the dialog does not appear automatically, open it with the toolbar button. Choose the settings as shown. We will use the total field method with cyclic convolutions and regard a frequency range of 2.56 THz centered around 193.1 THz. A single signal block of 4096 samples is chosen.



Built-In Simulation Parameters

Analysis method
Signal representation: ☑ Sampled, □ Combined simulation, □ Use sampled power
Method: ● Total field, ○ Separated channels
Convolutions: ● Cyclic (periodic)
Noise: ○ Numeric, ● Analytic
Rand. gener. initializations: ○ Deterministic, ○ Once Deterministic, ● Random
□ Include polarization effects

Analytical noise representation
Center frequency: 193.1 THz
Ref. bandwidth: 12.5 GHz
Noise bandwidth: 8 THz
Noise samples: 2000
Noise slice: 4 GHz
Noise threshold: -58 dBm

Sampled signal representation
Center frequency: 193.1 THz
Reference bitrate: 40 Gb/s
Number of blocks: 1
Samples per block: 4096
Samples per bit: 64
Bits per block: 64

Derived parameters
Frequency range: 2.56 THz
Sampling time: 0.390625 ps
Block time: 1600 ps
Frequency slice: 0.625 GHz
Number of bits: 64

□ Save these values as default for new simulations
☑ Show this dialog when opening a new document

Figure 5.1.2. Simulation Parameters

3. Now create the transmission network. Drag the required components from the component tree and place them onto the grid. At this point your network should look like figure 5.1.3.

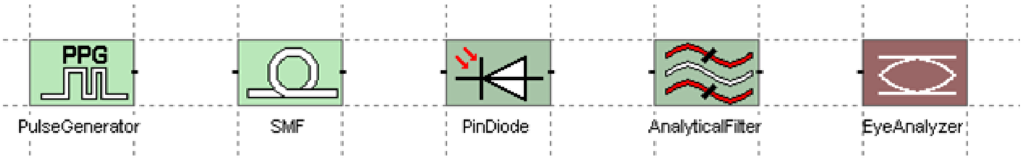


Figure 5.1.3. Modified Example Setup

4. Set the component parameters by double-clicking each component. Use settings as follows:
   - **Pulse Generator:** leave default settings

- **Single Mode Fiber:** *length = 5 km* (all other settings = default)
- **PIN Photodiode:** leave default settings
- **Analytical Filter:** *Mode = electrical* (all other settings = default)
- **Eye Analyzer:** leave default settings

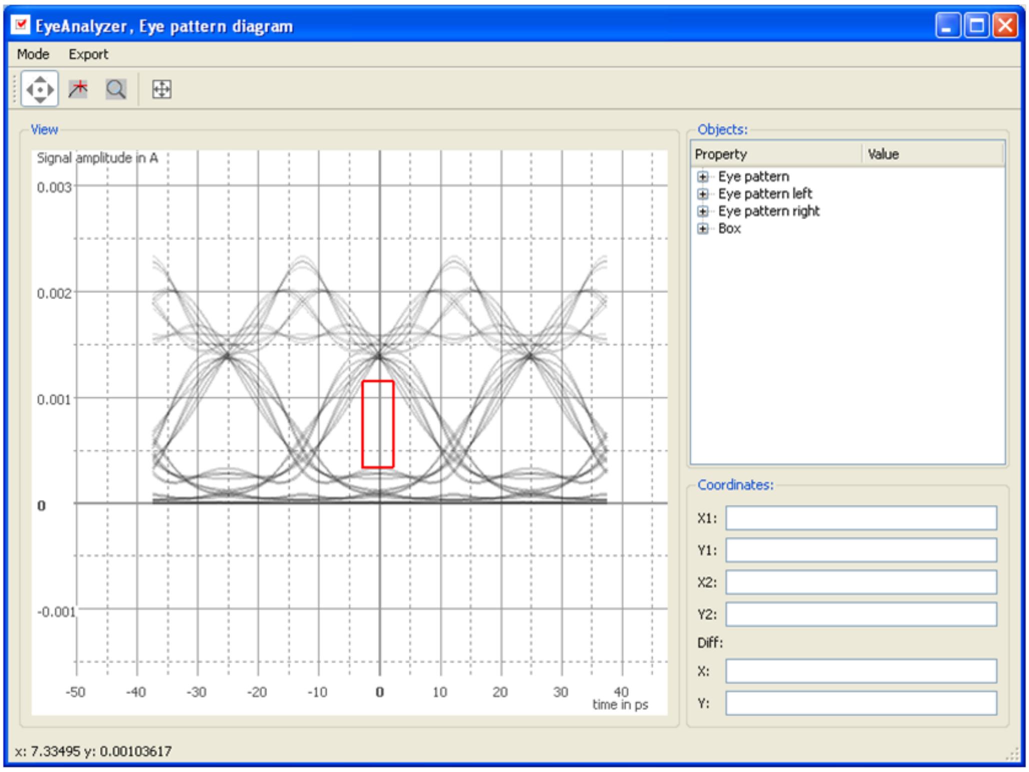
5. Connect the components to define the signal flow. Use one of the approaches described in section *Connecting Components*. Your network should now look like the initial picture.

6. Save the simulation by pressing the 💾 toolbar button or the *File/Save* menu entry. Name the simulation file as you like (with extension ".pho") and specify a path on your hard disk.

7. Run the simulation via the ▷ toolbar button or the *Simulation/Start simulation* menu entry. Depending on your hardware, our example should complete within a few seconds.

8. After the simulation is completed, right-click the eye analyzer component and select *Eye pattern diagram* from the pop-up menu to display the eye diagram at the receiver.

9. Clear the simulation results with the ✕ toolbar button or the *Simulation/Clear* simulation menu entry. You may now review or change parameters and re-calculate the network. Finally close the simulation file via the *File/Close* menu entry.



x: 7.33495 y: 0.00103617

Figure 5.1.4. Eye Analyzer Results for Example Setup

6 Grid Computing

6.1 Introduction to Grid Computing

This chapter comprises a step-by-step walkthrough that illustrates how to distribute PHOTOSS simulations onto multiple computers using Condor® automatically. Condor® is a grid computing tool that enables the simultaneous utilization of multiple systems/machines for complex calculations.

We will not explain the functionality of Condor® in all details (according documentation can be found at http://www.cs.wisc.edu/condor). Instead, we focus on how to use Condor® with PHOTOSS simulations, and present approaches that effectively reduce the effort required for distributing simulations onto multiple machines and collecting the according results.

6.1.1 Condor®

PHOTOSS comes with built-in support for Condor® based grid computing.

Condor® is a product of the Condor® Research Project at the University of Wisconsin-Madison (UW-Madison). It is not part of PHOTOSS, but is included in certain PHOTOSS distributions for your convenience with friendly permission of the University of Wisconsin. You may download the latest version and the source code from http://www.cs.wisc.edu/condor).

Figure 6.1.1. Logo of the Condor® Grid Tool

Trademark attribution:

Condor® is a registered trademark of the Board of Regents of the University of Wisconsin System (UW). While we are using the Condor® trademark with friendly permission of the owner, our own products, publications, and websites are not endorsed or approved by the UW or the Condor® Research Project.

Condor® is subject to the licensing conditions presented at

http://www.cs.wisc.edu/condor/license.html)

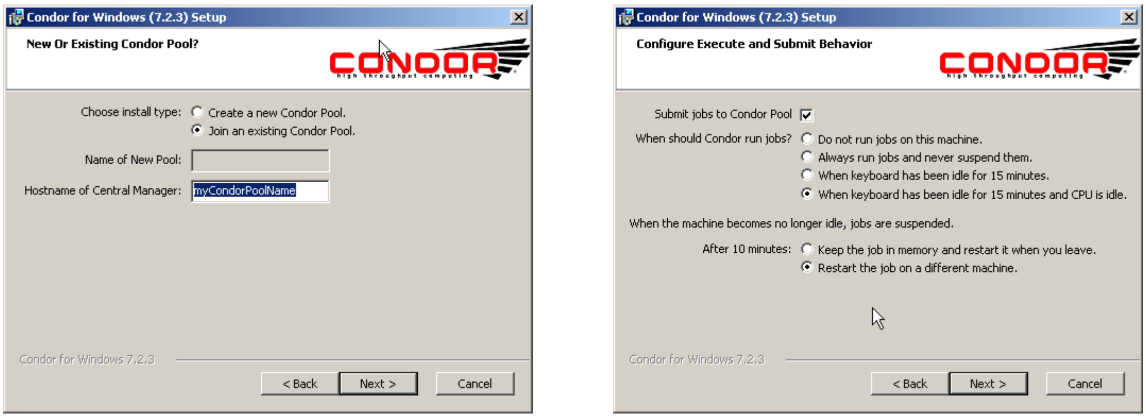
A copy of these (as valid at the time of writing) is enclosed with any PHOTOSS package that includes Condor.

6.1.2 MATLAB®

For the integration of user-defined algorithms, PHOTOSS includes a programming interface that connects to MATLAB®, a commercial software package for numerical calculations.

MATLAB® is a product of The MathWorks™ (http://www.mathworks.com).

MATLAB® is not part of PHOTOSS and is not included in PHOTOSS distributions. In order to use the programming interface, you must have a supported version of MATLAB® installed on your computer. Please refer to the MATLAB® documentation for details on the installation.



Condor for Windows (7.2.3) Setup

New Or Existing Condor Pool?

Choose install type: ○ Create a new Condor Pool. ● Join an existing Condor Pool.

Name of New Pool:

Hostname of Central Manager: myCondorPoolName

Condor for Windows (7.2.3) Setup

Configure Execute and Submit Behavior

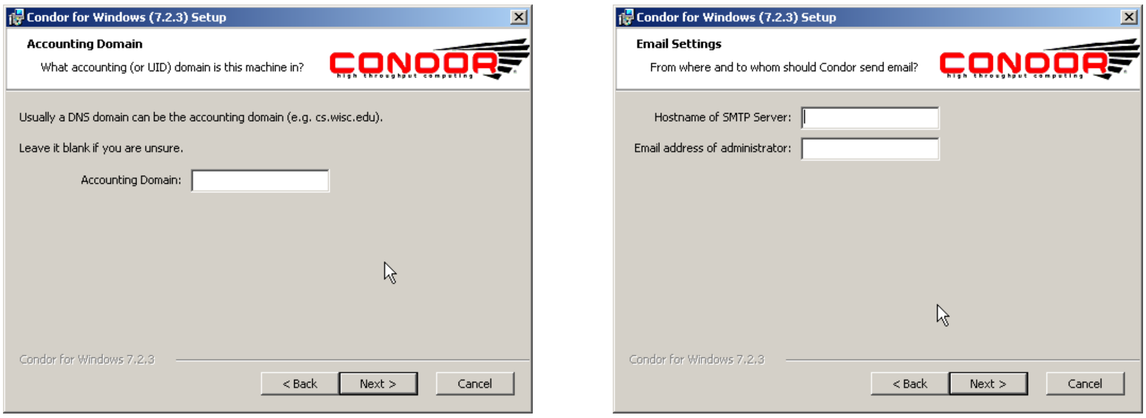
Submit jobs to Condor Pool ☑

When should Condor run jobs?
○ Do not run jobs on this machine.
○ Always run jobs and never suspend them.
○ When keyboard has been idle for 15 minutes.
● When keyboard has been idle for 15 minutes and CPU is idle.

When the machine becomes no longer idle, jobs are suspended.

After 10 minutes:
○ Keep the job in memory and restart it when you leave.
● Restart the job on a different machine.

Figure 6.2.1. Condor® Join Pool (left) and Run Jobs Dialog (right)



Condor for Windows (7.2.3) Setup

Accounting Domain

What accounting (or UID) domain is this machine in?

Usually a DNS domain can be the accounting domain (e.g. cs.wisc.edu).

Leave it blank if you are unsure.

Accounting Domain:

Condor for Windows (7.2.3) Setup

Email Settings

From where and to whom should Condor send email?

Hostname of SMTP Server:

Email address of administrator:

Figure 6.2.2. Condor® Accounting Domain (left) and Email Settings Dialog (right)

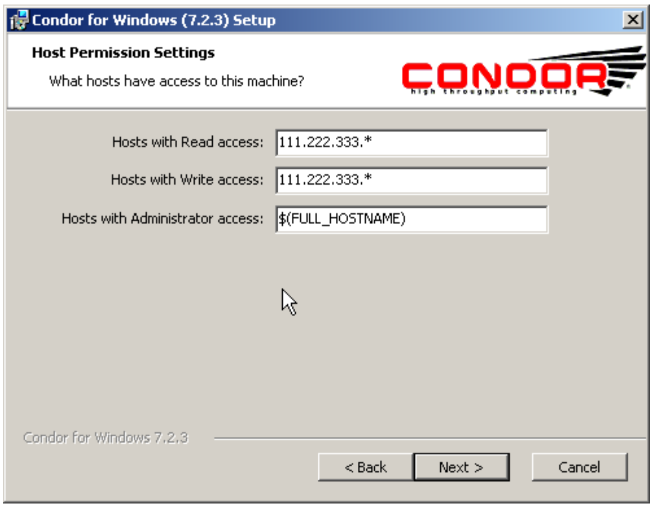
6.2 Installing Condor®

Condor® must be installed on every machine in your pool. Here we focus on the GUI installer for Microsoft® Windows® systems. Before installing, define one computer to become the manager (or master). The manager is the first computer to install Condor® on, and is ideally never turned off or disconnected from the network, because Condor® cannot distribute or collect any jobs when the manager is not available.

When installing Condor® on the manager, select *Create a new Condor® Pool* and specify a name for it. On all other computers, select *Join an existing Condor® Pool* and enter the hostname of the manager.

On the *Execute and Submit* Behavior page (see fig. 6.2.1), check *Submit jobs to Condor® Pool* to be able to *submit* jobs from this computer. We recommend this for all computers in the pool.
Then specify under which conditions jobs may be *run* on this computer. We recommend selecting *Do not run jobs on this machine* on the manager, as its absolute priority is being ready to distribute and manage jobs whenever required (but not running jobs itself).
Finally, you may choose what measures to take if this computer becomes no longer idle while it is running Condor® jobs.

On the *Accounting Domain* and *Email Settings* pages (see figure6.2.2), you may leave all settings blank. If you want to use Condor® with Java programs, you may set the location of your Java installation on the next page. However, Java is not necessary for using Condor® with PHOTOSS.



Condor for Windows (7.2.3) Setup

Host Permission Settings

What hosts have access to this machine?

Hosts with Read access: 111.222.333.*

Hosts with Write access: 111.222.333.*

Hosts with Administrator access: $(FULL_HOSTNAME)

Figure 6.2.3. Condor®Host Permission Settings

On the *Host Permission Settings* page (see figure 6.2.3), enter which computers have read and/or write access to Condor® data on this computer. We recommend granting both read and write access to all other computers in the pool in order to make Condor® operate properly.
You may use the asterisk as a wildcard to specify an entire IP range. In the above example, all computers whose IP starts with "111.222.333." will have both read and write access. (Note: Never change the *Hosts with Administrator access* field without a good reason!)

When asked if you want to enable the VM universe, select *No* and proceed with the installation. Finally restart the computer after the installation of Condor® has finished.

6.2.1 The Condor® Configuration File

After the installation, all options that have been selected during the installation can be found (and also changed, if necessary) in the `condor_config` file in the Condor® installation directory (C:Condor®by default). Please be careful with modifications to the configuration file, as a wrong setting may easily result in rejection or idleness of submitted Condor® jobs.

- The `CONDOR_HOST` variable determines which computer is the manager; it should be set to: `$(FULL_HOSTNAME)` on the manager itself. On all other computers, it should be set to the hostname of the manager.
- The `HOSTALLOW_READ` and `HOSTALLOW_WRITE` variables specify which computers have read and/or write access to Condor® data on this computer. They should be set so that all other machines in the pool have both read and write access (e. g. using the asterisk as a wildcard).
- The `START` variable defines the conditions under which this computer runs a job.

6.3 Using Condor® with PHOTOSS

6.3.1 First Steps

Condor® itself does not include a graphical user interface. Instead, it is controlled from the command line by calling batch scripts or interactively entering commands. On Microsoft® Windows® XP, you can open the command line interpreter `cmd.exe` from the start menu via *Programs* ⇒ *Accessories* ⇒ *Command Prompt*.

We will first focus on the most essential Condor® commands that you can enter on the command line. Open the command line interpreter as described before. For simplicity, we assume that Condor® has been installed to `C:\Condor`.

Before you can submit jobs to the Condor® pool, you need to store your credentials. Enter:

```
c:\condor\bin\condor_store_cred.exe add
```

You will be asked to enter the username/password combination that Condor® will use to log into remote computers. You will typically enter your "normal" Windows® user account data which is valid on all computers in the pool. The command

```
c:\condor\bin\condor_status.exe
```

will list all computers that are currently available in the Condor® pool. As you have not yet submitted any jobs, the listed computers should either be marked as *unclaimed* or occupied by an *owner*. In this respect, "claimed" means that the computer is currently in use by Condor® jobs, while "owner" indicates that a different user is currently using the computer.

Now use the `cd` command to switch to the Condor® Examples subfolder of the PHOTOSS examples directory. For any installed PHOTOSS version `x.yz`, the examples are typically located in

```
%AllUsersProfile%\Application Data\Lenge\PHOTOSS x.xx\Examples
```

First run the CheckCondorFunctionality.txt example with:

```
c:\condor\bin\condor_submit.exe "CheckCondorFunctionality.txt"
```

You have just submitted your first Condor® job. The command

```
c:\condor\bin\condor_q
```

shows the status of all jobs that you have currently submitted. You should now see one job which is *running* (or *idle*, if all computers are currently *claimed* or in *owner* state). If no active or idle job is listed, it has probably already been completed.
After the `CheckCondorFunctionality.txt` example has finished, you should find two new files in the Condor® Examples directory, `test1.dat` and `test2.dat`. These indicate that Condor® has been properly installed and configured.

Now take a look at the other examples. The `Condor Examples` folder holds various jobs that illustrate the combined use of Condor® with PHOTOSS and (optionally) MATLAB®. Refer to `Readme.txt` and the individual `.txt` files for a detailed description of each example.

The following commands might also be useful:

- `c:\condor\bin\condor_rm clientName` removes all jobs that have been submitted by the user `clientName`
- `net stop condor`, resp. `net start condor` disables (resp. restarts) Condor® on this computer

6.3.2 Condor® Job Files

Every Condor® job is represented by an according job file. Job files are plain text files with an arbitrary extension (we use `.txt` for simplicity) that can be submitted to Condor®. Have a look on the `CheckCondorFunctionality.txt` example. After some comments lines (starting with #), this contains:

```
log=log.txt
Executable = checkCondorFunctionality.bat
Universe = vanilla
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
Requirements = (Arch == "INTEL" && (OpSys == "WINNT51"||OpSys== \
"WINNT52"||OpSys == "WINNT60"||OpSys == "WINNT61"))
##############################################################
transfer_input_files = checkCondorFunctionality.bat
Queue
```

- `log` makes Condor® create a log file that records the start time, end time, and the return code of the job. A return code of 0 typically means that the job has been completed without errors.
- `Executable` specifies the program to execute when the job has been transferred to a designated computer.
- `Universe` is a specific Condor® environment setting. Do not set it to any value other than vanilla unless you have a good reason.
- `when_to_transfer_output` defines when the result files (i. e. files that are created or modified by the job) should be transferred back to the computer that submitted the job.
- `transfer_input_files` specifies all files that are required for executing the job, separated by commas. (In the above example above, only the executable itself is required.)

In order to submit a PHOTOSS simulation as a Condor® job, you need to

- **have PHOTOSS installed on every computer in the Condor® pool**, including the Visual C++ Redistributable and the WibuKey drivers.
- **include all files that are required by PHOTOSS in the job file**, including all executables and `.dll` files. As Condor® cannot handle directories for a transfer, you have to include every single file.

The following is an excerpt from the `MonteCarloSimulation.txt` example and includes all the files necessary to compute the simulation `monteCarlo.pho`:

```
photoss_dir = C:\Programme\Lenge\PHOTOSS 5.10
log=log.txt
Executable = $(photoss_dir)\PHOTOSS_grid.exe
Universe = vanilla
should_transfer_files = YES
```

```
when_to_transfer_output = ON_EXIT
Requirements = (Arch == "INTEL" && (OpSys == "WINNT51"||OpSys == "WINNT52"
    ||OpSys == "WINNT60"||OpSys == "WINNT61"))
###############################################################################

transfer_input_files = \
$(photoss_dir)\DefaultModelTree.txt, \
$(photoss_dir)\DefaultUserTree.txt, \
$(photoss_dir)\component_dlls\dev_kit.dll, \
$(photoss_dir)\matlabR12_dll.dll, \
$(photoss_dir)\matlabR13_dll.dll, \
$(photoss_dir)\matlabR14_dll.dll, \
$(photoss_dir)\matlabR2006a_dll.dll, \
$(photoss_dir)\matlabR2007a_dll.dll, \
$(photoss_dir)\PHOTOSS_grid.exe, \
$(photoss_dir)\photoss_dll.dll, \
$(photoss_dir)\QtCore4.dll, \
$(photoss_dir)\QtGui4.dll, \
$(photoss_dir)\QtOpenGL4.dll, \
$(photoss_dir)\QtScriptClassic1.dll, \
$(photoss_dir)\QtXml4.dll, \
$(photoss_dir)\cudart.dll, \
$(photoss_dir)\cufft.dll, \
$(photoss_dir)\cudart32_30_14.dll, \
$(photoss_dir)\cufft32_30_14.dll, \
$(photoss_dir)\ToolkitPro1113vc80.dll, \
..\ExampleSimulations\monteCarlo.pho

Arguments = -r monteCarlo.pho
Initialdir = monteCarloRun1
Queue
Initialdir = monteCarloRun2
Queue
Initialdir = monteCarloRun3
Queue
```

Condor® copies the executable and all additionally specified files from the computer that submitted the job to a temporary directory on the remote computer that will execute the job. The temporary directory must not have any subdirectories. All files, including the .pho simulation, will be placed in the very same directory.

⚠ **Depending on which computer Condor® selects for job execution, it may occur that the job is carried out on the same computer that submitted it.**

`photoss_dir` must contain the full path to your PHOTOSS installation.

`Initialdir` specifies the directory where to store the result files (`monteCarloRun1` for the first run in the above example). This directory must exist (but may be empty) when submitting the job, or else Condor® will report an error.

6.3.3 Using Condor® and MATLAB®

In order to submit PHOTOSS simulations that involve the MATLAB® programming interface as jobs for Condor®, MATLAB® must be installed on all computers in the Condor® pool. Please refer to the example `MonteCarloSimulationMatlab.txt` to get an idea of how to use Condor® with both PHOTOSS and MATLAB®. In this respect, the following additional remarks may be helpful:

- By default, Condor® executes job files on the remote computer using the user account and permissions of the user who submitted the job. Therefore you must have the permission to run MATLAB® on all computers in the pool, or your jobs might not be properly executed.
- If you do not have the permission to run MATLAB® on all computers, you might have to modify your Condor® configuration file as follows:
  - Uncomment the `CREDD_HOST` variable
  - Add CREDD to the `DAEMON_LIST` variable
  - Add the following lines to the configuration file:

    ```
    STARTER_ALLOW_RUNAS_OWNER = true
    CREDD_CACHE_LOCALLY = true
    SEC_CLIENT_AUTHENTICATION_METHODS = NTSSPI, PASSWORD
    ```

  - Restart Condor® with the net stop condor and net start condor commands
  - Depending on your user account permissions, you might have to add the following line to your job file(s):
    `run_as_owner = true`
- Keep in mind that Condor® will copy all input files into a single temporary directory before executing the job. You may therefore have to review all absolute or relative path settings that are used in the MATLAB® component or your MATLAB® scripts.

6.3.4 Troubleshooting

- Check which machines are available in your Condor® pool by entering
  `c:\condor\bin\condor_status -available` on the command line.
- Before submitting Condor® jobs, make sure to store your user credentials by typing
  `c:\condor\bin\condor_store_cred add` on the command line. When prompted, enter your user password.
- First try the CheckCondorFunctionality.txt example. This example only deals with Condor® itself and does not involve PHOTOSS or MATLAB®. If this example does not yield the expected result files, then Condor® is probably not properly installed or configured.
- Make sure that you have PHOTOSS installed on every computer in the Condor® pool, including the Visual C++ Redistributable and the WibuKey drivers. The WibuKey software must be configured as described in the PHOTOSS Setup Manual, or else your submitted PHOTOSS jobs will show up an error on the remote machine and remain in *running* status without ever finishing.
- Then try examples that only utilize Condor® and PHOTOSS, but do not involve MATLAB®. If these fail, the problem is probably related to PHOTOSS settings.
- Finally try combinations of Condor®, PHOTOSS, and MATLAB® to detect any problems that are related to MATLAB®. Keep in mind that you must have the permission to run MATLAB® on all computers in the pool, and have your user privileges set accordingly.
- Be patient when running Condor® jobs that involve MATLAB®. Keep in mind that a new instance of a MATLAB® workspace must be opened before the simulation can actually start.

- Never submit more jobs that involve `PHOTOSS_grid.exe` than you own PHOTOSS grid licenses. PHOTOSS typically comes with three grid licenses. If you submit more jobs, these will show up an error on the remote machine and remain in *running* status without ever finishing.

- Keep in mind that some operating systems (such as Windows® Vista) may be very choosy about user privileges such as write permissions.

- By default, Condor® will only try to run a submitted job on computers with the **same operating system**. For example, if you submit a job from a computer that is the only one running Windows® Vista (while all others are running Windows® XP), your job may not be accepted by any other computer, but only by the one that submitted the job. If this one is busy or configured to not accept any jobs at all, then the job will remain idle until you remove it manually. If you add the command

  ```
  Requirements = (Arch == "INTEL" && (OpSys == "WINNT51"||OpSys== \
  "WINNT52"||OpSys == "WINNT60"||OpSys == "WINNT61"))
  ```

  to your job file, you allow Condor® to run the job on any available computer with either Windows® XP, Windows® Vista or Windows® 7. The Condor® examples that come with PHOTOSS include the above command. Please refer to the Condor® documentation for a more detailed description of job requirements and how to adapt your job files.

7 PHOTOSS Components

7.1 Sources

7.1.1 CW Laser

Fundamentals

The cw laser is the idealized form of the single mode laser. It is noiseless and generates a single spectral line with the power and center frequency defined in the parameters. The normalized electrical field can be expressed in complex representation as

$$E(t) = \sqrt{P}\exp\left(j\left(2\pi\left(f_o - f_c\right)t\right)\right), \tag{1}$$

where f_c stands for the center frequency defined in the *simulation parameters* (see page 27). Furthermore, it is possible to add a Gaussian distributed phase noise. This leads to a broadening of the laser linewidth.

Input / Output

	Number	Type of signal
Input	0	None
Output	1	Optical field

Component Parameters

Parameter Name	P
Description	Output power
Default Value	0.001
Parameter Range	0 ≤ P ≤ 5
Unit	W

Parameter Name	f0
Description	Center frequency
Default Value	193.1
Parameter Range	within frequency range of simulation parameters around the simulation parameters center frequency
Unit	THz

Parameter Name	linewidth
Description	Laser linewidth
Default Value	0
Parameter Range	0 ≤ linewidth ≤ 0.01
Unit	nm

References

none

7.1.2 Multi Mode Laser

Fundamentals

The model of the single mode laser explained in the section above can easily be extended to multimode lasers. For brevity, only the equations which have to be modified are described. Each laser mode is represented by a rate equation for the photon number and the optical phase. The equation for the charge carriers includes the contributions of all photons related to the different laser modes. Applying this formalism results in [1, 2]:

$$\frac{dS_{pi}(t)}{dt} = \left(G_i(t) - \frac{1}{\tau_p}\right) S_{p,i}(t) + R_{sp}(t) + F_{S_{p,i}}(t) \tag{1}$$

$$\frac{dN(t)}{dt} = \frac{I(t)}{q} - \frac{N(t)}{\tau_e} - \sum_i G_i(t) S_{p,i}(t) + F_N(t) \tag{2}$$

$$\frac{d\varphi_i(t)}{dt} = \frac{1}{2}\alpha_H \frac{dG_i(t)}{dN}(N(t) - N_{dc}) + F_{\varphi,i}(t) \tag{3}$$

$S_{P,i}(t)$:	photon population of laser mode i
$N(t)$:	overall charge carrier population
$\varphi_i(t)$:	optical phase of laser mode i
$G_i(t)$:	gain of mode i
R_{sp} :	effective rate of spontaneous emission
τ_p :	photon lifetime
τ_e :	carrier lifetime
q :	electron charge
α_H :	linewidth enhancement factor (Henry factor)
N_{dc} :	carrier population at average current (DC part)
F_k :	Langevin noise forces

Equations (1) and (3) describe the optical field of each individual mode, equation (2) the overall electron carrier density, and *i* is the mode index.
The spectral dependence of the gain can be modeled using either a parabolic gain shape [1, 2]

$$G_i(t) = G_{sm}(t)\left(1 - \left(\frac{f_i - f_c}{\Delta f_{FWHM}}\right)^2\right) \tag{4}$$

$G_{sm}(t)$:	gain function of single mode laser (see previous section)
f_c :	frequency of gain maximum
f_i :	frequency of i-th laser mode
Δf_{FWHM} :	FWHM bandwidth of gain function

or an input file containing the center frequencies of the different laser modes and a gain attenuation factor for each mode explicitly

$$G_i(t) = \alpha_i G_{sm}(t) \tag{5}$$

α_i :	attenuation factor of *i*-th laser mode

G_{sm} is given by eqs. (5) and (6) on page 155 of the section Single Mode Laser. The noise properties of the laser are modeled by Langevin noise sources, which are represented by

$$F_{S,i}(t) = \sqrt{\frac{2S_i(t)\,R_{sp}(t)}{\Delta t}}\,x_{S,i}(t) \tag{6}$$

$$F_N(t) = \sqrt{\frac{2N(t)}{\tau_e \Delta t}}\,x_N(t) - \sum_i \sqrt{\frac{2S_i(t)\,R_{sp}(t)}{\Delta t}}\,x_{S,i}(t) \tag{7}$$

$$F_{\varphi,i}(t) = \sqrt{\frac{R_{sp}(t)}{2S_i(t)\,\Delta t}}\,x_{\varphi,i}(t) \tag{8}$$

x_k :	normalized Gaussian random process, $\langle x_k \rangle = 0$ and $\left\langle x_k^2 \right\rangle = 1$
t :	time step for discretization

All other relations can be taken from the single mode laser model. The electric field at the laser output is given by

$$E(t) = \sum_i \sqrt{P_i(t)}\exp\left(j\left(2\pi f_{0,i} t + \varphi_i(t)\right)\right) \tag{9}$$

The mode power is calculated utilizing

$$P_i(t) = \frac{1}{2} h f_i v_g \alpha_m S_i(t) \tag{10}$$

h :	Planck's constant
f_i :	mission frequency of i-th mode

Input / Output

	Number	CW-Usage	Signal-Type
Input	1	On	None
Input	1	Off	Electric current (pulse generator)
Output	1	On/Off	Optical field

Component Parameters

Parameter Name	Ld
Description	length of active region
Bulk structure	3.4e-4
MQW structure	3e-4
Parameter Range	1e-6 ≤ Ld ≤ 1e-2
Unit	m

Parameter Name	V
Description	volume of active region
Bulk structure	1e-16
MQW structure	1.53e-17
Parameter Range	1e-20 ≤ V ≤ 1e-10
Unit	m^3

Parameter Name	Gamma
Description	mode confinement factor
Bulk structure	0.4
MQW structure	0.06
Parameter Range	0 < Gamma < 1

Parameter Name	Index
Description	group index
Default Value	3.8
Parameter Range	1 ≤ Index

Parameter Name	R
Description	facet reflectivity
Bulk structure	0.32
MQW structure	0.28
Parameter Range	0 < R < 1

Parameter Name	a_int
Description	intrinsic loss
Bulk structure	4400
MQW structure	4100
Parameter Range	0 < a_int < 1e6
Unit	m^{-1}

Parameter Name	a_diff
Description	idiff. gain coefficient
Bulk structure	2e-20
MQW structure	3e-12
Parameter Range	1e-10 < a_diff < 1e-3
Unit	m^2

Parameter Name	k
Description	gain compression coefficient
Bulk structure	5e-8
MQW structure	1.57e-7
Parameter Range	1e-12 < k < 1e-3

Parameter Name	A
Description	coeff. of non-radiative recombination
Default Value	1e8
Parameter Range	1e6 ≤ A ≤ 1e12

Parameter Name	B
Description	coeff. of radiative recombination
Default Value	1.5e-16
Parameter Range	1e-20 ≤ B ≤ 1e-10
Unit	m^3/s

Parameter Name	C
Description	oeff. of Auger recombination
Default Value	4.5e-41
Parameter Range	1e-60 ≤ C ≤ 1e-30
Unit	m^6/s

Parameter Name	BetaSP
Description	spontaneous emission factor
Bulk structure	1e-5
MQW structure	5e-5
Parameter Range	BetaSP < 1

Parameter Name	N0
Description	carrier number at transparency
Bulk structure	1e8
MQW structure	2e7
Parameter Range	1e3 ≤ N0 ≤ 1e12

Parameter Name	alpha
Description	line width enhancement factor
Bulk structure	5
MQW structure	3
Parameter Range	1 ≤ alpha

Parameter Name	t_Step
Description	time discretion
Default Value	0.5e-12
Parameter Range	0.01 ≤ t_Step ≤ 10
Unit	ps

Parameter Name	f_0
Description	carrier frequency
Default Value	193.1
Parameter Range	within frequency range of simulation parameters around the simulation parameters center frequency
Unit	THz

Parameter Name	CW
Description	CW mode on/off
Default Value	false
Parameter Range	boolean

Parameter Name	I_cw
Description	CW current
Default Value	5e-002
Parameter Range	0 ≤ I_cw ≤ 1e6
Unit	A

Parameter Name	warmup_time
Description	warmup time
Default Value	54000
Parameter Range	0 ≤ warmup_time ≤ 1e6
Unit	ps

Parameter Name	LaserType
Description	laser type
Default Value	MQW
Parameter Range	Bulk, MQW

Parameter Name	Noise
Description	include noise yes/no
Default Value	true
Parameter Range	boolean

Parameter Name	Angle
Description	polarization angle
Default Value	0 (TE)
Parameter Range	-180 ≤ Angle ≤ 180
Unit	rad

Parameter Name	numModes
Description	p number of laser modes
Default Value	3
Parameter Range	1 ≤ numModes

Parameter Name	d_fModes
Description	frequency spacing (parabolic gain shape only)
Default Value	0.1
Parameter Range	1e-6 ≤ d_fModes ≤ 1

Parameter Name	gShape
Description	spectral gain shape)
Default Value	Parabolic
Parameter Range	Parabolic, Read From File

Parameter Name	B_FWHM
Description	FWHM width of gain function (parabolic gain shape only)
Default Value	0.5
Parameter Range	1e-6 ≤ B_FWHM ≤ 1
Unit	THz

Parameter Name	f_c
Description	center frequency of gain function (parabolic gain shape only)
Default Value	193.1
Parameter Range	within frequency range of simulation parameters around the simulation parameters center frequency
Unit	THz

Parameter Name	gFile
Description	file containing gain shape data (user defined gain only)
Default Value	none
Parameter Range	valid file path

The file containing the mode data (user defined gain shape only) has the following structure:

```
frequency [THz] gain attenuation [dB]
193.0        0
193.2        0.6
192.8        0.6
```

References

[1] G .P. Agrawal, N.K. Dutta, Semiconductor Laser, Van Nostrad Reinhold, New York, 1993.

[2] K. Petermann, Laser Diode Modulation and Noise; Kluwer Academic Publishers, Dordrecht, 1991.

7.1.3 Pulse Generator

Fundamentals

The pulse generator is a versatile instrument that **generates either electrical or optical pulse sequences**. It combines the functionality of a bit pattern generator, a bit coder, an electrical or optical pulse former, and optionally includes an ideal modulator. When used as an **electrical** device, it supplies the driver current (or voltage) for the modulation of optical sources, whereas in the **optical** mode it acts as an idealized optical transmitter.
Due to its universal functionality, the pulse generator offers a wide range of different options and parameters. The following sections give an explanation of each parameter group along with a description of the corresponding functionalities.

The following **Physical Parameters** exist:

- *Unit:* The operating mode is selected by specifying the unit of the generated output signal. The unit can be chosen to be Ampere for direct modulation of a laser, Volt for driving external modulator devices, or $\sqrt{W}$ for the generation of optical pulses.
- A_{min}, A_{max}: The peak-to-peak amplitude of the pulses is determined by its minimum and maximum values A_{min} and A_{max}, which correspond to the selected Unit parameter. For convenience, the amplitude of an optical signal is entered in W rather than $\sqrt{W}$! The pulse can also be defined by the average signal $A_{avg} = \frac{1}{2}[d \cdot A_{max} + (2 - d) \cdot A_{min}]$ with the *duty cycle d* (see the parameter "duty_cycle" below) and the extinction $A_{ext} = 10 \cdot \log_{10}(A_{max}/A_{min})$.

The Pulse Geometry consist of the parameters:

- *Pulse_Shaping_Mode:* Pulse shaping can either be applied to $\sqrt{A}$, A, or A^2 where A stands for the amplitude value given in the selected Unit.
- *Pulse_Shape:* Specifies the shape of the pulse edges. Available shapes are rectangle, squared cosine, sine, triangular, sawtooth up (leading edge), and sawtooth down (trailing edge).
  Two more shapes, Gaussian (including super-Gaussian) and Sech (Soliton pulse shape), specify not only the pulse edges but the complete pulse geometry. This leads to the following consequences (see figure 7.1.1):
  - They are applicable for RZ signals only (see *pulse_code*).
  - The roll-off factor α is ignored (see *Alpha_Roll-Off*).
  - The pulse has no plateau region, so the pulse width is exclusively defined by its duty cycle d (see *duty_cycle*).
  - Unlike all other pulse forms, Gaussian and Sech (Solion) pulses may extend over the bit boundaries and overlap with precedent or subsequent pulses.
- *Alpha_Roll-Off:* The edge roll-off factor α specifies the width of the pulse edges in terms of the bit duration T_{Bit} (see figures 7.1.1, 7.1.2). It can vary between 0 . . . 0.5 for RZ signals or 0 . . . 1 for NRZ signals (see *Pulse_Code*). α is ignored if Gaussian or Sech (Soliton) pulse shape is used (see *Pulse_Shape*).
- *Edge_Shape_Exponent:* Specifies an additional exponent for the shape of the pulse edges. In case of a Gaussian pulse type, a *shape_exponent* different from 1.0 will act as a super-Gaussian exponent, whereas for all other pulse geometries the edge functions are raised to the power of *shape_exponent*.

- *cut_off:* This parameter specifies the normalized amplitude ($0 < cut\_off \leq 1$) where non-vanishing Gaussian or Sech (Soliton) pulse edges are cut to zero level. The normalized amplitude always refers to the same domain where the pulse is shaped (see *Pulse_Shaping_Mode*). *cut_off* is ignored when pulse forms other than Gaussian or Sech are used (see *Pulse_Shape*).

The optical signal properties are listed below:

- f_0: Specifies the carrier frequency for the output signal in THz. If chosen different from zero, the output signal will be ideally modulated onto the carrier frequency f_0. When the pulse generator is used as an electrical device, f_0 is typically set to zero.

- *Chirp:* Switches the inclusion of chirp effects on or off. Has no effect for electrical pulse generators. Chirp is modeled according to [1]:

$$\frac{d\varphi}{dt} = \frac{\alpha_H}{2}\left[\frac{d}{dt}\ln P(t) + \kappa P(t)\right] \tag{1}$$

φ :	phase of the optical signal
α_H :	linewidth enhancement factor, Henry factor
κ :	adiabatic chirp constant
P :	optical power

and is properly normalized to ensure that the desired emission frequency is maintained.

- *alpha_h:* Is the dynamic chirp parameter α_H (linewidth enhancement factor, Henry factor). Has no effect for electrical pulse generators or if *chirp* is disabled.

- *kappa:* The adiabatic chirp parameter κ. Has no effect for electrical pulse generators or if *chirp* is disabled.

- *PowerIndepChirp:* Puts a linear or sine chirp on the pattern. This kind of chirp depends only on the parameter *ChirpScalingFactor*.

- *ChirpScalingFactor:* Chirp difference of the maximum and the minimum chirp in a bit slot.

- *Phase_Noise:* Enables or disables the generation of noise. Has no effect for electrical pulses.

- *CW_Linewidth:* The CW linewidth in THz. Applies to optical pulses only.

- *initial_phase:* Sets the initial optical signal phase to either zero or a random value.

The Bit Sequence is specified by the following parameters:

- *Bit_Rate:* The bit rate f_{Bit} in Gbit/s. The bit duration is derived by $T_{Bit} = 1/f_{Bit}$. Please see the Simulation Parameters Section on page 27 for details on how to perform mixed bitrate simulations.

- *Pattern_generation:* Determines whether a new random bit sequence is generated or if the bit sequence is read from an existing bitfile that contains a user defined bit pattern of arbitrary length. In the latter case, the *Input File Name* parameter must specify the corresponding filename. If you are using mixed bitrate simulations, it is recommended to read the PRBS from an existing bitfile.

- *PRBS_length:* If the *PRBS_length* value N is chosen to be different then zero, a $(2^N - 1)$ PRBS (*Pseudo Random Bit Sequence*) pattern is generated. If set to zero, the bit pattern is generated using the built-in random number generator instead of the PRBS algorithm. The *PRBS_length* parameter has no effect if an input bitfile is used.

- *Calc. bit shift:* If more than one pattern generator is used during the simulation, the bit sequence will be shifted, to minimize a correlation between the bits sequences, the sequences will be shifted. For example, a simulation which continues 3 pattern generators and a sequence contains 30 bits, the second generator will shift 10 and the third will shift 20 bits. If the value is set to *false*, the user can specify an own shift in *Bit shift*.

- *Loop_Mode:* If the input bitfile specified by the *Input File Name* parameter contains less bits than required for the simulation, the *Loop_ Mode* parameter lets you choose to extend the bit sequence by appending zero bits, continuously repeating the last bit, or periodically repeating the complete bit pattern. The *loop* parameter has no effect if a random or PRBS bit pattern is used (PRBS patterns are always periodically repeated).

- *Input File Name:* Specifies the filename for the input bitfile. A bitfile is a plain ASCII text file that contains exactly one bit value (0 or 1) per line. The *Input File Name* has no effect if a random or PRBS bit pattern is used.

- *Output File:* Enables or disables the creation of an output bitfile that contains the generated bit sequence. Even if a user defined bit sequence is used via an input bitfile, the generated bit sequence may differ due to internal or user defined modifications (see *Loop_Mode*, *NoPaddingBits*, *Bit_Code*).

- *Output File Name:* Specifies the filename for the output bitfile. The file format is the same as used for the input bitfile. The *outBitName* has no effect if *file_out* is disabled.

- *NoPaddingBits:* If the *cyclic convolution mode* is selected in the *simulation parameters* dialog, each signal block is considered periodically continued and therefore should not exhibit a signal discontinuity when repeated in order to avoid the creation of artifical frequency components. For non-chirping noise free signals, a periodic behavior is achieved by automatically adjusting the pulse edges at the block boundaries without changing the bit sequence or specified edge geometry. In some cases it can be helpful to enforce that each block is bounded by a certain number of padding bits of the same value both at the beginning and the end of the block in order to avoid (or minimize) signal discontinuities when the block is periodically repeated.
  If *NoPaddingBits* is chosen to be different than zero, the specified number of padding bits is inserted into the bit sequence at the beginning and the end of each signal block. Note that inserting padding bits will affect the bit sequence and therefore may yield a different bit pattern than specified by the input bitfile or the PRBS algorithm.
  The insertion of padding bits will not be performed if the *linear* convolution mode is chosen in the simulation parameters dialog, or if bit coding is applied (see *Bit_Code*). In either case, the *NoPaddingBits* value is ignored.

- *Padding_Bit:* Specifies the value of the padding bits (0 or 1). Has no effect if *NoPaddingBits* equals zero, or if no bit padding is performed due to other parameters (see *NoPaddingBits*).

The Bit Coding includes the following parameters:

- *Pulse_Code:* Specifies whether pulse forming is performed using NRZ (non return-to-zero) or RZ (return-to-zero) signals.

- *Duty_Cycle:* Specifies the duty cycle d of an RZ pulse (see figures 7.1.2, 7.1.3). Has no effect for NRZ signals. The FWHM (*Full Width at Half Maximum*) time $T_{FWHM} = dT_{Bit}$ is always related to the half maximum (HM) value of the optical power even if an electrical pulse generator is used. Therefore, the HM amplitude value A_{HM} depends on the selected Unit. For the electrical domain it can be derived using:

$$\begin{aligned} Electrical(A,V): \quad A_{HM} &= \frac{1}{2}(A_{max} - A_{min}) \\ Optical(\sqrt{W}): \quad A_{HM} &= \sqrt{\frac{1}{2}(A_{max}^2 - A_{min}^2)} \end{aligned} \tag{2}$$

Therefore, we have:

$$P_{opt,HM} = \frac{1}{2}(P_{opt,max} - P_{opt,min}) \tag{3}$$

If user defined values of the duty cycle d and the edge roll-off factor α lead to concurring requirements for the pulse shape, a corresponding warning message is displayed. The user can then choose to have either d or α automatically adjusted accordingly.

- *Bit_Code:* Specifies how the bit sequence is coded prior to pulse forming. While the option None maintains the original bit sequence, you can also choose to have the sequence coded using the *duobinary code* or the *AMI code*. Note that no padding bits can be inserted into the bit sequence when bit coding is used (see *NoPaddingBits*, P*adding_Bit(s)*). In addition, the bit sequence might have to be modified if cyclic convolution is selected in the *simulation parameters* dialog in order to maintain periodic behavior.
  The modification will be made automatically, but keep in mind that the resulting bit sequence is possibly no longer exactly the same as specified in an input bit file (see *Pattern_generation*, *Input File Name*) or by the *PRBS_length* option.

The low frequency dither can be described using the following parameters:

- *Dither:* Enables or disables frequency dithering. If enabled, a low frequency modulation (“dither”) is imposed on the signal according to

$$A(t) = A_0(t)\left[1 + m\cos(2\pi f_{dither} t)\right] \tag{4}$$

A_0 :	original electrical amplitude or optical power
A :	dithered electrical amplitude or optical power
m :	modulation depth
f_{dither} :	dither frequency

(Physical systems typically exhibit dither of the electrical amplitude resp. the optical power.)

- *Dither_Frequency:* The dither frequency f_{dither} in GHz. Has no effect if *Dither* is disabled.
- *Dither_Depth:* Specifies the modulation depth m. Has no effect if *Dither* is disabled.

The following polarization-related parameters can be specified:

1. *Polarization:* Appears only if polarization effects are activated in the *simulation parameters*. This property specifies the polarization of the signal. Default value is *constant*, but it can be set to *read from file*. Then an input file must be given, in which the polarization angles of every bit are listed one per line. Make sure, that there are enough values. E. g. if your signal consists of 64 bits per block, 64 values are required.
2. *Polarization Input File Name:* Appears only if polarization effects are activated in the simulation parameters and *Polarization* is set *to read from file*. This property specifies the name of the file that contains the polarization angles for each bit. Write one value per line, i.e. one line per bit.

Summary: Pulse geometry

This section gives a summary on how the pulse form is influenced by the different geometry parameters in dependence of the selected *Pulse_Code* (NRZ or RZ) and, in the RZ case, the pulse shape (⇒ *Pulse_Shape*). figures 1a-1c illustrate the resulting pulse geometry for a single “1” bit on a signal vs. time diagram.

The signal values (ordinate) correspond to the selected *Pulse_Shaping_Mode* parameter: If you choose the *Pulse_Shaping_Mode* to be $\sqrt{A}$ or A^2 rather than the electrical or optical amplitude A, the amplitude markers A_{min}, A_{max} and A_{HM} in figures 7.1.1-7.1.3 have to be replaced by their respective square root or power of two.

In the **NRZ case** (figure 7.1.1), the pulse geometry is determined by the amplitudes A_{min} and A_{max}, the bit rate $f_{Bit} = 1/T_{Bit}$ (⇒ *Bit_Rate*), and the edge roll-off factor α (⇒ *Alpha_Roll-Off*).

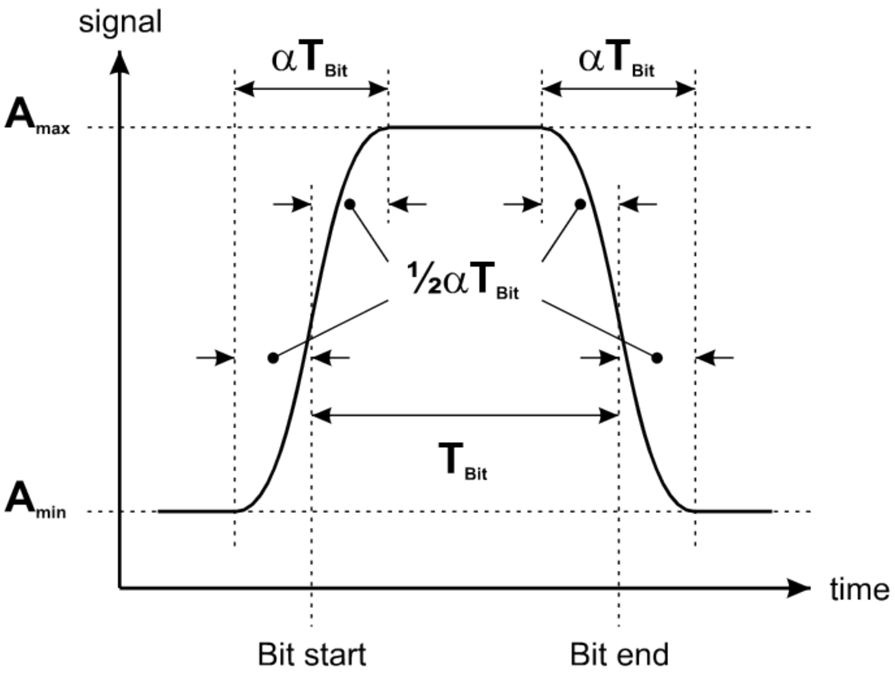


Figure 7.1.1. Pulse geometry for NRZ signals

The pulse edges are centered at the bit boundaries, so they extend into both adjacent bits by half their width αT_{Bit} each. If subsequent bits are both equal to 0 or 1, the respective plateau values A_{min} and A_{max} are kept constant over the edge time (NRZ), so that a single pulse may extend over several subsequent “1” bits.

In the **RZ case** (Figure 7.1.2), every “1” bit corresponds to a complete pulse including both leading and trailing edges. As in the NRZ case, the roll-off factor determines the edge width in terms of the bit duration TBit.

The position of the leading and trailing edges within the bit period is specified by the duty cycle d (⇒ *Duty_Cycle*). As d is related to the FWHM time $T_{FWHM} = dT_{Bit}$, it determines the time span between the two points where the leading and trailing edges reach the HM amplitude value A_{HM} according to equation (2). The time span T_{FWHM} is centered in the middle of the bit period T_{Bit}.

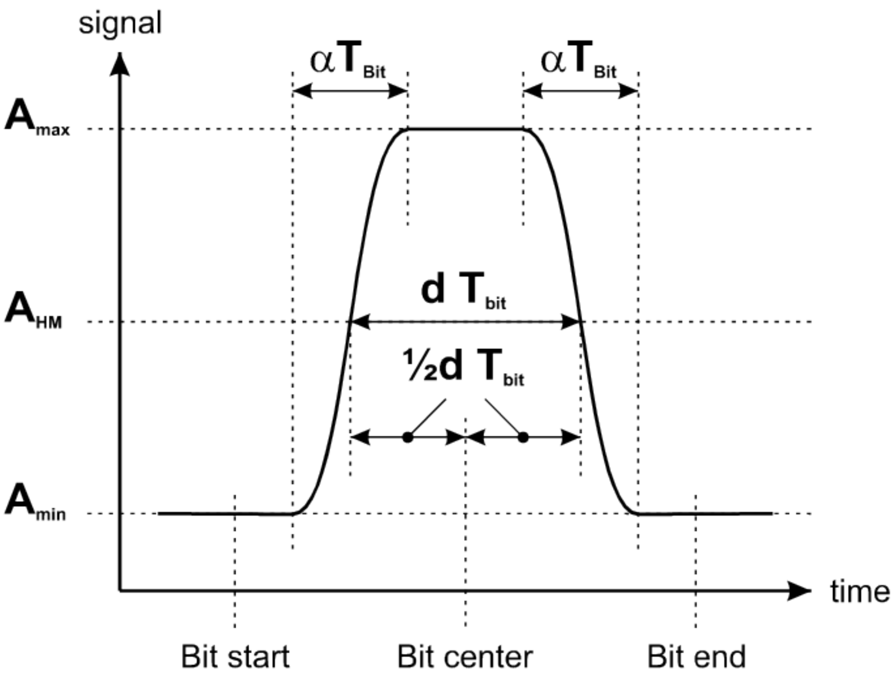


Figure 7.1.2. Pulse geometry for RZ signals

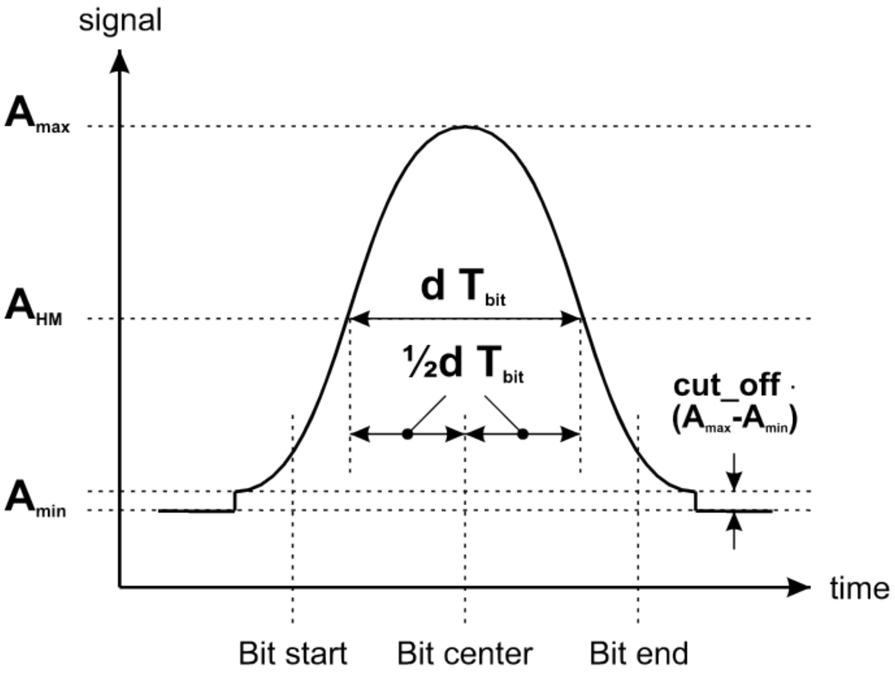


Figure 7.1.3. Pulse geometry for Gaussian/Sech RZ pulses

If **Gaussian or Sech (Soliton) RZ pulses** are selected, the pulse width is exclusively determined by the duty cycle d as shown in Figure 7.1.3. Therefore the bit pulses may extend into adjacent bit cells and overlap with precedent or subsequent pulses.
The roll-off factor α has no influence on the pulse geometry as there are no plateau regions before or after the edges. Due to the non-vanishing amplitude of Gaussian and Sech edges, they are cut to zero when the normalized amplitude *cut_off* is reached.

Input / Output

	Number	Type of signal
Input	0	None
Output	1	Electric current or voltage / optical field amplitude

Component Parameters

Parameter Name	Unit
Description	Operating mode/ Unit of the output signal Ampere: For direct modulation of a laser Volt: For driving external modulator devices Sqrt(Watt): For the generation of optical pulses
Laser driver	A
Modulator driver	V
Optical Source	$\sqrt{W}$
Unit	

Parameter Name	A_Min
Description	minimum amplitude (0)
Laser driver	0.03
Modulator driver	1.7
Optical Source	0.0 [W]
Unit	A / V / W

Parameter Name	A_Max
Description	maximum Amplitude (1)
Laser driver	0.07
Modulator driver	5.8
Optical Source	A / V / W
Unit	0.02 [W]

Parameter Name	Pulse_Shaping_Mode
Description	apply pulse shaping to
Unit	$\sqrt{\text{amplitude}}$, amplitude, power

Parameter Name	Pulse_Shape
Description	pulse shape (edges)
Laser driver	Rectangle, Squared_Cosine, Sine, Sech (soliton), Gaussian, Triangle, Saw_up Saw_down

Parameter Name	Alpha_Roll-Off
Description	roll-off factor
Parameter Range	0 . . . 1
Unit	

Parameter Name	Edge_Shape_Exponent
Description	Edge shape exponent (e.g. super-Gaussian exponent)
Default Value	1
Parameter Range	≥ 1
Unit	

Parameter Name	cut_off
Description	edge cut-off value
Default Value	0.0001
Parameter Range	$0 < \text{cut\_off} \leq 1$
Unit	

Parameter Name	f_0
Description	carrier frequency (optical pulses only)
Default Value Electrical	0
Default Value Optical	193.1
Parameter Range	within frequency range of simulation parameters around the simulation parameters center frequency
Unit	THz

Parameter Name	Chirp
Description	include chirp (optical pulses only)
Default Value	false
Parameter Range	boolean
Extended Parameter	true

Parameter Name	PowerIndepChirp
Description	Puts a chirp on the pattern
Default Value	no Cirp
Parameter Range	no chirp: The pattern is chirp-free linear chirp: Linear chirp in each bit slot (chirp = 0 in the center of the bit slot) sine chirp: Sine chirp in each bit slot (chirp = 0 in the center of the bit slot and at its borders) (The slope and the scaling are determined by the parameter "ChirpScalingFactor")
Extended Parameter	true

Parameter Name	alphaH
Description	chirp parameter, linewidth enhancement factor (optical pulses only)
Default Value	0.6
Parameter Range	$0 \leq \text{alphaH}$
Extended Parameter	true

Parameter Name	kappa
Description	adiabatic chirp parameter (optical pulses only)
Default Value	0.05
Parameter Range	no parameter limits
Extended Parameter	true

Parameter Name	Phase_Noise
Description	include phase noise
Default Value	false
Parameter Range	boolean
Extended Parameter	true

Parameter Name	CW_Linewidth
Description	CW linewidth
Default Value	0
Parameter Range	$0 \leq \text{CW\_Linewidth} \leq 1\text{E-}3$
Unit	THz
Extended Parameter	true

Parameter Name	initial_phase
Description	Initial signal phase
Default Value	random
Parameter Range	random zero

Parameter Name	Bit_Rate
Description	signal bit rate
Default Value	40
Parameter Range	$0 <$ Bit_Rate
Unit	Gbit/s

Parameter Name	Pattern_generation
Description	Algorithm for bit pattern generation
Default Value	Create random sequence
Parameter Range	Create random bit sequence read the bit sequence from a file single pulse sequence of 1´s alternating sequence

Parameter Name	PRBS_length
Description	Length of PRBS (Pseudo Random Bit Sequence) pattern
Default Value	6
Parameter Range	$0 \leq$ PRBS_length if set to 0, the built-in random number generator will be used instead otherwise the length is $2^x - 1$

Parameter Name	Loop_Mode
Description	loop mode
Default Value	repeat all bits

Parameter Range	append zeros
	repeat last bit
	repeat all bits

Extended Parameter	for input bit file only

Parameter Name	Input File Name
Description	name of input file (bits)
Default Value	none
Parameter Range	valid file path

Parameter Name	Output File
Description	generate output file
Default Value	false
Parameter Range	boolean

Parameter Name	Output File Name
Description	name of output file (bits)
Default Value	none
Parameter Range	valid file path

Parameter Name	NoPaddingBits
Description	number of padding bits (beginning/end of every block, cyclic convolution mode only)
Default Value	0
Parameter Range	$0 \leq$ NoPaddingBits

Parameter Name	Padding_Bit
Description	Specifies the value of the padding bits (0 or 1).
Default Value	0
Parameter Range	one (1) / zero (0)

Parameter Name	Pulse_Code
Description	Used pulse code
Default Value	NRZ
Parameter Range	signal coding return-to-zero (RZ), non return-to-zero (NRZ)

Parameter Name	Bit_Code
Description	Used bit code
Default Value	None
Parameter Range	None, Duobinary, AMI, PhaseSwitch, AmplitudeSwitch

Parameter Name	Format
Description	Used Modulation Format
Default Value	ASK
Parameter Range	ASK, DPSK

Parameter Name	Duty_Cycle
Description	duty cycle (RZ pulses only)
Default Value	1
Parameter Range	0 < Duty_Cycle < 1

Parameter Name	Dither
Description	use dithering
Default Value	false
Parameter Range	boolean
Extended Parameter	true

Parameter Name	Dither_Frequency
Description	dither frequency
Default Value	5
Parameter Range	0 ≤ Dither_Frequency
Unit	GHz
Extended Parameter	true

Parameter Name	Dither_Depth
Description	dither modulation depth
Default Value	0.2
Parameter Range	0 ≤ Dither_Depth
Unit	GHz
Extended Parameter	true

References

[1] T. P. Lee et al.: *Optical Telecommunications II, Chapter 12, "Light Emitting Diodes for Telecommunication"*, ed. by S. E. Miller, I. P. Kaminow, Academic Press, London (1988)

7.1.4 Single Mode Laser

Fundamentals

The behavior of the single mode laser is described by the large signal rate equations [1], [2].

$$\frac{dS_p(t)}{dt} = \left(G(t) - \frac{1}{\tau_p}\right) S_p(t) + R_{sp}(t) + F_{S_p}(t) \tag{1}$$

$$\frac{dN(t)}{dt} = \frac{I(t)}{q} - \frac{N(t)}{\tau_e} - G(t) S_p(t) + F_N(t) \tag{2}$$

$$\frac{d\varphi(t)}{dt} = \frac{1}{2}\alpha_H \frac{dG(t)}{dN}(N(t) - N_{dc}) + F_\varphi(t) \tag{3}$$

$S_p(t)$:	photon population
$N(t)$:	charge carrier population
$G(t)$:	gain
R_{sp} :	effective rate of spontaneous emission
τ_p :	photon lifetime
τ_e :	carrier lifetime
q :	electron charge
α_H :	linewidth enhancement factor (Henry factor)
N_{dc} :	carrier population at average current (DC part)
F_i :	Langevin noise forces

The chirp, i.e. the deviation of the instantaneous frequency with respect to the emission frequency, is given by

$$\delta f(t) = \frac{1}{2\pi}\frac{d\varphi}{dt} \tag{4}$$

The emission frequency is chosen such that the chirp is zero, if the laser diode is driven by the average dc current.
For a bulk structure, the gain of the laser is defined as a linear function of the carrier number:

$$G(t) = G_0 (N(t) - N_t) \frac{1}{1 + kS_p(t)} \tag{5}$$

k :	gain compression coefficient
N_t :	carrier number

In the case of an MQW-laser, a logarithmic gain dependence [3] is assumed:

$$G(t) = G_0 \ln\left(\frac{N(t)}{N_t}\right) \frac{1}{1 + kS_p(t)} \tag{6}$$

G_0 is defined in both cases by

$$G_0 = \frac{\Gamma a v_g}{V_{akt}} \tag{7}$$

Γ:	confinement factor
a:	differential gain coefficient
v_g:	group velocity
V_{akt}:	volume of active region

and the group velocity may be rewritten as

$$v_g = \frac{c_0}{n_g} \tag{8}$$

For eq. (3) dG/dN can be obtained using eqs. (5) and (6). The gain compression term is disregarded for these calculations.
The photon lifetime yields

$$\frac{1}{\tau_p} = v_g \left(\alpha_m + \alpha_{\text{int}}\right) \tag{9}$$

α_{int}:	intrinsic loss
α_m:	facet loss

where the facet loss depends on the facet reflectivity and the length of the active region according to

$$\alpha_m = \frac{1}{2L_{akt}} \ln\left(\frac{1}{R^2}\right) \tag{10}$$

L_{akt}:	length of active region
R:	reflectivity of the facets

The carrier lifetime is defined by

$$\tau_e^{-1} = R_A + R_B \frac{N(t)}{V_{akt}} + R_C \frac{CN(t)^2}{V_{akt}^2} \tag{11}$$

R_A:	coefficient of non-radiative recombination
R_B:	coefficient of spontaneous recombination
R_C:	coefficient of Auger recombination

and the effective rate of spontaneous emission may be written as

$$R_{sp}(t) = \beta_{sp} R_B \frac{N(t)^2}{V_{akt}} \tag{12}$$

β_{sp}:	spontaneous emission factor

Langevin noise forces are included in the rate equations to take into account the random fluctuations caused by the spontaneous emission and the generation and recombination of carriers. These effects are responsible for relative intensity noise (RIN) and phase noise.
The noise forces are modeled as memoryless. Their cross-correlation function is assumed to be a delta distribution [4]:

$$\langle F_i(t) \rangle = 0, \tag{13}$$

$$\left\langle F_i(t) F_j(\tau) \right\rangle = D_{ij} \delta(t-\tau) \tag{14}$$

D_{ij} are the diffusion coefficients, given by

$$D_{SS} = \left\langle R_{sp}(t) \right\rangle \langle S(t) \rangle \tag{15}$$

$$D_{NN} = \left\langle R_{sp}(t) \right\rangle \langle S(t) \rangle + \frac{\langle N(t) \rangle}{\tau_e} \tag{16}$$

$$D_{\varphi\varphi} = \frac{\left\langle R_{sp}(t) \right\rangle}{4 \langle S(t) \rangle} \tag{17}$$

$$D_{SN} = - \left\langle R_{sp}(t) \right\rangle \langle S(t) \rangle \tag{18}$$

$$D_{S\varphi} = D_{N\varphi} = 0 \tag{19}$$

The noise forces can be expressed by means of normalized Gaussian random processes [5]:

$$F_S(t) = \sqrt{\frac{2S(t) R_{sp}(t)}{\Delta t}} x_S(t) \tag{20}$$

$$F_N(t) = \sqrt{\frac{2N(t)}{\tau_e \Delta t}} x_N(t) - \sqrt{\frac{2S(t) R_{sp}(t)}{\Delta t}} x_S(t) \tag{21}$$

$$F_\varphi(t) = \sqrt{\frac{R_{sp}(t)}{2S(t) \Delta t}} x_\varphi(t) \tag{22}$$

x_i:	normalized Gaussian random process, $\langle x_i \rangle = 0$ and $\left\langle x_i^2 \right\rangle = 1$
Δt :	time step for discretization

For solving the rate equations (1)-(3) a Runge-Kutta scheme is used. In general, a time discretization of 1 ps is sufficient. To start the simulations with a steady state value for the photon number, the warm-up time has to be chosen appropriately. A random initial phase is assumed. The polarization angle of the emitted light can be set by the user.

The emitted power P is related to the photon number N by

$$P(t) = \frac{1}{2} h f_0 v_g \alpha_m S(t) \tag{23}$$

h :	Planck's constant
f_0 :	emission frequency

whereby the factor 0.5 accounts for an equal transmission through both facets.

Finally, the normalized electrical field can be expressed in complex representation as

$$E(t) = \sqrt{P(t)} \exp\left(j\left(2\pi f_0 t + \varphi(t)\right)\right) \tag{24}$$

Input / Output

	Number	CW-Usage	Signal-Type
Input	1	On	None
Input	1	Off	Electric current (pulse generator)
Output	1	On/Off	Optical field

Component Parameters

Parameter Name	Ld
Description	length of active region
Bulk structure [1], [6]	3.4e-4
MQW structure [3]	3e-4
Parameter Range	1e-6 ≤ Ld ≤ 1e-2
Unit	m

Parameter Name	V
Description	volume of active region
Bulk structure [1], [6]	1e-16
MQW structure [3]	1.53e-17
Parameter Range	1e-20 ≤ V ≤1e-10
Unit	m^3

Parameter Name	Gamma
Description	mode confinement factor
Bulk structure [1], [6]	0.4
MQW structure [3]	0.06
Parameter Range	0 ≤ Gamma ≤ 1

Parameter Name	Index
Description	group index
Default Value	3.8
Parameter Range	1 ≤ Index

Parameter Name	R
Description	facet reflectivity
Bulk structure [1], [6]	0.32
MQW structure [3]	0.28
Parameter Range	0 ≤ R ≤ 1

Parameter Name	a_int
Description	intrinsic loss
Bulk structure [1], [6]	4400
MQW structure [3]	4100
Parameter Range	0 ≤ a_int ≤ 1e-6
Unit	m^{-1}

Parameter Name	a_diff
Description	diff. gain coefficient
Bulk structure [1], [6]	2e-20
MQW structure [3]	3e-12
Parameter Range	1e-30 ≤ a_diff ≤ 1e-10
Unit	m^2

Parameter Name	k
Description	gain compression coefficient
Bulk structure [1], [6]	5e-8
MQW structure [3]	1.57e-7
Parameter Range	1e-12 ≤ k ≤ 1e-3

Parameter Name	A
Description	coeff. of non-radiative recombination
Default Value	1e8
Parameter Range	1e6 ≤ A ≤ 1e12

Parameter Name	B
Description	coeff. of radiative recombination
Default Value	1.5e-16
Parameter Range	1e-20 ≤ B ≤ 1e-10
Unit	m^3/s

Parameter Name	C
Description	oeff. of Auger recombination
Default Value	4.5e-41
Parameter Range	1e-60 ≤ C ≤ 1e-30
Unit	m^6/s

Parameter Name	BetaSP
Description	spontaneous emission factor
Bulk structure [1], [6]	1e-5
MQW structure [3]	5e-5
Parameter Range	BetaSP < 1

Parameter Name	N0
Description	carrier number at transparency
Bulk structure [1], [6]	1e8
MQW structure [3]	2e7
Parameter Range	1e3 ≤ N0 ≤ 1e12

Parameter Name	alpha
Description	line width enhancement factor
Bulk structure [1], [6]	5
MQW structure [3]	3
Parameter Range	1 ≤ alpha

Parameter Name	t_Step
Description	time discretion
Default Value	0.5e-12
Parameter Range	0.01 ≤ t_Step ≤ 10
Unit	ps

Parameter Name	f_0
Description	carrier frequency
Default Value	193.1
Parameter Range	within frequency range of simulation parameters around the simulation parameters center frequency
Unit	THz

Parameter Name	CW
Description	CW mode on/off
Default Value	false
Parameter Range	boolean

Parameter Name	I_cw
Description	CW current
Default Value	5e-002
Parameter Range	0 ≤ I_cw ≤ 1e6
Unit	A

Parameter Name	warmup_time
Description	warmup time
Default Value	54000
Parameter Range	0 ≤ warmup_time ≤ 1e6
Unit	ps

Parameter Name	LaserType
Description	laser type
Default Value	MQW
Parameter Range	Bulk, MQW

Parameter Name	Noise
Description	include noise yes/no
Default Value	true
Parameter Range	boolean

Parameter Name	Angle
Description	polarization angle
Default Value	0 (TE)
Parameter Range	-180 ≤ Angle ≤ 180
Unit	rad

References

[1] G .P. Agrawal, N.K. Dutta: *Semiconductor Laser*, Van Nostrad Reinhold, New York, 1993.

[2] K. Petermann: *Laser Diode Modulation and Noise*; Kluwer Academic Publishers, Dordrecht, 1991.

[3] H. Burkhard, S. Mohrdiek, H. Schöll, H. Hillmer, S. Hansmann, A. Mattheus: *Effects of Injected Light and Optical Feedback on Directly Modulated MQW Lasers and OTDM Multigigbit-Per-Second System Performance*, Preprint from Physics and Simulation of Optoelectronic Devices III, Proc. of the SPIE, Vol. 2399, 1995.

[4] C. H. Henry: *Phase Noise in Semiconductor Lasers*, J. Lightwave Technol., Vol. 4, No. 3, 1986, pp. 298-311.

[5] N. Schunk, K. Petermann: *Noise Analysis of Injection-Locked Semiconductor Injection Lasers*, IEEE J. Quantum Electron., Vol. 22, No. 5, 1986, pp. 642-650.

[6] S. Mohrdiek, H. Burkhard, H. Walter: *Chirp Reduction of Directly Modulated Semiconductor Lasers at 10 Gb/s by Strong CW Light Injection*, J. Lightwave Technol., Vol. 12, No. 3, 1994, pp. 418-424.

7.1.5 Single Pulse Generator

Fundamentals

The single pulse generator generates a single optical pulse with Gaussian or Soliton (sech) geometry. The pulse is defined by its peak power (*Max.Power* parameter), FWHM width (via the *Bit_Rate* and *Duty_Cycle* parameters), and carrier frequency (f_0). The extended *Edge_Shape_Exponent* parameter is an additional exponent applied on the pulse edges that acts as a super-Gaussian exponent for Gaussian pulses. The extended *cut_off* parameter specifies the normalized amplitude ($0 \leq cut\_off \leq 1$) where below which the pulse edges are set to zero level.

Input / Output

	Number	Type of signal
Input	0	None
Output	1	Optical field amplitude

Component Parameters

Parameter Name	Max.Power
Description	Maximum power of the pulse
Default Value	0.001
Parameter Range	0 ≤ Max.Power ≤ 10
Unit	W

Parameter Name	Pulse_Shape
Description	pulse shape (edges)
Default Value	Gaussian
Parameter Range	Sech (soliton), Gaussian

Parameter Name	Edge_Shape_Exponent
Description	pulse shape exponent (or super-Gaussian exponent)
Default Value	1
Parameter Range	1 ≤ Edge_Shape_Exponent

Parameter Name	f0
Description	carrier frequency (optical pulses only
Default Value	193.1
Parameter Range	within frequency range of simulation parameters around the simulation parameters center frequency
Unit	THz

Parameter Name	T_FWHM
Description	Full width at half maximum of the pulse's power
Default Value	12.5
Parameter Range	0 < T_FWHM ¡ bitDuration
Unit	ps

7.1.6 White Noise Source

Fundamentals

This component is able to generate an optical Gaussian distributed white noise. This noise is ideal and not related to the behavior of a physical component. The output noise power can be adjusted by setting the mean value of the component.

Input / Output

	Number	Type of signal
Input	0	None
Output	1	Optical field

Component Parameters

Parameter Name	Noise variance
Description	Mean value of the noise
Default Value	1E-8
Parameter Range	0 < Noise variance
Unit	variable (see below)

Parameter Name	Unit
Description	defines the unit of the parameter Noise variance
Default Value	sqrt(W)
Parameter Range	sqrt(W), Ampere, Volt

References

none

7.2 Modulators

7.2.1 Directional Coupler Modulator Fundamentals

A model of the directional coupler modulator (DCM) can be obtained using the theory of coupled modes.

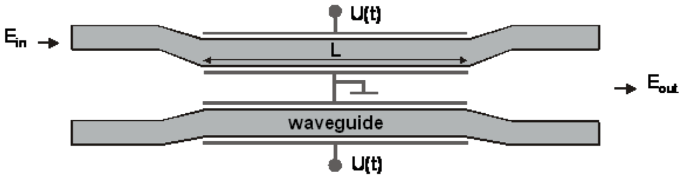


Figure 7.2.1. Directional coupler modulator

Under the condition that the length of the modulator equals its coupling length the calculations lead to the following equations [1-6]

$$E_1 = E_0 . \sqrt{\frac{\kappa^2 \cos^2\left(\sqrt{\kappa^2 + \eta^2 U^2(t)}L\right) + \eta^2 U^2(t)}{\kappa^2 + \eta^2 U^2(t)}} \cdot \exp\left(-j\beta_0 L + j\phi_1\right) \tag{1}$$

$$E_2 = E_0 . \frac{j\kappa \sin\left(\sqrt{\kappa^2 + \eta^2 U^2(t)}L\right)}{\sqrt{\kappa^2 + \eta^2 U^2(t)}} . \exp\left(-j\beta_0 L + j\phi_2\right) \tag{2}$$

$$\phi_1 = -\arctan\left(\frac{\eta U(t) \sin\left(\sqrt{\kappa^2 + \eta^2 U^2(t)}L\right)}{\sqrt{\kappa^2 + \eta^2 U^2(t)} \cos\left(\sqrt{\kappa^2 + \eta^2 U^2(t)}L\right)}\right) \tag{3}$$

$$\phi_2 = 0 \tag{4}$$

$$L = L_K = \frac{\pi}{2}\frac{1}{\kappa} \tag{5}$$

E_0 :	input electric field
E_1 :	output electric field, port 1
E_2 :	output electric field, port 2
κ :	coupling coefficient
η :	sensitivity
β_0 :	propagation constant
ϕ_1 :	phase, output port 1
ϕ_2 :	phase, output port 2
$U(t), \varphi_{out}$:	effective voltage (see below)
L :	electrode length
L_κ :	coupling length

As explained in the “Mach-Zehnder Modulator” section, the effective modulation voltage is calculated by filtering the externally applied voltage with a transmission function that describes the traveling wave effect [4-6].

Input / Output

	Number	Type of signal
Input	1	Optical field
Input	2	Modulating signal (Pulse generator)
Output	1	Optical field

Component Parameters

Parameter Name	L
Description	active modulator length
Default Value	8
Parameter Range	1e-3 $\leq$ L $\leq$ 1e3
Unit	mm

Parameter Name	alpha_0
Description	conductor loss
Default Value	0.55
Parameter Range	1e-3 $\leq$ alpha_0 $\leq$ 1e3
Unit	dB/cm GHz^2

Parameter Name	n_opt
Description	refractive index, optical
Default Value	2.17
Parameter Range	0 < n_opt

Parameter Name	n_el
Description	refractive index, electrical
Default Value	2.60
Parameter Range	0 < n_el

Parameter Name	eta
Description	sensitivity
Default Value	64.165
Parameter Range	0 < eta
Unit	1/Vm

Parameter Name	kappa
Description	coupling coefficient
Default Value	196.35
Parameter Range	0 < kappa
Unit	1/m

Parameter Name	D_0_1
Description	insertion loss, path 1
Default Value	6.2
Parameter Range	0 < D_0_1
Unit	dB

Parameter Name	phi_0_1
Description	phase shift, path 1
Default Value	0.0
Parameter Range	$-\pi \leq$ phi_0_1 $\leq +\pi$
Unit	rad

Parameter Name	D_0_2
Description	insertion loss, path 2
Default Value	6.7
Parameter Range	0 < D_0_12
Unit	dB

Parameter Name	phi_0_2
Description	phase shift, path 2
Default Value	0.0
Parameter Range	$-\pi \leq$ phi_0_2 $\leq +\pi$
Unit	rad

Parameter Name	output
Description	output branch
Default Value	1
Parameter Range	1, 2

References

[1] W. Goertz, “Analyse von Senderkonfigurationen für die optische Kommunikationstechnik”, pre-thesis at the Chair of High Frequency Technique, University of Dortmund, 1996

[2] F. Koyama, K. Iga, “Frequency Chirping in External Modulators”, IEEE Journal of Lightwave Technology, Vol. 6, No. 1, January 1988, pp. 87-93

[3] W. Karthe, R. Müller, “Integrierte Optik”, Akademische Verlagsges. Geest & Portig, Leipzig, 1991

[4] J.C. Cartledge, R.G. McKay, “Performance of 10 Gb/s lightwave systems using a adjustable chirp optical modulator and linear equalization”, IEEE Photonics Technology Letters, Vol. 4, No. 12, December 1992, pp. 1394-7

[5] H. Chung, W. Chang, E.L. Adler, “Computer modeling of Interferometric Microwave Travelling Wave Modulators and Switches in LiNbO3”, Workshop on Numerical Simulation and Analysis in Guided-Wave Optics and Optoelectronics, Houston, Texas, paper SF3

[6] S.K. Korotky, “Optimization of Traveling-Wave integrated-Optic Modulators”, Workshop on Numerical Simulation and Analysis in Guided-Wave Optics and Optoelectronics, Houston, Texas, paper SF2

7.2.2 Electro Absorption Modulator Fundamentals

The electro-absorption modulator (EAM) allows an external modulation of optical sources at high bit rates with low chirp. By means of the Franz-Keldysh (Bulk EAMs) or the quantum confined stark effect (MQW EAMs), the absorption of the device material can be controlled by an externally applied voltage.

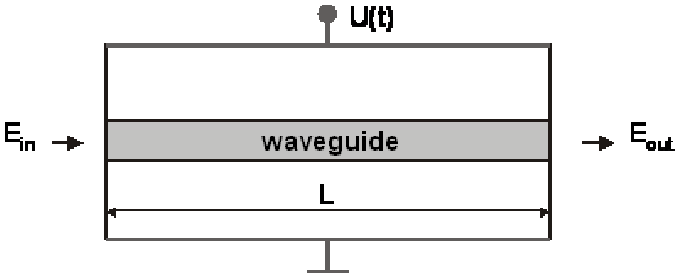


Figure 7.2.2. Electro-absorption modulator

The behavior of the EAM is phenomenologically resembled by the following equations [1-8]:

$$P_{out} = P_{in} D \exp\left[-\Gamma \frac{L}{L_0} \left(\frac{U(t)}{U_0} \right)^a \right] \tag{1}$$

$$\varphi_{out}(t) = \varphi_{in}(t) - \frac{1}{2} \alpha_H \Gamma \frac{L}{L_0} \left(\frac{U(t)}{U_0} \right)^a + \varphi_0(t) \tag{2}$$

P_{in}, P_{out} :	Optical power of the input (output) signal
D :	Insertion loss
Γ :	Confinement factor
L :	Length of active region (electrode)
L_0 :	reference length
$U(t)$:	effective modulation voltage (see below)
U_0 :	reference voltage
a :	exponential factor
$\varphi_{in}, \varphi_{out}$:	phase of input (output) signal
φ_0 :	initial phase value
α_H :	chirp parameter, Henry parameter

Due to the device parasitics the output power cannot follow the input voltage instantaneously. The effective modulation voltage is obtained by filtering the applied voltage with a lowpass filter of first order

$$H(f) = \frac{1}{1 + j2\pi f \tau_{RC}} \tag{3}$$

$H(f)$:	transmission function
τ_{RC} :	RC time constant

Input / Output

	Number	Type of signal
Input	1	Optical field
Input	2	Modulating signal (Pulse generator)
Output	1	Optical field

Component Parameters

Parameter Name	L
Description	length of active region (electrode)
Default Value	0.9
Parameter Range	1e-3 ≤ L ≤ 1e3
Unit	mm

Parameter Name	L_0
Description	reference length
Default Value	0.9
Parameter Range	1e-3 ≤ L_0 ≤ 1e3
Unit	mm

Parameter Name	tau_RC
Description	RC time constant (τ_{RC}, device parasitics)
Default Value	8
Parameter Range	1e-3 ≤ tau_RC ≤ 1e3
Unit	ps

Parameter Name	D
Description	insertion loss
Default Value	11.8
Parameter Range	0 ≤ D
Unit	dB

Parameter Name	phi_0
Description	constant phase offset (φ_0)
Default Value	0.0
Parameter Range	$-\pi \leq$ phi_0 $\leq +\pi$
Unit	rad

Parameter Name	Gamma
Description	confinement factor (Γ)
Default Value	0.05
Parameter Range	0 < Gamma < 1

Parameter Name	a
Description	exponential parameter
Default Value	4
Parameter Range	1e-3 < a < 1e3

Parameter Name	alpha_H
Description	chirp parameter, Henry factor (α_H)
Default Value	0.9
Parameter Range	0 ≤ alpha_H ≤ 1

Parameter Name	V0
Description	1/e-voltage
Default Value	1.608
Parameter Range	0 *y* V0 < 10
Unit	V

References

[1] W. Goertz, “Analyse von Senderkonfigurationen für die optische Kommunikationstechnik”, study thesis at the chair of High Frequency Technique, University of Dortmund, 1996

[2] F. Koyama, Iga, Kenichi: “Frequency Chirping in External Modulators”, IEEE Journal of Lightwave Technology, Vol. 6, No. 1, January 1988, pp. 87-93

[3] W. Karthe, R. Müller, “Integrierte Optik”, Akademische Verlagsgesellschaft Geest & Portig, Leipzig, 1991

[4] O. Mitomi, S. Nojima, I. Kotaka, K. Wakita, K. Kawano, M. Naganuma, “Chirping characteristic and frequency response of MQW optical intensity modulator”, Journal of Lightwave Technology, Vol. 10, No. 1, January 1992, pp. 71-7

[5] G. Grau, W. Freude, “Optische Nachrichtentechnik”, Springer-Verlag, 1991

[6] T.H. Wood, L.M. Ostar, M. Suzuki, “The effect of modulator nonlinearity on measurements of chirp in electroabsorption modulators”, Journal of Lightwave Technology, Vol. 12, No. 7, July 1994, pp. 1152-8

[7] E. Gay, M. Le Ligne, D. Hui Bon Hua, “Modélisation d’éléments optiques pour la simulation des systèmes de transmission”, L’Onde Électrique, Novembre-Décembre 1994, Vol. 74, No. 6, pp. 56-65

[8] P. Brindel, J.P. Hamaide, S. Landais, C. Starck, J.G. Provost, D. Lesterlin, “Compression of picosecond pulses by an electroabsorption modulator, in negative dispersion fibres”, Electronics Letters, vol.31, No. 10, May 1995, pp. 817-9

7.2.3 Frequency Modulator (ideal)

Fundamentals

The ideal frequency modulator imposes a frequency modulation on an electrical or optical carrier signal:

$$A_{out}(t) = A_{car}(t)\exp\left[j2\pi\Delta f\int\limits_0^t A_{\text{mod}}(\tau)\,d\tau\right] \tag{1}$$

A_{out} :	modulated output signal
A_{car} :	carrier signal
A_{mod} :	modulating signal
Δf :	frequency deviation

All signals are assumed to be in a complex baseband representation. A_{mod} is considered as a dimensionless quantity.

Input / Output

	Number	Type of signal
Input	1	Optical field / electrical signal
Input	2	Modulating signal (Pulse generator)
Output	1	Optical field / electrical signal

Component Parameters

Parameter Name	fd
Description	frequency deviation
Default Value	0.01
Parameter Range	1e-6 < fd < Frequency Range /2 of Simulation Parameters
Unit	THz

References

none

7.2.4 Ideal Modulator

Fundamentals

The ideal modulator can be used to convert an electrical signal to the optical domain with ideal properties (linear modulator). As an input a cw-laser has to be connected to the upper port. The lower port takes the incoming electrical signal. To enable an arbitrary power level the average power of the (optical) output signal can be defined in the component parameters.

Input / Output

	Number	Type of signal
Input	1	Optical field
Input	2	Modulating signal (electrical)
Output	1	Optical field

Component Parameters

Parameter Name	Pavg
Description	Average power of the output signal
Default Value	1
Parameter Range	0 < Pavg < 1000
Unit	mW

References

none

7.2.5 Mach-Zehnder Modulator

Fundamentals

The Mach-Zehnder (MZ) modulator is an intensity modulator based on an interferometric principle. It consists of two 3-dB-couplers which are connected by two waveguides of equal length. By means of the linear electro-optic effect an externally applied voltage can be used to force a refractive index variation in the waveguide branches.

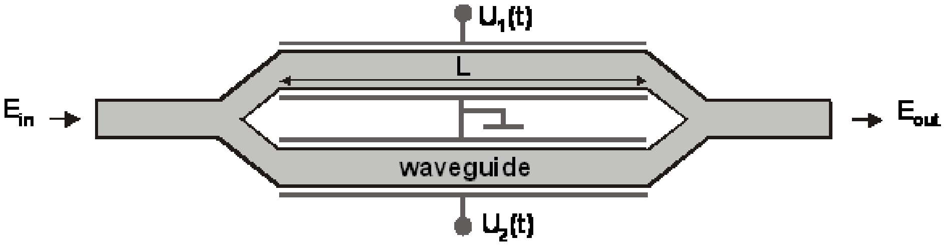


Figure 7.2.3. Mach-Zehnder Modulator

If the voltage is appropriately chosen, the propagation constants of the different paths lead to constructive or destructive interference in the 3 dB coupler at the output of the device. Thus the output intensity can be modulated between 0 and the input power (diminished by the insertion loss). Because of the large electro-optic coefficient, LiNbO$_3$ is commonly used as the material for MZ modulators. Due to the polarization sensitivity of LiNbO$_3$ devices, a polarization control is necessary. Under this assumption, the polarization dependence has not to be considered further. An advantage of MZ modulators is that an easy control of the frequency chirp is possible (by choosing a suitable bias voltage).
The fundamental equation describing the behavior of the MZ modulator is [1-10]

$$E_{out} = E_{in} \cdot D_0 \cdot \cos\left(\frac{\beta_2 - \beta_1}{2} L\right) \exp\left(-j\frac{\beta_2 + \beta_1}{2} L\right) \tag{1}$$

E_{in} :	electric field at the input
E_{out} :	electric field at the output
β_1 :	propagation constant, path 1
β_2 :	propagation constant, path 2
L :	length of the waveguides

The propagation constants β_1 and β_2 are given by

$$\beta_1(t) = \beta_0 - \eta_1 U(t) \tag{2}$$

$$\beta_2(t) = \beta_0 - \eta_2 U(t) \tag{3}$$

β_0 :	unperturbed propagation constant
η_1 :	sensitivity, path 1
η_2 :	sensitivity, path 2
$U(t)$:	effective modulation voltage

whereby the sensitivities characterize the strength of the linear electro-optical effect and can in general be different for the two waveguide branches due to an unequal field overlap.

If only one switch electrode is used (c.f. figure 7.2.3), the electric field at the device output may be written as

$$E_{out}\left(t\right) = E_{in}\left(t\right) \cdot D_0 \cdot \cos\left(\frac{\eta_1 - \eta_2}{2} LU\left(t\right)\right) \exp\left(j\frac{\eta_1 + \eta_2}{2} LU\left(t\right)\right) \exp\left(-j\beta_0 L\right) \tag{4}$$

The switching voltage is defined as the voltage which introduces a phase shift of π

$$U_\pi = \frac{\pi}{\eta_1 - \eta_2} \frac{1}{L} \tag{5}$$

According to (4) the maximum output power is obtained at $U(t) = 0$. However, an additional bias voltage U_0 is required for this switch state because of tolerances during to the fabrication process

$$E = E_0 \cdot D_0 \cdot \cos\left(\frac{\pi}{2} \frac{U\left(t\right) - U_0}{U_\pi}\right) \exp\left(j\vartheta \frac{\pi}{2} \frac{U\left(t\right)}{U_\pi}\right) \tag{6}$$

E_0 :	electric field at the input
E :	electric field at the output
ϑ :	chirp-factor
$U(t)$:	effective modulation voltage (see below)
U_π :	switching voltage
U_0 :	bias voltage
L :	waveguide length

$$\vartheta = \frac{\eta_1 + \eta_2}{\eta_1 - \eta_2} \tag{7}$$

describes the chirp of the modulator and can be expressed by the linewidth enhancement factor [2][3]

$$\alpha_H\left(t\right) = \vartheta \tan\left(\frac{\pi}{2} \frac{U\left(t\right) - U_0 - U_\pi}{U_\pi}\right). \tag{8}$$

Extending the equations to a Mach-Zehnder modulator with two electrodes leads to

$$E_{out}\left(t\right) = E_{in}\left(t\right) \cdot D_0 \cdot \cos\left(\frac{\pi}{2} \frac{U_1\left(t\right) - U_2\left(t\right) - U_0}{U_\pi}\right) \exp\left(j\frac{\pi}{2} \frac{U_1\left(t\right) + U_2\left(t\right)}{U_\pi}\right), \tag{9}$$

with

$$U_\pi = \frac{\pi}{\eta L} \tag{10}$$

and

$$U_2\left(t\right) = a \cdot U_1\left(t\right) + V_0 \tag{11}$$

E_{in} :	electric field at the input
E_{out} :	electric field at the output
$U_1(t)$:	effective modulation voltage, branch 1
$U_2(t)$:	effective modulation voltage, branch 2
U_π :	switching voltage
η :	Sensitivity of both paths
L :	length of waveguides
V_0 :	offset voltage
a :	voltage multiplicator

In this case, the linewidth enhancement factor is

$$\alpha_H(t) = -\frac{\frac{dU_1(t)}{dt} + \frac{dU_2(t)}{dt}}{\frac{dU_1(t)}{dt} - \frac{dU_2(t)}{dt}} \cot\left(\frac{\pi}{2}\frac{U_1(t) - U_2(t)}{U_\pi}\right) \tag{12}$$

In eq. (6) and (9) the insertion loss has to be added.

At high modulation frequencies, traveling wave electrodes [7-9] are commonly used. The effective modulation voltage is obtained by filtering the applied voltage by a transmission function which describes the different velocities of the optical and the electrical waves and also includes a frequency-dependent attenuation [10]

$$H(f) = \frac{1}{[\alpha(f) + j\gamma(f)]L}\left\{1 - e^{-[\alpha(f)+j\gamma(f)]L}\right\} \tag{13}$$

where the conductor loss exhibits a square-root dependence on the frequency of the applied voltage

$$\alpha(f) = \frac{\alpha_0}{2}\sqrt{f} \tag{14}$$

and

$$\gamma(f) = 2\pi f\frac{n_{el} - n_{opt}}{c} \tag{15}$$

$H(f)$:	transmission function
L :	electrode length
$\alpha(f)$:	conductor losses
α_0 :	conductor loss factor
$\gamma(f)$:	velocity mismatch
n_{opt} :	refractive index, optical
n_{el} :	refractive index, electrical

characterizes the difference between the propagation constant of the optical and the electrical wave.

Input / Output

	Number	Type of signal
Input	1	Optical field
Input	2	Modulating voltage (pulse generator)
Output	1	Optical field

Component Parameters

(LiNbO$_3$ based Mach-Zehnder modulator)

Parameter Name	alpha
Description	conductor loss
Default Value	0.55
Parameter Range	$0 \leq$ alpha ≤ 10
Unit	dB/(cmGHz$^{1/2}$)

Parameter Name	N_o
Description	refractive index, optical
Default Value	2.17
Parameter Range	0 ≤ N_o

Parameter Name	N_e
Description	refractive index, electrical
Default Value	2.17
Parameter Range	0 ≤ N_e

Parameter Name	D_0
Description	insertion loss
Default Value	5.1
Parameter Range	0 ≤ D_0 ≤ 10
Unit	dB

Parameter Name	phi_0
Description	constant phase shift
Default Value	0.0
Parameter Range	$-\pi$ < phi_0 ≤ π
Unit	rad
Extended Parameter	true

Parameter Name	U_pi
Description	switching voltage
Default Value (one electrode)	3.4
Default Value (two electrodes)	3.9
Parameter Range	-10 ≤ U_pi ≤ 10
Unit	V

Parameter Name	eta
Description	sensitivity of both paths
Default Value (two electrodes)	89
Parameter Range	-1000 ≤ eta ≤ 1000
Unit	1/Vm

Parameter Name	eta_1
Description	sensitivity, path 1
Default Value (one electrode)	89
Parameter Range	-1000 ≤ eta_1 ≤ 1000
Unit	1/Vm

Parameter Name	eta_2
Description	sensitivity, path 2
Default Value (one electrode)	-12.7
Parameter Range	-1000 ≤ eta_2 ≤ 100
Unit	1/Vm

Parameter Name	U_0
Description	bias voltage
Default Value	1.2
Parameter Range	-20 ≤ U_0 ≤ 20
Unit	V

Parameter Name	V_0
Description	offset voltage
Default Value (two electrodes)	0
Parameter Range	-20 ≤ V_0 ≤ 20
Unit	V

Parameter Name	a
Description	voltage multiplicator
Default Value (two electrodes)	-0.7
Parameter Range	-1000 ≤ a ≤ 1000

Parameter Name	electrodes
Description	number of electrodes
Default Value	1
Parameter Range	1,2

References

[1] W. Goertz, “Analyse von Senderkonfigurationen für die optische Kommunikationstechnik”, thesis at the Chair of High Frequency Technique, University of Dortmund, 1996

[2] A. Djupsjöbacka, “Residual chirp in integrated-optic modulators”, IEEE Photonics Technology Letters, Vol. 4, No. 1, January 1992, pp. 41-3

[3] M. Schiess, H. Carlden, “Evaluation of the chirp parameter of a Mach-Zehnder intensity modulator”, Electronics Letters, Vol. 30, No. 18, September 1994, pp. 1524-5

[4] J. C. Cartledge, C. Rolland, S. Lemerle, A. Solheim, “Theoretical performance of 10 Gb/s lightwave systems using a III-V semiconductor Mach-Zehnder modulator”, IEEE Photonics Technology Letters, Vol. 6, No. 2, February 1994, pp. 282-4

[5] Agrawal, P. Govind, “Fiber-Optic Communication Systems”, Verlag Wiley & Sons, 1992

[6] W. Karthe, R. Müller: “Integrierte Optik”, Akademische Verlagsgesellschaft Geest & Portig, Leipzig, 1991

[7] J. C. Cartledge, R.G. McKay, “Performance of 10 Gb/s lightwave systems using a adjustable chirp optical modulator and linear equalization”, IEEE Photonics Technology Letters, Vol. 4, No. 12, December 1992, pp. 1394-7

[8] H. Chung,; W. Chang, E. L. Adler, “Computer modeling of Interferometric Microwave Travelling Wave Modulators and Switches in LiNbO3”, Workshop on Numerical Simulation and Analysis in Guided-Wave Optics and Optoelectronics, Houston, Texas, paper SF3

[9] S. K. Korotky, “Optimization of Traveling-Wave integrated-Optic Modulators”, Workshop on Numerical Simulation and Analysis in Guided-Wave Optics and Optoelectronics, Houston, Texas, paper SF2

[10] F. Heismann, S.K. Korotky, et al., “Lithium Niobate Integrated Optics: Selected Contemporary Devices and System Applications”, in Optical Fiber Telecommunications IIIB, ed. I.P.Kaminow, T.L. Koch, Academic Press 1997

7.2.6 Phase Modulator (ideal)

Fundamentals

PSK (Phase Shift Keying):
In PSK mode, a phase modulation is imposed on an electrical or optical carrier signal:

$$A_{out}(t) = A_{car}(t) \exp\left[j\pi \frac{\Delta\varphi}{180^0} A_{\text{ mod }}(t) \right] \quad (1)$$

A_{out} :	modulated output signal
A_{car} :	carrier signal
A_{mod} :	modulating signal
$\Delta\varphi$:	frequency deviation

All signals are assumed to be in a complex baseband representation. A_{mod} is dimensionless. The signal A_{mod} will be normalized to the interval [0;1] so that the phase deviation may not exceed the user defined value.

DQPSK (Differential Quadrature PSK):
In DQPSK mode, two different signals I and Q are put onto the same carrier. The signals have to be modulated in an orthogonal way in order to minimize mutual interference. The constellation diagram of the four possible bit combinations is depicted below.

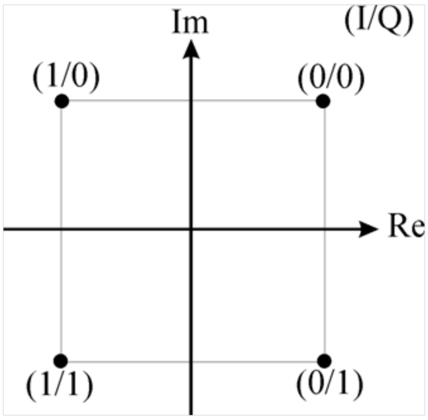


Figure 7.2.4. Constellation diagram of DQPSK modulation

Input / Output

	Number	Type of signal
Input	1	Optical field / electrical signal
Input	2	Modulating signal I
Input	3	Modulating signal Q
Output	1	Optical field / electrical signal

Component Parameters

Parameter Name	ModulationFormat
Description	switches between two operational modes
Default Value	PSL
Parameter Range	PSK, QPSK

Parameter Name	ph
Description	Phase deviation in degree
Default Value	90
Parameter Range	$-180 < ph \leq 180$
Unit	degree

Parameter Name	ModulationSource
Description	Modulation source
Default Value	Optical port
Parameter Range	Optical port, Electrical port

References

none

7.3 Fibers

7.3.1 Single Mode Fiber

Fundamentals

The physical behavior of a single mode fiber can be described by the following numerical model. This model includes different effects of an optical fiber such as fiber loss, group velocity dispersion (GVD), polarization mode dispersion (PMD), self-phase modulation (SPM), cross-phase modulation (XPM), four-wave mixing (FWM) and stimulated Raman scattering (SRS) [6, 7].
It is generally necessary to consider the combined effects simultaneously, since they affect each other. The wave envelopes of the two orthogonal linearly polarized modes obey the following system of coupled nonlinear equations [7, 8]:

$$\frac{\partial}{\partial z}A_i + \frac{\alpha}{2}A_i \mp \frac{1}{2}\Delta\beta_1\frac{\partial A_i}{\partial\tau} + \frac{j}{2}\beta_2\frac{\partial^2 A_i}{\partial\tau^2} - \frac{1}{6}\beta_3\frac{\partial^3 A_i}{\partial\tau^3} =$$

$$j\gamma\left\{|A_i|^2 A_i + \frac{j}{\omega_0}\frac{\partial}{\partial\tau}\left(|A_i|^2 A_i\right) + \frac{1}{3}{A_i}^*{A_j}^2 + \frac{2}{3}\left|A_j\right|^2 A_i - (\tau_{R1} + \tau_{R2})\,A_i\frac{\partial}{\partial\tau}|A_i|^2 - \tau_{R2}A_i\frac{\partial}{\partial\tau}\left|A_j\right|^2\right\} \qquad (1)$$

A_i, A_j :	Wave envelopes of the two orthogonal linearly polarized modes i, j (= 1,2)
$\beta(\omega)$:	Propagation constant
β_2 :	Group velocity dispersion (GVD) parameter
β_3 :	Third order dispersion (TOD) parameter
α :	Attenuation constant
$\Delta\beta_1$:	Differential group delay (see below)
γ :	Nonlinearity coefficient (see below)
τ_{R1}, τ_{R2} :	Raman time constants (parallel and orthogonal polarization)
z :	z - coordinate (points in the direction of propagation)
τ :	Time, measured in a frame of reference moving with the pulse at the group velocity

A center frequency ω_0 is assumed for A_i and A_j . The indices i und j denote the orthogonal modes. The nonlinearity coefficient is given by [4]

$$\gamma = \frac{n_2\omega_0}{c_0 A_{eff}} \qquad (2)$$

n_2 :	Nonlinear refractive index
A_{eff} :	Effective core area

The propagation constant $\beta(\omega)$ is expanded in a Taylor series around a reference frequency ω_r

$$\beta(\omega) = \beta_0 + \beta_1(\omega - \omega_r) + \frac{\beta_2}{2}(\omega - \omega_r)^2 + \frac{\beta_3}{6}(\omega - \omega_r)^3 + \frac{\beta_4}{24}(\omega - \omega_r)^4 + \ldots \qquad (3)$$

$$\beta_n = \left.\frac{\partial^n\beta}{\partial\omega^n}\right|_{\omega=\omega_r} \qquad (4)$$

which does not need to be equal to the simulation center frequency ω_0. The relationships between the coefficients of the series expansion and the dispersion parameters (which are normally used in data sheets of single mode fibers) are given by

$$D = \frac{-2\pi\cdot {f_0}^2}{c_0}\beta_2, \qquad (5)$$

$$S = \frac{4\pi \cdot {f_0}^3}{{c_0}^2}\beta_2 + \left(\frac{2\pi \cdot {f_0}^2}{c_0}\right)^2 \beta_3 \tag{6}$$

D :	Dispersion constant
S :	Dispersion slope constant
f_0 :	(Reference) frequency of the fiber

Both parameter sets (β_2, β_3 and D, S) are available in the component overview dialog. By default D, S are shown. Uncecking *Use S and D* will hide D, S and show β_2, β_3 instead.
The differential group delay $\Delta\beta_1$, which is used to describe the polarization mode dispersion, can be expressed as

$$\Delta\beta_1 = \beta_{1i} - \beta_{1j} \tag{7}$$

where β_{1i} is the propagation constant of the linearly polarized mode i.

When analyzing WDM-systems it is often helpful to represent the propagation equation as a set of coupled nonlinear differential equations. This approach is supported by PHOTOSS and is called Separated Channels Approach. In this case each WDM-channel is stored in a separate signal vector. Interactions between the WDM-channels (i.e. XPM or FWM) are modeled by certain coupling terms. In the case of FWM, only FWM crosstalk into the existing channels is modeled (within ±5 GHz of the center frequency). No new channels are opened. To allow accurate modeling of the FWM effect additional channels with no signal power may be added (compare FWM example file). Please note that for FWM calculation in SC-mode it is assumed that the channel power is concentrated at the center frequency of each WDM channel (cw-assumption). For calculations with channel frequency detuning and unequal channel spacing it is advised to use the TF-mode. The nonlinear Schrï¿½dinger equation with the relevant coupling terms is stated below [1]. It is valid for pulses with duration longer than 1 ps. SRS leads to a coupling of the different WDM-channels, and intra-channel SRS has been neglected. n depicts the channel number and the channels are sorted with increasing frequency. For the implementation in the case of two orthogonal polarizations (s. eqs. (20) and (21)).

$$\begin{aligned}\frac{\partial A_n}{\partial z} + \underbrace{\frac{\alpha}{2}A_n}_{\text{Attenuation}} + \underbrace{\frac{j}{2}\beta_{2n}\frac{\partial^2 A_n}{\partial \tau^2}}_{Dispersion} - \underbrace{\frac{1}{6}\beta_{3n}\frac{\partial^3 A_n}{\partial \tau^3}}_{Dispersion\,slope} &= \\ j\gamma A_n\left\{\underbrace{|A_n|^2}_{SPM} + \underbrace{2\sum_{i=1;i\neq n}^{N}|A_i|^2}_{XPM}\right\} + \underbrace{j\gamma\sum_{n=i+j-k;i,j\neq k} A_iA_jA_k^*\exp(-j\Delta kz)}_{FWM}& \\ \underbrace{-\sum_{i=1}^{n-1}\frac{\omega_n}{\omega_i}\frac{g_{ni}}{2}|A_i|^2 + \sum_{i=n+1}^{N}\frac{\omega_n}{\omega_i}\frac{g_{ni}}{2}|A_i|^2}_{SRS}&\end{aligned} \tag{8}$$

A_n :	Wave envelopes of the considered channel (single polarization)
β_{2n} :	Group velocity dispersion (GVD) parameter
β_{3n} :	Third order dispersion (TOD) parameter
α :	Attenuation constant
γ :	Nonlinearity coefficient
g_{ni} :	Raman gain coefficient
z :	coordinate (points in the direction of propagation)

with

$$\Delta k = k_i + k_j - k_k - k_n = -\beta_2 (\omega_i - \omega_k)\left(\omega_j - \omega_k\right) \\ -\beta_3 (\omega_i - \omega_k)\left(\omega_j - \omega_k\right)\left[\left(\omega_i + \omega_j\right)/2 - \omega_0\right] \tag{9}$$

Δk :	Wave number
ω_i :	Frequency of channel i

The following values are added to the **parameterized signal** values by the fiber.

- Delay (Separated Channels Approach):

$$delay_c = \beta_2 (\omega_c - \omega_0) + \frac{1}{2}\beta_3(\omega_c - \omega_0)^2 \tag{10}$$

- Delay (Total Field Approach):

$$delay_c = \begin{array}{l} \beta_2\left(\omega_c - \omega_{ref}\right) \\ +\frac{1}{2}\beta_3\left(\omega_c - \omega_{ref}\right)^2 \\ +\beta_3\left(\omega_c - \omega_{ref}\right)\left(\omega_0 - \omega_{ref}\right) \end{array} \tag{11}$$

- Accumulated group-velocity dispersion [ps/nm]:

$$D \cdot L = \left[D + S \cdot \left(\frac{c_0}{f_c} - \frac{c_0}{f_0}\right)\right] \cdot z \tag{12}$$

- Accumulated third-order dispersion [ps/nm^2]:

$$S \cdot L = S \cdot z \tag{13}$$

The power calculation of the parameterized signal depends on the alpha settings, described below.

ω_c :	Wavelength of channel c
ω_b :	Center wavelength of fiber
ω_{ref} :	Center wavelength of simulation parameters

Polarization mode dispersion (PMD) is an important effect in high bitrate communication systems. The linear part of the propagation equation (1) coincides with the waveplate model, whereby the step-size equals the waveplate length. The PMD-value of a long fiber is defined as:

$$PMD = \frac{\langle\Delta\tau\rangle}{\sqrt{L}} \tag{14}$$

L :	fiber length

Due to the waveplate model, the average differential group delay $\langle\Delta\tau\rangle$ (DGD) can be calculated from the step size LC and the dispersion of one waveplate β_1:

$$\langle\Delta\tau\rangle = \sqrt{\frac{8N}{3\pi}}.\beta_1.L_C \tag{15}$$

whereby N denotes the number of waveplates:

$$N = \frac{L}{L_C} \tag{16}$$

For a correct simulation of **higher order PMD**, a noise process can be added to the waveplate length [13, 14] by the parameter random process. A Gaussian- as well as a uniform distribution is available. The width of the random process is defined by the standard deviation **sigma**. PHOTOSS ensures that the length of the fiber is maintained after the additional random process.

For solving eq. (1) the user can choose between two main approaches. First, the fiber can be calculated linearly. By that, the fiber is calculated in one single step applying the fiber transfer function in the frequency domain. To do so, the option **Linear** in the calculation mode menu needs to be tagged. In the linear calculation mode, all terms on the right hand side of eq. (1), except the first one, are neglected (i.e. $\gamma = 0$).

The random coupling is modeled by a random change of the principle birefringence axis (θ) and a random phase shift (φ) between the two modes at the end of each fiber section (n). This description is valid for weak mode coupling [3]. It can be expressed as follows

$$\begin{bmatrix} A_x^n \\ A_y^n \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_n) & -\sin(\Delta\theta_n) \\ \sin(\Delta\theta_n) & \cos(\Delta\theta_n) \end{bmatrix} \begin{bmatrix} \mathrm{e}^{-j\omega\Delta\tau/2} & 0 \\ 0 & \mathrm{e}^{j\omega\Delta\tau/2} \end{bmatrix} \begin{bmatrix} \mathrm{e}^{-j\varphi/2} & 0 \\ 0 & \mathrm{e}^{j\varphi/2} \end{bmatrix} \cdot$$

$$\begin{bmatrix} \cos(\Delta\theta_n) & \sin(\Delta\theta_n) \\ -\sin(\Delta\theta_n) & \cos(\Delta\theta_n) \end{bmatrix} \begin{bmatrix} A_x^{n-1} \\ A_y^{n-1} \end{bmatrix} \tag{17}$$

with

$$\Delta\theta_n = \theta_n - \theta_{n-1} \tag{18}$$

The angles are described by discrete random processes

$$\theta_n = \theta_{n-1} + \pi\eta_1(1 - C_1) \tag{19}$$

where the random variable η_1 is uniformly distributed within the interval [-1, 1].

C_1 is describing the correlation between the angles θ of two consecutive fiber segments. A value of these constants of zero means that the consecutive angles are uncorrelated whereas one means that they are equal.
Optionally, a given set of angles φ and θ can be read from a file using the parameter *Phi* and *Theta*. In addition to this option the values of the two parameters can also be stored to a file by choosing the write option. A parameter is offered to edit the according filenames.
By default the angle φ is set to zero. However, you may read different values from a file.

Including nonlinearities in the calculation the fiber propagation is described using a split-step approach. The user is enabled to choose one of two modes for the calculation of the linear operator in eq. (1).

The options for the **linear** operator can be set in the linear calculation mode menu, where an **IIR-filter** time domain and a transfer function frequency domain approach (**FFT**) are offered. The **IIR-filter** option will no longer be supported by PHOTOSS since its application is limited to very special cases and the speedup is negligible. However, the option remains for reasons of compatibility.

The approaches for the **nonlinear** operator differ only if both polarization modes are used, i.e. if the flag *include polarization effects* is set in the *simulation parameters dialog*.
In the case of a **low birefringent fiber**, the coherent terms on the right hand side of eq. (1) are taken into account. A linear to circular transform is used to solve eq. (1) [1]. It is assumed that τ_{R2} is typically much smaller than τ_{R1}, so terms containing only τ_{R2} are set to zero.
In case of **high birefringence**, the coherent terms on the right hand side of eq. (1) will not be phase-matched. For that reason, they can be neglected. Again, terms containing only τ_{R2} are dropped.
To find a solution to eq. (1) without any approximations, a 4th-order **Runge-Kutta** scheme can be applied. In general, this requires very low step-sizes to account for the oscillatory terms of the propagation equation.
Fiber nonlinearities of both orthogonal polarization planes can be switched on and off separately. Using the parameter *IncludeParallelNl* the nonlinearities of the aligned polarization axes are switched on and off. Using the parameter *IncludeOrthogonalNl* the nonlinearities of the orthogonal polarization axes are switched on and off. Both parameters can be accessed using the *extended* button in the *component dialog*. *IncludeOrthogonalNl* is only available, if polarization effects are desired in the *simulation parameters dialog*. In Separated Channels mode the following equations (compare also eq. (8)) are used to model the PolXPM effect (XPM-induced birefringence of the fiber). Eqs. (20) and (21) model the XPM effect between orthogonal states of polarization accurately for fibers without PMD.

$$\begin{aligned}\frac{\partial A_{xi}}{\partial z} + \frac{j}{2}\beta_2(z)\frac{\partial^2 A_{xi}}{\partial t^2} &= j\gamma\left(|A_{xi}|^2 + \frac{2}{3}\left|A_{yi}\right|^2\right)\cdot A_{xi}\\ &+ j\gamma\sum_{j\neq i}^{N}\left(2\left|A_{xj}\right|^2 + \frac{2}{3}\left|A_{yj}\right|^2\right)\cdot A_{xi}\\ &+ \mathrm{j}\gamma\frac{2}{3}\sum_{j\neq i}^{N}\left(A_{xj}A_{yj}^{*} + A_{xj}^{*}A_{yj}\right)\cdot A_{yi} + j\gamma\frac{1}{3}A_{xi}^{*}A_{yi}^{2}\end{aligned} \tag{20}$$

$$\begin{aligned}\frac{\partial A_{yi}}{\partial z} + \frac{j}{2}\beta_2(z)\frac{\partial^2 A_{yi}}{\partial t^2} &= j\gamma\left(\left|A_{yi}\right|^2 + \frac{2}{3}|A_{xi}|^2\right)\cdot A_{yi}\\ &+ j\gamma\sum_{j\neq i}^{N}\left(2\left|A_{yj}\right|^2 + \frac{2}{3}\left|A_{xj}\right|^2\right)\cdot A_{yi}\\ &+ \mathrm{j}\gamma\frac{2}{3}\sum_{j\neq i}^{N}\left(A_{yj}A_{xj}^{*} + A_{yj}^{*}A_{xj}\right)\cdot A_{xi} + j\gamma\frac{1}{3}A_{yi}^{*}A_{xi}^{2}\end{aligned} \tag{21}$$

An **auto-stepping** algorithm can be used to calculate the **optimum discretization step-size** for the fiber propagation. (The parameter step-size is considered as maximum allowable step-size then.)
Whereas in the **cyclic convolution mode** the adaptive algorithm is applied repeatedly thus resulting in an increasing step-size towards the fiber end due to the decreasing influence of the fiber nonlinearities, in the **linear convolution mode** the fiber is simulated using a fixed predetermined step-size for efficiency reasons.
If the **adaptive** *step size control* is chosen, the parameter *step-size* should be specified as the full fiber length, due to the fact, that the step size control ensures a correct choice of the step width.

If the nonlinearity-coefficient γ equals zero or the calculation mode is set to **linear**, only the linear part of the propagation equation is solved. In this case a linear step-size-allocation algorithm is used, so that the parameter *maximum four-wave-mixing* and the *maximum nonlinear phase-shift* are without effect.

In the **cyclic convolution mode**, the algorithm is based on the maximum nonlinear phase shift and the tolerable suppression of artificial four-wave mixing 3 (multi-channel systems only), which is inherent in every spatial discretization scheme. Please note that in the case of very **low local dispersion fibers** (i.e. DSF or some NZDSF) and **equidistant channel spacing** the **FWM calculation** may lead to large fluctuations of the generated FWM products (depending on the initial phase in the pulse generators). To overcome this problem it is suggested to use a small detuning of the WDM channels (e.g. by ±0.5 GHz).

In the **linear convolution mode** (which is not supported from PHOTOSS 5.0 onwards) the fiber operates with causal impulse responses. To reduce the memory requirements as well as the latency time due to the causal network elements, the linear transfer function as well as the corresponding impulse response is appropriately windowed. Since the windowing process may induce distortions in the spectral GVD shape as well as ripples in the power transfer function, the **maximum tolerable GVD error** and the **maximum amplitude ripple** have to be specified. It is assumed, that only a fraction of the simulation bandwidth has to be described with highest accuracy. At the edges of the calculation window the linear fiber transfer function is attenuated to zero. The region where the filtered function has to agree with the theoretical fiber transfer function is given by the **relative fiber bandwidth**. The smaller the relative fiber bandwidth, the higher is the computational efficiency of the algorithm. To ensure that the pulse train in multi-channel systems cannot be shifted relative to each other by more than the block length. This condition is also checked when determining the optimum step-size in the linear convolution mode. The dispersion-induced walk-off between two pulses of different channels is limited in each step by the **maximum relative walk-off**. This walk-off is related to the shortest bit time of the two considered channels.

Raman pumping is a very attractive way to transmit a signal in a long distance optical transmission system. Due to the Raman pumping an equalized power profile is achieved along the fiber, compared to the use of lumped amplification, as for example using EDFAs. An equalized power profile is advantageous to keep signal distortion due to nonlinearities, especially self-phase modulation (SPM) small. The Raman effect operates as a stimulated spectral power transfer from higher towards lower frequencies. Thus, a Raman pump channel is coupled into the fiber at a frequency higher than the signal channel. Both forward and backward pumping is possible, whereby the pump-channel is dropped ideally from the signal spectrum after propagation over the fiber. Because the attenuation spectrum of the fiber is not flat (especially for pumps with a considerable spacing from the signal channels), the attenuation constants of the pump channels are set separately. The attenuation constant at the pump wavelengths is set to 0.5 dB/km as default. The Raman effect causes a **modified attenuation constant** α , which is constant for all signal wavelengths in the case of total field mode (**attention**: this behavior is only accurate, if the simulated pump configuration also leads to a flat gain spectrum for all signal wavelengths):

$$\alpha(z) = \alpha_{Fiber} - g_{RamanForward}(z) - g_{RamanBackward}(z) \tag{22}$$

In separated channels mode the attenuation constant of each signal wavelength is computed separately, leading to a spectral dependence of the Raman gain (i.e. to pumping of different signal wavelengths with different Raman gains). In both cases in the parameterized signal the spectral properties of the noise vector are modeled as wavelength dependent.
The **Raman gain** g_{Raman} spectrum is read from a file

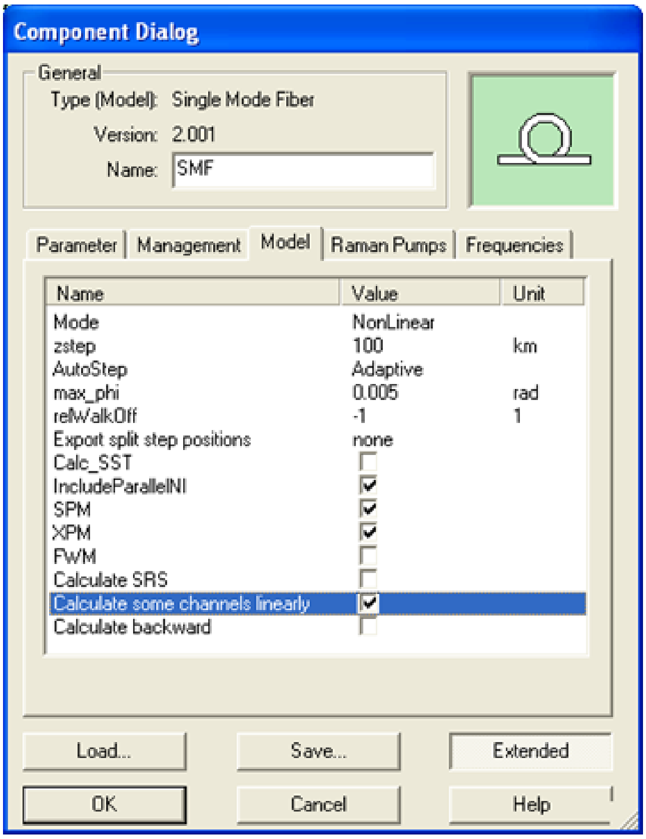
```
Comment
Standard Single Mode Fiber
Frequency [THz] Raman-Gain [m/W]
0 0
0.1 1.84E-16
0.2 3.68E-16
0.3 5.52E-16
```

The first three lines in the file are ignored and may be used for a comment. Afterwards in the first column the frequency shift in THz is defined and in the second column the Raman gain in m/W is given.

Alternatively, it is possible to read the modified attenuation constant from a file by setting the parameter alpha-progression to “read from file”. As an additional parameter it can be determined whether the data shall be imported in neper/km or dB/km. The format of this file is as follows (for both settings):

```
SSMF Raman Pump
km  neper/km  or dB/km
1 0.046051702
2 0.043749117
3 0.046051702
```

The first two lines can be comments of any kind. Afterwards in the first column there is the distance in km. At this position a split step will be inserted. The second column defines the attenuation coefficient for this split step in Neper/km (or dB/km). If additional split-steps are computed between the positions defined in the file, the attenuation profile is interpolated linearly. The first position has not to be zero, because it defines an end of a split step; the last position has to be the fiber length.



General
Type (Model): Single Mode Fiber
Version: 2.001
Name: SMF

Name	Value	Unit
Mode	NonLinear	
zstep	100	km
AutoStep	Adaptive	
max_phi	0.005	rad
relWalkOff	-1	1
Export split step positions	none	
Calc_SST	☐	
IncludeParallelNl	☑	
SPM	☑	
XPM	☑	
FWM	☐	
Calculate SRS	☐	
Calculate some channels linearly	☑	
Calculate backward	☐	

Figure 7.3.1. SMF Component Parameters Dialog

To speed up simulations with a high number of channels in separated channels mode it is possible to calculate the fiber nonlinearities only for a subset of channels. This enables the system designer to evaluate e.g. a fully-loaded system with 80 channels in reasonable time with only 10 channels computed nonlinearly. The remaining channels are considered to be impaired by linear effects only. Please note that the nonlinear coupling terms (eq. (8)) are only computed between the nonlinear channels. Also FWM products are only generated within the nonlinear channels. If “calculate some channels linearly” is selected from the model parameters, a new dialog page is opened where the frequencies of the channels, which should be computed nonlinearly, have to be defined.

Futhermore, it is possible to enable backward calculation of the fiber (*Calculate backward*). This option can be used to calculate a pre-distorted signal. PHOTOSS automatically inverts the nonlinearity coefficient of the fiber, the dispersion parameters and the attenuation coefficient. Also the split-step position are adopted accordingly. Please note that in this mode the fiber amplifies the input signal.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	alpha (α)
Description	Attenuation constant
Default Value	0.2
Parameter Range	-100 ≤ alpha ≤ 100
Unit	dB/km

Parameter Name	Alpha-Progression
Description	Allows to read attenuation profile from file (only works if Raman pumping is deactivated)
Default Value	constant
Parameter Range	constant, Read from file
Unit	dB/km

Parameter Name	Alpha-Progression Unit
Description	Allows to determine unit of imported attenuation profile
Default Value	Neper/km
Parameter Range	Neper/km, dB/km
Unit	variable

Parameter Name	Beta_2 (β_2)
Description	First order dispersion
Default Value	-21.75330296230518
Parameter Range	derived through D and S
Unit	ps^2/km

Parameter Name	Beta_3 (β_3)
Description	Second order dispersion
Default Value	0.1838787839607115
Parameter Range	derived through D and S
Unit	ps^2/km

Parameter Name	Use D and S
Description	Use either dispersion and dispersion slope parameters or β_2 and β_3
Default Value	true
Parameter Range	boolean

Parameter Name	f0
Description	Reference frequency, zero dispersion frequency
Default Value	193.1
Parameter Range	within frequency range of simulation parameters around the simulation parameters center frequency
Unit	THz

Parameter Name	length
Description	Fiber length
Default Value	100
Parameter Range	0 < length
Unit	km

Parameter Name	gamma (γ)
Description	Nonlinearity coefficient
Default Value	1.365
Parameter Range	-1e6 ≤ gamma ≤ 1e6
Unit	(W· km)

Parameter Name	Tau_1
Description	Raman time constant
Default Value	0.0
Parameter Range	0 ≤ Tau_1 ≤ BlockTime
Unit	ps

Parameter Name	n_2
Description	Nonlinear index coefficient
Default Value	2.69824387047475e-020
Parameter Range	-1e-10 ≤ n_2 ≤ 1e-10
Unit	m^2/W

Parameter Name	A_eff
Description	Effective core area
Default Value	80
Parameter Range	-1 < A_eff < 1e-3
Unit	μm^2

Parameter Name	rayleigh_coeff
Description	Rayleigh coefficient
Default Value	0.000631000000000001
Parameter Range	0 ≤ rayleigh_coeff < 1
Unit	1/km

Parameter Name	temperature
Description	Temperature
Default Value	293
Parameter Range	0 < temperature < 1000
Unit	K

Parameter Name	PMD value
Description	PMD value (only if polarization effects are activated)
Default Value	0.4370193722368316
Parameter Range	0 ≤ PMD value
Unit	ps/sqrt(km)

Parameter Name	DeltaBeta_1
Description	Differential group delay (only if polarization effects are activated)
Default Value	1.5
Parameter Range	0 ≤ DeltaBeta_1
Unit	ps/km

Parameter Name	Use PMD Value
Description	Use either PMD value or DeltaBeta_1 (works only, if random coupling and polarization effects are activated)
Default Value	false
Parameter Range	boolean

Component Parameters (Model)

Parameter Name	Mode
Description	Calculation algorithm
Default Value	linear
Parameter Range	linear, nonlinear

Parameter Name	zstep
Description	Maximum step-size
Default Value	100
Parameter Range	0 ≤ zstep ≤ length of the SMF
Unit	km

Parameter Name	AutoStep
Description	Auto stepping
Default Value	adaptive
Parameter Range	adaptive, uniform, import

Parameter Name	max_phi
Description	Maximum nonlinear phase-shift in one split-step
Default Value	0.001
Parameter Range	max_phi < 1
Extended Parameter	true
Unit	rad

Parameter Name	max_FWM
Description	FWM suppression (only in total field mode)
Default Value	30
Parameter Range	0 < max_FWM
Extended Parameter	true
Unit	dB

Parameter Name	relWalkOff
Description	Maximum relative walk off between two channels
Default Value	-1 (off)
Parameter Range	either -1 (deactivated) or 0 < relWalkOff <1
Extended Parameter	true

Parameter Name	Export split step positions
Description	Should the split step positions be exported?
Default Value	none
Parameter Range	none, write
Extended Parameter	true

Parameter Name	Export_filename_SSP
Description	File containing split-step positions
Default Value	none
Parameter Range	none, valid filename
Extended Parameter	true

Parameter Name	Calc_SST
Description	Calculate Self-Steepening Effect?
Default Value	false
Parameter Range	boolean
Extended Parameter	true

Parameter Name	IncludeParalle|N|
Description	Include nonlinearities of the parallel polarization?
Default Value	true
Parameter Range	boolean
Extended Parameter	true

Parameter Name	Birefringence Mode
Description	Calculation mode of the birefringence (only if polarization effects are activated)
Default Value	HighBiref
Parameter Range	HighBiref, LowBiref
Extended Parameter	true

Parameter Name	Rnd_Cpl
Description	Random mode coupling (only if polarization effects are activated)
Default Value	false
Parameter Range	boolean

Parameter Name	WavePlateLength
Description	Average length of a wave plate (only if random coupling is activated)
Default Value	0.1
Parameter Range	0 < WavePlateLength < length of the SMF
Unit	km

Parameter Name	WavePlateAssignment
Description	Method of wave plate length assignment (only if random coupling is activated)
Default Value	AssignUniform
Parameter Range	AssignUniform, AddUniformNoise, AddGaussianNoise, Read from file

Parameter Name	Import_Filename_WP
Description	File containing wave plate positions (only if WavePlateAssignment has been set to “Read from file”)
Default Value	none
Parameter Range	valid file path

Parameter Name	Jones_matrices
Description	Calculation of the Jones matrix (only if random coupling and total field are activated))
Default Value	centre frequency
Parameter Range	centre frequency, All frequencies

Parameter Name	Import/Export Theta
Description	Allows to read or write the rotation angles (only if random coupling is activated)
Default Value	none
Parameter Range	valid file path

Parameter Name	Import/Export Phi
Description	Allows to read or write the phase factors between the neighboring wave plates (only if random coupling is activated)
Default Value	none
Parameter Range	valid file path

Parameter Name	Export WavePlatePositions
Description	Write wave plate positions into a file (only if random coupling is activated)
Default Value	none
Parameter Range	none, write

Parameter Name	C1
Description	Correlation coefficient for random polarization change between two wave plates (only if random coupling is activated)
Default Value	0.0
Parameter Range	0 ≤ C1 ≤

Parameter Name	Raman_gain_file
Description	File containing Raman gain profile (only if Raman pumping or SRS is activated)
Default Value	none
Parameter Range	valid file path

Parameter Name	SPM
Description	Calculate SPM (only in separated channels mode))
Default Value	true
Parameter Range	boolean

Parameter Name	XPM
Description	Calculate XPM (only in separated channels mode)
Default Value	true
Parameter Range	boolean

Parameter Name	FWM
Description	Calculate FWM (only in separated channels mode)
Default Value	false
Parameter Range	boolean

Parameter Name	Calculate SRS
Description	Calculate SRS (only in separated channels mode)
Default Value	false
Parameter Range	boolean

Parameter Name	Calculate some channels linearly
Description	Only some channels are calculated incl. nonlinear fiber effects (only in separated channels mode)
Default Value	false
Parameter Range	boolean

Parameter Name	Calculate backward
Description	Inverse calculation of the fiber (negative nonlinearity, negative attenuation and inverse dispersion)
Default Value	false
Parameter Range	boolean

Standard Parameters (Raman Pumps)

Parameter Name	Calc_Raman
Description	Activate Raman pumping
Default Value	false
Parameter Range	boolean

Parameter Name	Include_noise
Description	Include noise in Raman calculation
Default Value	true
Parameter Range	boolean

Parameter Name	Noise_noise_Interaction
Description	Include noise-noise interaction
Default Value	false
Parameter Range	boolean

Parameter Name	Noise_signal_Interaction
Description	Include noise-signal interaction
Default Value	false
Parameter Range	boolean

Parameter Name	Noise_pump_Interaction
Description	Include noise-pump interaction
Default Value	false
Parameter Range	boolean

Parameter Name	Signal_noise_Interaction
Description	Include signal-noise interaction
Default Value	false
Parameter Range	boolean

Parameter Name	Signal_signal_Interaction
Description	Include signal-signal interaction
Default Value	true
Parameter Range	boolean

Parameter Name	Signal_Pump_Interaction
Description	Include signal-pump interaction
Default Value	true
Parameter Range	boolean

Parameter Name	Pump_Noise_Interaction
Description	Include pump-noise interaction
Default Value	true
Parameter Range	boolean

Parameter Name	Pump_Noise_Interaction
Description	Include pump-noise interaction
Default Value	true
Parameter Range	boolean

Parameter Name	Pump_Signal_Interaction
Description	Include pump-signal interaction
Default Value	true
Parameter Range	boolean

Parameter Name	Pump_Pump_Interaction
Description	Include pump-pump interaction
Default Value	true
Parameter Range	boolean

Parameter Name	#_of_sections
Description	Number of sections for Raman calculation
Default Value	100
Parameter Range	$1 \leq$ #_of_sections

Parameter Name	maximum deviation
Description	Maximum relative deviation between two iteration steps
Default Value	1e-6
Parameter Range	$0 \leq$ maximum deviation ≤ 1

Parameter Name	maximum iterations
Description	Maximum number of iteration steps
Default Value	100
Parameter Range	1 ≤ maximum iterations

Parameter Name	#ForwardPumps
Description	Number of pumps in forward directions
Default Value	0
Parameter Range	0 ≤ #ForwardPumps

Parameter Name	#BackwardPumps
Description	Number of pumps in backward directions
Default Value	0
Parameter Range	0 ≤ #BackwardPumps

Parameter Name	ForPumpFreq
Description	Forward Raman pump frequency
Default Value	206.1
Parameter Range	in the PHOTOSS simulation bandwidth and < 1000
Unit	THz

Parameter Name	BackPumpFreq
Description	Backward Raman pump frequency
Default Value	206.1
Parameter Range	in the PHOTOSS simulation bandwidth and < 1000
Unit	THz

Parameter Name	ForPumpPower
Description	Forward Raman pump power
Default Value	0.1
Parameter Range	0 < ForPumpPower < 10
Unit	W

Parameter Name	BackPumpPower
Description	Backward Raman pump power
Default Value	0.1
Parameter Range	0 < BackPumpPower < 10
Unit	W

Parameter Name	ForPumpAlpha
Description	Forward Raman pump attenuation constant
Default Value	0.5
Parameter Range	0 < ForPumpAlpha < 10
Unit	dB/km

Parameter Name	BackPumpAlpha
Description	Backward Raman pump attenuation constant
Default Value	0.5
Parameter Range	0 < BackPumpAlpha < 10
Unit	dB/km

Parameter Name	ForPumpAeff
Description	Forward Raman pump effective core area
Default Value	80
Parameter Range	1 < ForPumpAeff < 1000
Unit	μm^2

Parameter Name	BackPumpAeff
Description	Backward Raman pump effective core area
Default Value	80
Parameter Range	1 < BackPumpAeff < 1000
Unit	μm^2

Graphs

The Single Mode Fiber contains the following Visualizations:

- The phase of the transfer function can be plotted in the frequency domain.
- The amplitude of the transfer function can be plotted.
- The impulse response can be plotted.
- Raman gain spectrum (Raman pumping only).
- Pump signal gain evolution (Raman pumping only).
- Pump signal power evolution (Raman pumping only).

References

[1] C. De Angelis et al., “Soliton instabilities from resonant random mode coupling in birefringent optical fibers”, Opt. Lett., Vol. 17, No.16, No. 5, pp. 173-178, 1995.

[2] G. Bosco et al., “Suppression of Spurious Tones in Fiber Systems Simulations based on the Split-Step Method”, LEOS 1999, WH 4.

[3] C. Francia, “Constant Step-Size Analysis in Numerical Simulation for Correct Four-Wave-Mixing Power Evaluation in Optical Fiber Transmission Systems”, IEEE Photon. Technol. Lett., Vol. 11, No. 1, January 1999.

[4] G. P. Agrawal, “Nonlinear Fiber Optics”, Second Edition, Academic Press, 1995.

[5] E. A. Golovchenko, A. N. Pilipetskii, “Unified analysis of fourphoton mixing, modulational instability, and stimulated Raman scattering under various polarization conditions in fibers”, J. Opt. Soc. Am. B, Vol. 11, No. 1, pp. 92-101, 1994.

[6] R. Hellwarth, J. Cherlow, et l., “Origin and frequency dependence of nonlinear optical susceptibilities of glasses”, Phys. Rev. B, Vol. 11, No. 2, pp. 964-967, 1975.

[7] D. Marcuse, A. R. Chraplyvy, et al., “Effect of Fiber Nonlinearity on Long Distance Transmission”, J. Lightwave Tech., Vol. 9, No. 1, pp. 121-128, 1991.

[8] F. Matera, A. Mecozzi, et al., “Nonlinear evolution of polarization in long-haul fiber links: performence evaluation of polarization multiplexed systems”, conference paper, ECOC 91.

[9] K. J. Blow, D. Wood, “Theoretical Desciption of transient Stimulated Raman Scattering in Optical Fibers”, IEEE J. of Quantum Electronics, Vol 25, No. 12, pp. 2665-2673, 1989.

[10] J.-P. Elbers, “Modellierung und Simulation faseroptischer ï¿½bertragungssysteme”, Diploma thesis, Chair of High Frequency Technique, University of Dortmund, 1996.

[11] C. Glingener, “Modellierung und Simulation optischer Netze mit Wellenlï¿½ngenmultiplex”, PhD thesis, Chair of High Frequency Technique, University of Dortmund, 1996.

[12] C. D. Poole, R. E. Wagner, “Phenomenological Approach to Polarisation Dispersion in Long Single-Mode Fibres”, Electronics Letters, vol. 22, no. 19, Sep. 1986.

[13] C. H. Prola, J. A. Pereira da Silva, A. O. Dal Forno, R. Passy, J. P. von der Weid, N. Gisin, “PMD Emulators and Signal Distortion in 2.48 - Gb/s IM - DD Lightwave Systems”, IEEE Photonics Technology Letters, vol. 9, no. 6, June1997.

[14] A. O. Dal Forno, A. Paradisi, R. Passy, J. P. von der Weid, “Experimental and Theoretical Modeling of Polarization Mode Dispersion in Single-Mode Fibers”, IEEE Photonics Technology Letters, vol. 12, no. 3, March 2000.

7.4 Amplifiers

7.4.1 EDFA (Black Box)

Fundamentals

The black box EDFA model resembles an optical amplifier for which either a fixed gain or a fixed output power can be specified. The input power is amplified according to

$$P_{out} = G \cdot P_{in} \tag{1}$$

where the ASE noise contribution to the output power is assumed to be negligible.

P_{out} :	Output power
P_{in} :	Input power
G :	Constant gain

If a negative gain G is defined, the EDFA will act as a noise-free attenuator. Otherwise, there are two choices for the generation of noise:

- If the *adaptOSNR* parameter is set true, the desired optical signal-to-noise ratio (*OSNR*) at the EDFA output can be specified. However, as a physical EDFA can never improve the OSNR that is already present at its input, no noise will be generated, if the desired OSNR exceeds the input OSNR. Instead, the output OSNR will be set equal to the input OSNR. To override this functionality the parameter *Ignore OSNR restrictions* can be set to true, and an improvement of the OSNR is possible.

- If the *adaptOSNR* parameter is set *false*, a noise figure *NF* must be specified instead of an OSNR. Noise figures below zero result in noise-free EDFAs. Otherwise, noise is added in order to satisfy

$$NF = \frac{SNR_{in}}{SNR_{out}} \tag{2}$$

NF :	Noise figure
SNR_{in} :	Input SNR
SNR_{out} :	Output SNR

where the signal-to-noise ratios (*SNR*) at the input and output are assumed to be electrically measured behind a PIN receiver diode (and therefore differ from the OSNR). While SNR_in is affected by shot noise from the diode, SNR_{out} is primarily subject to ASE noise [1]. When both polarization modes and the amplified input noise are regarded, the power spectral density (PSD) of the output noise follows

$$N_{out}(f) = 2hfn_{sp}(G-1) + GN_{in}(f) \tag{3}$$

f :	Optical frequency
N_{out} :	PSD of output ASE
N_{in} :	PSD of input ASE
h :	Planck's constant
n_{sp} :	Spontaneous emission factor

The spontaneous emission factor n_sp is related to the noise figure NF according to [1]

$$n_{sp} = \frac{NF \cdot G - 1}{2\,(G-1)} \tag{4}$$

The physical limit $ofn_{sp} = 1$ results in an optimum noise figure of 3dB for amplifiers with high gain values.

In either case, white noise is added to the amplified input signal. It is modeled as thermal (incoherent) light that is generated using Gaussian random variables for the real and imaginary part of the electrical field. (Note: If you need to resemble spectrally shaped EDFA noise, please use the *Spectral EDFA* component model instead of this black box approach.)

If *use sampled power* is active the correct input power value is computed by using the sampled representation of the input signal instead of using the power value provided by the parameterized signal. This option improves accuracy and is only available in ’run together’ mode.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	Gainhandling
Description	Method of gain handling
Default Value	Constant gain
Parameter Range	Constant gain, Constant power

Parameter Name	Gain
Description	Amplifier gain
Default Value	10
Parameter Range	0 ≤ Gain ≤ 30
Unit	dB

Parameter Name	Pout
Description	Output saturation power (constant power EDFA only)
Default Value	5
Parameter Range	-30 ≤ Gain ≤ 30
Unit	dBm

Parameter Name	NF
Description	Amplifier noise figure
Default Value	5
Parameter Range	0 ≤ Gain ≤ 50
Unit	dB

Parameter Name	adaptOSNR
Description	Adapt the output OSNR
Default Value	false
Parameter Range	boolean

Parameter Name	Ignore OSNR restrictions
Description	Only applicable for analytical noise handling: EDFA may increase the OSNR value to the level determined by parameter *OSNR*
Default Value	false
Parameter Range	boolean

Parameter Name	Use sampled power
Description	Use sampled input signal to calculate input power. (more accurate, only available in *Combined Simulation* mode)
Default Value	false
Parameter Range	boolean

References

[1] Agrawal, P. Govind, “Fiber-Optic Communication Systems”, Wiley & Sons, 1992

[2] [2] E. Desurvire, “Erbium-Doped Fiber Amplifiers - Principles and Applications”, Wiley & Sons, 1994

7.4.2 EDFA (Spectral)

Fundamentals

Erbium-doped fiber amplifiers (EDFAs) are the most commonly used type of optical amplifiers for advanced optical communication systems. They combine advantages such as high gain over a broad spectral bandwidth, low noise figure, good signal polarization insensitivity, very low coupling losses and immunity to interference between different channels. The amplification process is always associated with the generation of noise, resulting from the amplified spontaneous emission (ASE).

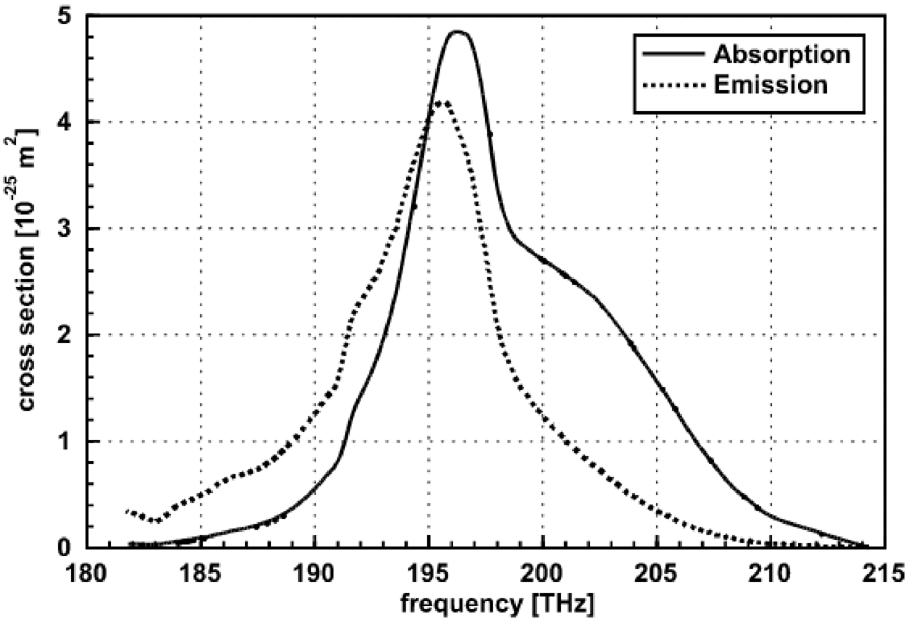


Figure 7.4.1. : Typical absorption and emission cross-sections

The behavior of an EDFA is described by first-order propagation and rate equations. The former shows how the pump power, signal powers of different channels and noise power at different wavelengths vary along the EDFA, the latter shows how the number of erbium ions in the different energy levels change. As there is no analytical solution for the general case, numerical methods have to be applied. Our model is based on the improved average power analysis technique [2]. This frequency dosimulation is suitable for multi-channel amplification and includes forward and backward generated spontaneous emission and the effect of saturation due to ASE power. We do not consider performance degradations due to polarization related effects [2] such as polarization dependent gain (PDG) due to anisotropic gain saturation and polarization dependent loss (f), which originates from components such as isolators. The impairments due to these effects are very small when considering only a few numbers of amplifiers.

The EDFA is treated as a two-level system and divided into a series of concatenated elemental amplifying sections.

$$\frac{dP_s^{i\pm}(z)}{dz} = \pm\left(\Gamma^i\left\{\left(\sigma_e^i+\sigma_a^i\right)N_2(z)-\sigma_a^iN_g\right\}-\alpha^i\right)P_s^{i\pm}(z) \tag{1}$$

and

$$\frac{dP_{ASE}^{j\pm}(z)}{dz} = \pm\left(\Gamma^j\left\{\left(\sigma_e^j+\sigma_a^j\right)N_2(z)-\sigma_a^jN_g\right\}-\alpha^j\right)P_{ASE}^{i\pm}(z) \pm 2hf^j\Delta f^j\sigma_e^j\Gamma^jN_2(z) \tag{2}$$

P_S :	Optical frequency
P_{ASE} :	ASE power per spectral slice
Γ :	Confinement factor
h :	Planck's constant
f :	Frequency
Δf :	PBandwidth of ASE slice
σ_e :	Cross section of stimulated emission
σ_a :	Cross section of stimulated absorption
N_2 :	Erbium density of level 2

describe the evolution of the signal and noise power, respectively.
Applied to an elemental amplifier section of length ΔL , the solution of (1) and (2) yields

$$P_s^{i+}(z) = P_s^{i+}(0)\, G^i(z)\,, \tag{3}$$

$$P_s^{i-}(z) = P_s^{i-}(\Delta L)\, G^i(\Delta L - z)\,, \tag{4}$$

$$P_{ASE}^{j+}(z) = P_{ASE}^{j+}(0)\, G^j(z) + 2hf^j \Delta f^j n_{sp}^j \left(G^j(z) - 1\right), \tag{5}$$

$$P_{ASE}^{j-}(z) = P_{ASE}^{j-}(\Delta L)\, G^j(\Delta L - z) - 2hf^j \Delta f^j n_{sp}^j \left(G^j(\Delta L - z) - 1\right), \tag{6}$$

where the gain can be expressed as

$$G^{i,j}(z) = \exp\left[\left(\Gamma^{i,j}\left\{\left(\sigma_e^{i,j} + \sigma_a^{i,j}\right) N_2(z) - \sigma_a^{i,j} N_g\right\} - \alpha^{i,j}\right) z\right] \tag{7}$$

The inversion factor n_{sp}^j (spontaneous emission factor) is defined by

$$n_{sp}^j = \frac{N_2}{N_2\left(1 + \frac{\sigma_a^j}{\sigma_e^j}\right) - N_g \frac{\sigma_a^j}{\sigma_e^j} - \frac{\alpha^j}{\Gamma^j \sigma_e^j}} \tag{8}$$

N_g :	Erbium ion density
α :	Attenuation constant

Assuming an exponential dependence of the gain, the local power of a section can be replaced by its average value

$$\left\langle P_s^{i\pm} \right\rangle = P_s^{i\pm}(0)\, \frac{G^i - 1}{\ln\left(G^i\right)}, \tag{9}$$

$$\left\langle P_{ASE}^{j+} \right\rangle = P_{ASE}^{j+}(0)\, \frac{G^j - 1}{\ln\left(G^j\right)} + 2hf^j \Delta f^j n_{sp}^j \left\{\frac{G^j - 1}{\ln\left(G^j\right)} - 1\right\}, \tag{10}$$

$$\left\langle P_{ASE}^{j-} \right\rangle = P_{ASE}^{j-}(\Delta L)\, \frac{G^j - 1}{\ln\left(G^j\right)} + 2hf^j \Delta f^j n_{sp}^j \left\{\frac{G^j - 1}{\ln\left(G^j\right)} - 1\right\}. \tag{11}$$

Introducing the saturation power

$$P_{sat}^{i,j} = \frac{hf^{i,j} \pi r_k^2}{\tau_{21} \Gamma^{i,j} \sigma_a^{i,j}} \tag{12}$$

τ_{12} :	Fluorescence lifetime
r_k :	Core radius

the ion density of the second energy level can be reformulated

$$N_2 = \frac{N_g \left(\sum_i \frac{\langle P_s^{i\pm} \rangle}{P_{sat}^i} + \sum_j \frac{\langle P_{ASE}^{j\pm} \rangle}{P_{sat}^j} \right)}{1 + \sum_i \frac{\langle P_s^{i\pm} \rangle}{P_{sat}^i} \left(\frac{\sigma_e^i}{\sigma_a^i} + 1 \right) + \sum_j \frac{\langle P_{ASE}^{j\pm} \rangle}{P_{sat}^j} \left(\frac{\sigma_e^j}{\sigma_a^j} + 1 \right)}. \tag{13}$$

For the overall gain in one section one obtains

$$G^{i,j} = \exp \left\{ \left(\frac{-N_g \Gamma^{i,j} \sigma_a^{i,j} \left(1 + P_b - \frac{\sigma_e^{i,j}}{\sigma_a^{i,j}} P_a \right)}{1 + P_a + P_b} - \alpha^{i,j} \right) \Delta L \right\}, \tag{14}$$

where the dummy power variables P_a and P_b are given by

$$P_a = \sum_i \frac{\langle P_s^{i\pm} \rangle}{P_{sat}^i} + \sum_j \frac{\langle P_{ASE}^{j\pm} \rangle}{P_{sat}^j}, \tag{15}$$

$$P_b = \sum_i \frac{\langle P_s^{i\pm} \rangle}{P_{sat}^i} \frac{\sigma_e^i}{\sigma_a^i} + \sum_i \frac{\langle P_{ASE}^{j\pm} \rangle}{P_{sat}^j} \frac{\sigma_e^j}{\sigma_a^j}. \tag{16}$$

The equations are solved iteratively for each section in both forward and backward direction until self-consistent results are obtained. This is done by propagating only the pump power first and determining the corresponding gain values. These values are used for the calculation of the signal, forward and backward ASE noise resulting in new gain values.
The calculations explained above are only used for the initialization of the EDFA component. If the gain and ASE spectra are determined, the EDFA is modeled as a gain element. Noise is described as filtered thermal light. The algorithm for generating ASE noise is the same as applied to the LED.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	cross12_file
Description	Absorption cross section file
Default Value	none
Parameter Range	valid file path

Parameter Name	cross21_file
Description	Emission cross section file
Default Value	none
Parameter Range	valid file path

Parameter Name	Tau
Description	Fluorescence lifetime
Default Value	0.01
Parameter Range	0 ≤ Tau ≤ 1e3
Unit	s

Parameter Name	Radius
Description	Core radius
Default Value	2
Parameter Range	1 ≤ Tau ≤ 10
Unit	µm

Parameter Name	Radius
Description	Local erbium density
Default Value	2000000
Parameter Range	0 ≤ Radius ≤ 1e100
Unit	$(\mu m)^{-3}$

Parameter Name	alpha
Description	Fiber background loss
Default Value	0.02
Parameter Range	0 < alpha < 1
Unit	dB/m

Parameter Name	Confinement
Description	Confinement factor
Default Value	0.3
Parameter Range	0 < Confinement < 1

Parameter Name	L
Description	Amplifier length
Default Value	0.02
Parameter Range	1 ≤ L ≤ 500
Unit	m

Parameter Name	p_pump_for
Description	Forward pumping power
Default Value	0.05
Parameter Range	0 ≤ p_pump_for ≤ 10
Unit	W

Parameter Name	lambda_for
Description	forward pump wavelength
Default Value	1.48
Parameter Range	1.4 < lambda_for < 1.8
Unit	µm

Parameter Name	p_pump_back
Description	Backward pump power
Default Value	0.05
Parameter Range	0 < p_pump_back < 10
Unit	W

Parameter Name	Pump_lamba_back
Description	Backward pump wavelength
Default Value	1.48
Parameter Range	1.4 < Pump_lamba_back < 1.8
Unit	µm

Parameter Name	wavelengths
Description	Number of wavelength slices (ASE)
Default Value	100
Parameter Range	1 < wavelengths

Parameter Name	lambda_min
Description	Minimum wavelength)
Default Value	1.5
Parameter Range	1.4 < lambda_min
Unit	µm

Parameter Name	lambda_max
Description	Maximum wavelength
Default Value	1.6
Parameter Range	lambda_max < 1.8
Unit	µm

Parameter Name	Sections
Description	Number of sections
Default Value	20
Parameter Range	1 ≤ Sections ≤ 10.000

Parameter Name	relative_error
Description	Relative error of the final iteration
Default Value	0.0001
Parameter Range	0 ≤ Sections ≤ 1e3

Input File Format

The files for the absorption and emission cross section are formatted as follows:

```
// cross_section definition file
diff_0 = 0.0
diff_n = 0.0
// (x_data   y_value)
1440e-9  0.5e-25
1445e-9  0.6e-25
1450e-9  0.75e-25
1455e-9  ...
```

$diff\_0$ and $diff\_n$ are the derivatives of the absorption and emission cross-sections at the boundaries of the calculation window and should be zero by default. The wavelength [m] and the cross section data [m^2] are listed in two columns, separated by blanks.

Graphs

The spectral EDFA contains the following Visualizers:

- The emission cross section can be shown in the wavelength domain.
- The absorption cross section can be plotted in the wavelength domain.
- The accumulated power gain spectrum after each section can be plotted in the wavelength domain.
- The accumulated forward noise power density spectrum after each section can be plotted in the wavelength domain.

- The accumulated backward noise power density spectrum before each section can be plotted in the wavelength domain.
- The power gain spectrum of each section can be plotted in the wavelength domain
- The spontaneous noise spectrum can be plotted in the wavelength domain

References

[1] T.G. Hodgkinson, “Improved average power analysis technique for erbium-doped fiber amplifiers,” IEEE Photon. Technol. Lett., vol. 4, pp. 1273-1275, 1992.

[2]] E. Lichtman, “Limitations imposed by polarization dependent gain and loss on all-optical ultralong communication systems,” J. Lightwave Technol., vol. 13, pp. 906-913, 1995.

7.4.3 Semiconductor Optical Amplifier (SOA) Fundamentals

In analogy to lasers, semiconductor optical amplifiers (SOAs) use the effect of stimulated emission for light amplification. The behavior of an SOA is described by the rate equation for electron carrier density [1], where the whole SOA is modeled as one lumped element

$$\frac{d\overline{N}(t)}{dt} = \frac{I}{qV_{act}} - \frac{\overline{N}(t)}{\tau_e} - R_s(t) - R_{ASE}(t) \tag{1}$$

$\overline{N}$:	Carrier density
I :	Pump current
Q :	Electron charge
L :	Length of active zone
w :	Width of active zone
d :	Thickness of active zone
τ_e :	Electron lifetime
R_S :	Recombination rate due to stimulated emission
R_{ASE} :	Recombination rate due to amplified spontaneous emission

where the volume of the active region can be expressed as

$$V_{act} = Lwd \tag{2}$$

The electron lifetime is defined by [1, 2]

$$\tau_e^{-1} = R_A + R_B\overline{N}(t) + R_C\overline{N}^2(t) \tag{3}$$

where R_A, R_B and R_C are the coefficients of non-radiative, spontaneous and Auger recombination, respectively. The recombination rate due to stimulated emission is given by

$$R_s(t) = \frac{\Gamma g_m(t) P_s^{in}(t)}{A_{akt}hf}\left(\frac{\exp(g(t)L) - 1}{g(t)L}\right) \tag{4}$$

P_s^{in} :	(Filtered) signal power
h :	Planck constant
f :	Signal frequency
Γ :	Mode confinement factor
g :	Net gain
g_m :	Material gain
L :	Amplifier length

$$A_{act} = wd \tag{5}$$

denotes the cross section area of the active region. The recombination rate due to amplified spontaneous emission and spontaneous emission may be written as

$$R_{ASE}(t) = 2R_{sp}(t)\frac{\Gamma g_m(t)}{g(t)}\left(\frac{\exp(g(t)L) - 1}{g(t)L} - 1\right) \tag{6}$$

$$R_{sp}(t) = \beta_{sp}R_B\overline{N}^2(t) \tag{7}$$

β_{sp} :	spontaneous emission factor

respectively.
Here, the material gain is

$$g_m(t) = a_G\left(\overline{N}(t) - \overline{N}_t\right) - k_G(f - f_P(t))^2 \tag{8}$$

a_G :	Material gain coefficient
k_G :	Spectral gain curvature
$\overline{N}_t$:	Electron density at transparency
f_p :	Gain peak frequency

where the frequency of the gain maximum can be obtained by

$$f_{peak}(t) = f_0 - \frac{df_{peak}}{d\overline{N}}\left(\overline{N}(t) - \overline{N}_r\right) \tag{9}$$

The net gain $G(t)$ takes into account the material gain, nonlinear gain compression and internal loss

$$g(t) = \frac{\Gamma g_m(t)}{1 + \varepsilon_{nl}\Gamma\langle P(t)\rangle / A_{act}} - \alpha_i \tag{10}$$

α_i :	Internal loss
ϵ_{nl} :	Spectral gain curvature

Assuming an exponential dependence of the gain on the amplifier length, the average power may be written as

$$\langle P_s(t)\rangle = P_s^{in}(t)\left(\frac{\exp(g(t)L) - 1}{g(t)L}\right) \tag{11}$$

$$\langle P_{ASE}(t)\rangle = \frac{2hf_{ASE}A_{akt}R_{sp}(t)}{g(t)}\left(\frac{\exp(g(t)L) - 1}{g(t)L} - 1\right) \tag{12}$$

f_{ASE} :	ASE center frequency

Since the amplifier is not able to follow an input signal arbitrarily fast, an effective SOA bandwidth is introduced to describe the transient behavior of the SOA correctly. In the simulation, the bandwidth limitation is modeled by filtering the signal power with an internal raised cosine filter or an arbitrary external filter.
The phase difference between input and output signals at any time instant is finally calculated using the relation

$$\Delta\varphi_s = -\frac{2\pi fL}{c_0}\left(n_{eff} + \frac{dn_{eff}}{d\overline{N}}\left(\overline{N} - \overline{N}_t\right)\right) \tag{13}$$

n_{eff} :	Group index at transparency
$\frac{n_{eff}}{dN}$:	Derivation of group index with respect to carrier density

For a proper operation, multi-channel signals have to be used as frequency-separated signals at the SOA input and output. By specifying numIn and f0i, the number of channels applied to the SOA and the corresponding center frequencies can be chosen.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	L
Description	Length of active region
Default Value	1e-4
Parameter Range	1e-6 ≤ L ≤ 1e-3
Unit	m

Parameter Name	w
Description	Width of active region
Default Value	1.5e-6
Parameter Range	1e-9 ≤ w ≤ 1e-3
Unit	m

Parameter Name	d
Description	Thickness of active region
Default Value	1.5e-7
Parameter Range	1e-9 ≤ d ≤ 1e-3
Unit	m

Parameter Name	I
Description	TPump current
Default Value	0.05
Parameter Range	0 < I < 1
Unit	A

Parameter Name	a
Description	Gain coefficient
Default Value	1.84e-20
Parameter Range	1e-30 ≤ a ≤ 1e-10
Unit	m^2

Parameter Name	N0
Description	Electron density at transparency
Default Value	1.6e24
Parameter Range	1e10 ≤ N0 ≤ 1e40
Unit	m^{-3}

Parameter Name	fp0
Description	Reference gain peal frequency
Default Value	197.352194
Parameter Range	inside the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	Nr
Description	Reference carrier density
Default Value	3e24
Parameter Range	1e10 ≤ Nr ≤ 1e40
Unit	m^{-3}

Parameter Name	dfp_dN
Description	Peak gain frequency derivative with respect to carrier density
Default Value	3.374e-24
Parameter Range	1e-30 ≤ dfp_dN ≤ 1e-10
Unit	m^{3}THz

Parameter Name	k
Description	Curvature of spectral gain
Default Value	106.593
Parameter Range	0 < k < 1e3
Unit	THz^2/m

Parameter Name	alpha
Description	Internal waveguide loss (α_{int})
Default Value	1500
Parameter Range	0 ≤ alpha ≤ 1e6
Unit	m^{-1}

Parameter Name	gamma
Description	Mode confinement factor (Γ)
Default Value	0.31
Parameter Range	0 < gamma < 1

Parameter Name	e_nl
Description	Nonlinear gain compression factor
Default Value	1e-15
Parameter Range	0 < e_nl < 1
Unit	1/W

Parameter Name	beta
Description	Spontaneous emission factor (β_{sp})
Default Value	5e-5
Parameter Range	0 < beta < 1
Unit	Ws/m

Parameter Name	ng
Description	Group index of active region
Default Value	3.1
Parameter Range	1 < ng

Parameter Name	dng_dN
Description	Derivative of group index with respect to carrier density
Default Value	6.0e-26
Parameter Range	0 < dng_dN < 1
Unit	m^3

Parameter Name	NumIn
Description	Number of input channels
Default Value	1
Parameter Range	1, 2

Parameter Name	soaBandwidth
Description	Effective FWHM-bandwidth
Default Value	200
Parameter Range	0 ≤ soaBandwidth ≤ Frequency Range of simulation
Unit	GHz

Parameter Name	Filter roll-off factor
Description	Internal filter roll-off factor
Default Value	0.8
Parameter Range	0 ≤ Filter roll-off factor ≤ 1

Parameter Name	ASE
Description	Include ASE effects?
Default Value	true
Parameter Range	boolean

References

[1] L. Gillner: “Properties of optical switching networks with passive or active space switches”, IEEE Proceeding J, Vol. 140, No. 5, 1993

[2] T. Durhuus, B. Mikkelsen, K. E. Stubkjaer: “Detailed Dynamic Model for Semiconductor Optical Amplifiers and Their Crosstalk and Intermodulation Distortion”, IEEE Journal of Lightwave Technology, Vol. 10, No.8, 1992

[3] S. Mottet, J. L. Pleumeekers, T. Mercier: “ Simulation of Semiconductor Optical Amplifiers”, France Telecom, TU Delft, Proceedings of the 7th European Conference on Integrated Optics, ECIO, 1995

[4] J. Lenge: “Modellierung optischer Netze mit Wellenlängenvermittlung”, Diploma thesis, Chair of High Frequency Technique, University of Dortmund, 1997

7.5 Filters

7.5.1 Acousto-Optic Tunable Filter

Fundamentals

The acousto-optic filter is based on the same functional principle as the electro-optic filter (c.f. section 7.5.4 on page 222). In contrast to its electro-optic counterpart, the acousto-optical filter uses surface acoustic waves for generating a dynamic grating structure in the mode converter. An advantage of the acousto-optic filter is that acoustic waves influence each other only weakly. Thus, several wavelengths can be selected simultaneously.
The device behavior is described using a sum of power transfer functions of the type

$$H(f) = \frac{\sin^2(\pi \Delta n L (f - f_0)/c_0)}{(\pi \Delta n L (f - f_0)/c_0)^2} \tag{1}$$

with different center frequencies.

Δn :	Refractive index perturbation
L :	Length of perturbation region
f_0 :	Center frequency
c_0 :	Velocity of light

The resulting transfer function is properly normalized to avoid unphysical gain or loss. For simplicity, the AOTF is characterized by its center frequency and the FWHM bandwidth only.
An additional attenuation constant has been introduced describing insertion and internal loss. For usage as OADM filter not only the transfer function eq. (1), but also the complementary function

$$\overline{H}(f) = 1 - H(f) \tag{2}$$

is available at the device output.
White Gaussian noise can be added at the filter input and will appear spectrally shaped by the transfer function at the filter output. The noise can be treated numerically by randomly generated noise samples. Alternatively, the noise variance is calculated analytically at the filter output and can be used for further processing.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field
Output	2	Optical field (Complementary output)

Component Parameters

Parameter Name	FWHM
Description	FWHM bandwidth (Δf_{FWHM})
Default Value	20
Parameter Range	0 < FWHM < Frequency Range
Unit	GHz

Parameter Name	N_Ch
Description	Number of channels
Default Value	3
Parameter Range	$1 < N\_Ch$
Unit	GHz

Parameter Name	f_i
Description	Center frequency of Channel $i + 1$
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

References

[1] D. J. G. Mestdagh: Fundamentals of multi-access optical fiber networks, Artech House, Inc., Norwood/USA, 1995

[2]] Y. Yariv, P. Yeh: Optical waves in crystals, Wiley, NY, 1984

7.5.2 Analytical Filter Fundamentals

PHOTOSS supports various kinds of ideal filters which can be used for optical as well as for electrical signals. In this section, a short description of all implemented filters is given. The characterization is based on the FWHM bandwidth of the filter, the center frequency and the filter loss. The FWHM bandwidth is defined as the double-sided bandwidth. The FWHM-bandwidth is related to the filter power transfer function.
White Gaussian noise can be added at the filter input and will appear spectrally shaped by the transfer function at the filter output. The noise can be treated numerically by randomly generated noise samples. Alternatively, the noise variance is calculated analytically at the filter output and can be used for further processing. Several filters require additional parameters, e.g. the filter order.
If the output port number is set to two, the second output port will provide the signal modified by the inverse transfer function.

Rectangle Filter

The transfer function of an ideal rectangle filter is described by

$$H(f) = \begin{cases} A \, |f - f_0| \leq \Delta f_{FWHM}/2 \\ 0 \, |f - f_0| > \Delta f_{FWHM}/2 \end{cases} \tag{1}$$

where Δf_{FWHM} is the full width at half maximum bandwidth of the filter. A accounts for the filter loss.

Gaussian Filter

The transfer function of a Gaussian filter yields

$$H(f) = \exp\left(-\left(\frac{f - f_0}{\sigma}\right)^2\right) \tag{2}$$

where Δf_0 is the center frequency and σ^2 the variance of the filter. Expressed in terms of the 3dB-bandwidth Δf_{FWHM} and multiplied with A this function yields

$$H(f) = A \cdot \exp\left(-2\ln(2)\left(\frac{f - f_0}{\Delta f_{FWHM}}\right)^2\right). \tag{3}$$

Gaussian filters of order N are given by

$$H(f) = A \cdot \exp\left(-2\ln(2)\left(\frac{f_0 - f}{\frac{\Delta f_{FWHM}}{\sqrt[N]{2^{N-1}}}}\right)^{2N}\right) \tag{4}$$

Butterworth Filter

The transfer function of a Butterworth filter of order N is given by

$$H(f) = A \cdot \frac{(\Delta f_{FWHM}/2)^N}{\prod\limits_{k=0}^{N-1} (j(f - f_0) - p_k)} \tag{5}$$

where Δf_{FWHM} is the 3dB bandwidth and p_k are the poles of the filter, which can be expressed as

$$p_k = \frac{\Delta f_{FWHM}}{2} \cdot e^{j\frac{\pi}{2}\left(1+\frac{2k+1}{N}\right)} \tag{6}$$

Chebyshev Filter

The transfer function of a Chebychev filter of order N can be described as

$$H(s) = -A \cdot \frac{\prod_{k=0}^{N-1} s_k}{\prod_{k=0}^{N-1} (s - s_k)} \tag{7}$$

with

$$s = j(f - f_0). \tag{8}$$

Here, s_k are the poles of the filter

$$s_k = \frac{\Delta f_{FWHM}}{2} \cdot (\sinh\alpha \cdot \cos\beta_k + j \cdot \cosh\alpha \cdot \sin\beta_k) \tag{9}$$

where

$$\alpha = \frac{1}{N} \operatorname{arc\,sinh}\left(p^{-1}\right) \tag{10}$$

and

$$\beta_k = \frac{\pi(2(k+1) + N - 1)}{2N}. \tag{11}$$

Bessel Filter

The transfer function of a Bessel filter of order N can be expressed in terms of the N-th order Bessel polynomial

$$H(s) = A \cdot \frac{d_0}{B_N(s)}, \tag{12}$$

where

$$d_0 = \frac{(2N)!}{2^n \cdot N!} \tag{13}$$

a normalizing constant and $B_N(s)$ is the Bessel polynomial of the form

$$B_N(s) = \sum_{k=0}^{N} d_k s^k \tag{14}$$

with

$$d_k = \frac{(2N-k)!}{2^{N-k} \cdot k!\,(N-k)!} \tag{15}$$

and

$$s = j\left(2\frac{(f - f_0) \cdot w_b}{\Delta f_{FWHM}}\right) \tag{16}$$

w_b denotes the normalized 3dB bandwidth and can be approximated by

$$w_b \approx \sqrt{(2N-1) \cdot \ln 2}. \tag{17}$$

for $N \geq 3$.

RC Filter

The transfer function of an RC filter is expressed as

$$H(f) = A \cdot \frac{1}{1 + j2\frac{f-f_0}{\Delta f_{FWHM}}} \tag{18}$$

Raised Cosine Filter

The transfer function of a raised cosine filter is described by

$$H(f) = \begin{cases} A & |f - f_0| < \frac{(1-\alpha)}{2}\Delta f \\ A \cdot \cos^2\left[\frac{\pi}{2\alpha\Delta f}\left(|f - f_0| - \frac{(1-\alpha)}{2}\Delta f\right)\right] & \frac{(1-\alpha)}{2}\Delta f \leq |f - f_0| < \frac{(1+\alpha)}{2}\Delta f \\ 0 & \frac{(1+\alpha)}{2}\Delta f \leq |f - f_0| \end{cases} \tag{19}$$

where

$$\Delta f = \Delta f_{FWHM} \cdot \frac{1}{1 - \alpha + 4/\pi \cdot \alpha \cdot \arccos \sqrt[4]{2}} \tag{20}$$

Cosine roll-off Filter

A filter, which can be used in the spectral power domain for windowing and exhibits cosine shaped edges, may be expressed as

$$H(f) = \begin{cases} A & |f - f_0| < f_1 \\ \sqrt{\frac{1}{2} \cdot A^2 \cdot \left[1 + \cos\left(\frac{|f-f_0|-f_1}{\alpha \cdot \Delta f_{FWHM}} \cdot \pi\right)\right]} & f_1 \leq |f - f_0| < f_2 \\ 0 & f_2 \leq |f - f_0| \end{cases} \tag{21}$$

where Δf_{FWHM} is the 3dB-bandwidth and α denotes the roll-off factor with

$$0 \leq \alpha \leq 1 \tag{22}$$

f_1 and f_2 are calculated by

$$f_1 = \frac{1-\alpha}{2}\Delta f_{FWHM} \tag{23}$$

and

$$f_2 = \frac{1+\alpha}{2}\Delta f_{FWHM} \tag{24}$$

respectively.

Squared Cosine roll-off Filter

A filter with cosine shaped edges in the frequency amplitude domain can be designed using the transfer function

$$H(f) = \begin{cases} A & |f - f_0| < f_1 \\ \frac{1}{2} \cdot A \cdot \left[1 + \cos\left(\frac{|f-f_0|-f_1}{\alpha \cdot \Delta f} \cdot \pi\right)\right] & f_1 \leq |f - f_0| < f_2 \\ 0 & f_2 \leq |f - f_0| \end{cases} \tag{25}$$

where α is the roll-off factor again. Here, Δf is related to the FWHM bandwidth by

$$\Delta f = \frac{\Delta f_{FWHM}}{1 + \left[\frac{2}{\pi} \cdot \arccos\left(\sqrt{2} - 1\right) - 1\right] \cdot \alpha} \tag{26}$$

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	N_{out}	Optical field

Component Parameters

Parameter Name	Mode
Description	Filter mode
Default Value	electrical
Parameter Range	electrical, optical

Parameter Name	Type
Description	Filter type
Default Value	Gauss
Parameter Range	Rectangle, Gauss, Butterworth, Chebyshev, Bessel, RC, Raised cosine, Cosine-roll off, Squared-cosine-roll off, Integrator

Parameter Name	Alpha
Description	Filter attenuation
Default Value	0
Parameter Range	$0 \leq$ Alpha
Unit	dB

Parameter Name	f0
Description	Center Frequency
Default Value Electrical	0
Default Value Optical	193.1
Parameter Range	within the PHOTOSS simulation bandwidth or the base band
Unit	THz

Parameter Name	del_f
Description	FWHM bandwidth
Default Value	100
Parameter Range	0 < del_f Frequency Range
Unit	GHz

Parameter Name	N
Description	Noise power spectral density
Default Value	0
Parameter Range	$0 \leq N$
Unit	W/Hz

Parameter Name	Order
Description	Order of the transfer function
Default Value	depending on chosen filter type
Parameter Range	0 < Order

Parameter Name	OutPorts
Description	Number of output ports
Default Value	1
Parameter Range	1 ≤ OutPorts

Graphs

The complex amplitude transfer function can be visualized in the frequency domain

References

none

7.5.3 APF Structure

Fundamentals

The all-pass filter (APF) structure shown below is composed of N feedback loops (stages). At each stage, a part of the signal power is coupled back to the input through the respective loop and interferes with the subsequent input signal, resulting in an IIR type filter characteristic.

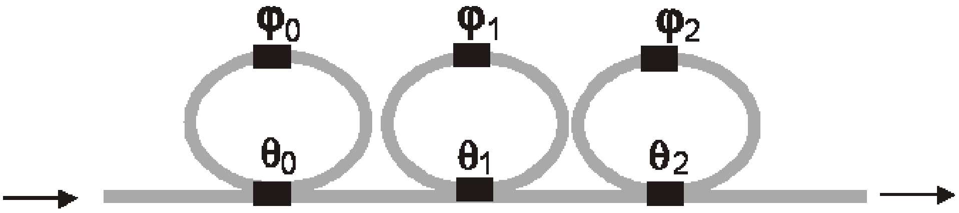


Figure 7.5.1. 3rd order APF structure

The transfer function $H_i(f)$ of a single stage i is given by

$$H_i(f) = \frac{e^{j2\pi(f-f_0)T-j\varphi_i} + e^{-j\theta_i} - 1}{(e^{-j\theta_i} - 1) \cdot e^{j2\pi(f-f_0)T-j\varphi_i} - 1} \tag{1}$$

with frequency f and center frequency f_0. While the feedback delay T is identical for all stages, the feedback phase shift φ_i can be used to tune the length of the feedback loop. The coupler phase shift θ_i controls how much signal power is coupled into the feedback loop. The corresponding coupler coefficient k_i follows

$$k_i = -10 \cdot \log_{10}\left(1 - \cos^2\frac{\theta_i}{2}\right) \tag{2}$$

In case of a given dispersion that shall be established in a known frequency range, one might use the auto-optimization feature. It needs the amount of stages for the filter, the desired dispersion and the bandwith as well as guessed initial values for the φs and θs. Care must be taken when checking the results because the target function might have a lot of local minima in which the optimization algorithm might come to a halt. The implemented method is a Levenberg-Marquardt algorithm (see [1]), that is a Gauss-Newton type non-linear least square solver.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	T
Description	Time delay
Default Value	10
Parameter Range	T < Block Time
Unit	ps

Parameter Name	FSR
Description	Free spectral Range ($FSR = 1/T$)
Default Value	100
Parameter Range	FSR < Frequency Range
Unit	GHz

Parameter Name	Stages
Description	Number of stages (filter order)
Default Value	2
Parameter Range	$1 \leq$ Stages

Parameter Name	f0
Description	Center frequency
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	Auto-optimize phi and theta
Description	Turns auto-optimization of angles on and off
Default Value	false
Parameter Range	boolean

Parameter Name	Low Frequency Bound
Description	Low bound of the auto-optimization
Default Value	193.0875
Parameter Range	up to the center frequency
Unit	THz

Parameter Name	High Frequency Bound
Description	High bound of the auto-optimization
Default Value	193.1125
Parameter Range	down to the center frequency
Unit	THz

Parameter Name	Step Width
Description	Step width in which the transfer function will be evaluated
Default Value	2.5e-5
Parameter Range	up to the difference of low and high frequency bound
Unit	THz

Parameter Name	Dispersion
Description	Target dispersion of the APF
Default Value	1000
Parameter Range	-10e5...10e5
Unit	ps/nm

Parameter Name	theta_i
Description	coupler phase shift in stage i (θ_i)
Default Value	$\pi/2$
Parameter Range	not limited
Unit	rad

Parameter Name	k_i
Description	coupler coefficient in stage i
Default Value	3
Parameter Range	not limited
Unit	dB

Parameter Name	phi_i
Description	feedback phase shift in stage i (φ_i)
Default Value	$\pi/2$
Parameter Range	not limited
Unit	rad

References

[1] K. Madsen, H.B. Nielsen, O. Tingleff: Methods for non-linear least squares problems, 2nd Edition, 2004

7.5.4 Electro-Optic Tunable Filter

Fundamentals

The electro-optic tunable filter is a device based on the electro-optic effect. A multi-channel input signal with arbitrary polarization is split into two orthogonal polarization states by a polarization splitter. In a TE/TM-mode converter a periodic perturbation is created by an external voltage due to the electro-optic effect. This grating structure leads to a frequency-dependent change of the polarization state of the TE and TM input signals, respectively. Only the channel satisfying the phase-matching condition will be converted. Thus, by a second polarization splitter the desired channel can be separated from the multi-channel signal.
To a first order, the power transfer function of the device can be described by

$$H(f) = \frac{\sin^2\left(\pi \Delta n L (f - f_0)/c_0\right)}{\left(\pi \Delta n L (f - f_0)/c_0\right)^2} \tag{1}$$

Δn :	Refractive index perturbation
L :	Length of perturbation region
f_0 :	Center frequency
c_0 :	Velocity of light

The FWHM bandwidth is approximately given by

$$\frac{f_0}{\Delta f_{FWHM}} \approx \frac{L}{\Lambda} \tag{2}$$

Δf_{FWHM} :	Effective filter bandwidth
Λ :	Period of electro-optically induced grating

and seen to depend on the period of the induced grating as well as the channel carrier frequency and the effective length of the mode converter.
For simplicity, the EOTF is characterized here not by its physical parameters L, Δn, Λ , but by its center frequency and the FWHM bandwidth.
An additional attenuation constant has been introduced describing insertion and internal loss. For usage as OADM filter not only the transfer function eq. (1), but also the complementary function

$$\overline{H}(f) = 1 - H(f) \tag{3}$$

is available at the device output.
White Gaussian noise can be added at the filter input and will appear spectrally shaped by the transfer function at the filter output. The noise can be treated numerically by randomly generated noise samples. Alternatively, the noise variance is calculated analytically at the filter output and can be used for further processing.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field
Output	2	Optical field (Complementary output)

Component Parameters

Parameter Name	f_0
Description	Center frequency
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	FWHM
Description	FWHM bandwidth (Δf_{FWHM})
Default Value	100
Parameter Range	0 < FWHM < Frequency Range
Unit	GHz

References

[1] D. J. G. Mestdagh: Fundamentals of multi-access optical fiber networks, Artech House, Inc., Norwood/USA, 1995

[2]] Y. Yariv, P. Yeh: Optical waves in crystals, Wiley, NY, 1984

7.5.5 Fabry-Perot Filter

Fundamentals

A Fabry-Perot filter (FPF) represents a tunable optical bandpass filter. It consists of two partial reflective mirrors spaced within a certain distance forming an optical resonator.
The reflected waves in the resonator interfere constructively, if the round-trip phase equals a multiple of 2π. Otherwise, destructive interference is obtained. The filter transfer function is periodic in the frequency domain.
To tune the filter from one channel to another, the resonant condition of the FPF must be changed. This can be done by adjusting the Fabry-Perot cavity length by a piezoelectric translator.
The FP power transfer function is given by

$$T_{FPF}(f) = C\frac{A(1-R)^2}{(1-AR)^2 + 4AR\sin^2\left(\frac{\pi f}{FSR}\right)} \tag{1}$$

A :	Internal loss
C :	Insertion and additional losses
R :	Reflectivity of the facets
FSR :	Free spectral range

The period of the transfer function is determined by the FSR

$$FSR = \frac{c_0}{2nL} \tag{2}$$

c_0	I Velocity of light
n :	Effective refractive index
L :	Cavity length

The 3dB bandwidth Δf_{FWHM} of the FPF is obtained from (1) and is given by

$$\Delta f_{FWHM} = \frac{c_0}{2nL}\frac{1-AR}{\pi\sqrt{AR}} \tag{3}$$

The ratio of the free spectral range to Δf_{FWHM} is known as the finesse F of the Fabry-Perot filter

$$F = \frac{FSR}{\Delta f_{FWHM}} = \frac{\pi\sqrt{AR}}{1-AR} \tag{4}$$

Typically, F ranges from 20 to 100.
Instead of characterizing the FPF by its FSR, the reflectivity R and the attenuation A, the filter center frequency f_0 , the 3dB bandwidth Δf_{FWHM} and the minimum FSR can be defined as alternative parameters. In this case, the following relations hold [3]:

$$FSR(f_0) = \frac{f_0}{\left\lfloor\frac{f_0}{FSR_{\min}}\right\rfloor}, \tag{5}$$

$$R(f_0, \Delta f_{FWHM}) = \frac{1}{2A}\left[2+\left(\frac{\pi\Delta f_{FWHM}}{FSR(f_0)}\right)^2\right] - \sqrt{\frac{1}{4A^2}\left[2+\left(\frac{\pi\Delta f_{FWHM}}{FSR(f_0)}\right)^2\right]^2 - \frac{1}{A^2}}, \tag{6}$$

where $\lfloor ... \rfloor$ yields the integer part of its argument.
To simulate e.g. an optical pre-amplifier with an FPF-limited bandwidth, noise with user-defined power spectral density (PSD) can be added at the filter input representing ASE. White Gaussian noise will be

added at the filter input and will appear spectrally shaped by the transfer function at the filter output. The noise can be treated numerically by randomly generated noise samples. Alternatively, the noise-variance is calculated analytically at the filter output and can be used for further processing.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	f_0
Description	Center frequency
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	FWHM
Description	FWHM bandwidth (Δf_{FWHM})
Default Value	10
Parameter Range	0 < FWHM < Frequency Range
Unit	GHz

Parameter Name	FSR
Description	Free spectral range
Default Value	99.9741
Parameter Range	0 < FSR < Frequency Range
Unit	GHz

Parameter Name	facet_ref
Description	Facet reflectivity
Default Value	0.7313
Parameter Range	0 < facet_ref < 1
Unit	dB

Parameter Name	n*L
Description	Refractive index times cavity length
Default Value	1.499
Parameter Range	0 < n*L
Unit	mm

Parameter Name	Finesse
Description	Finesse
Default Value	9.997
Parameter Range	0 < Finesse

References

[1] J.Kastner: “Übertragungsverzerrungen durch Filter und Multiplexern in optischen WDM-Systemen”, Diploma thesis, Chair of High Frequency Technique, University of Dortmund, 1996

[2] D. J. G. Mestdagh: “Fundamentals of multi-access optical fiber networks”, Artech House, Inc., Norwood/USA, 1995

[3] J. Lenge: “Modellierung optischer Netze mit Wellenlängenvermittlung”, Diploma thesis, Chair of High Frequency Technique, University of Dortmund, 1997

7.5.6 Fiber Bragg Grating Fundamentals

Fiber Bragg Gratings are photosensitive fibers in which a grating structure is written by UV exposure. They can be used as optical filters in WDM systems, in fiber grating lasers, for gain flattening of EDFAs, or for dispersion compensation. The device behavior is modeled using the theory of coupled modes and the transfer matrix method.
The modulation of the effective refractive index Δn_{eff} of the fiber is given by

$$\Delta n_{eff}(z) = Y(z)\,\Delta n_{eff,dc} + Y(z)\Delta n_{eff,ac}\cos\left(\frac{2\pi}{\Lambda}z + \Phi_{FG}(z)\right) \tag{1}$$

$\Delta n_{eff,dc}$:	Refractive index perturbation
$\Delta n_{eff,ac}$:	Length of perturbation region
Y :	Apodizing function
Λ :	Grating period
Φ_{FG} :	Chirp of the grating

The apodizing function can be chosen to have either raised cosine

$$Y(z) = \frac{1}{2}\left[1 + \cos\left(\frac{\pi z}{FHWM}\right)\right] \tag{2}$$

or Gaussian shape

$$Y(z) = \exp\left[-\frac{4\ln 2z^2}{FHWM}\right]. \tag{3}$$

$FWHM$:	FWHM bandwidth of apodizing function

The coupled differential equations of the complex envelopes A^+ and A^- of the forward and backward propagating waves are

$$\frac{dA^+(z)}{dz} = j\kappa_{dc}A^+(z) + j\kappa_{ac}A^-(z), \tag{4}$$

$$\frac{dA^-(z)}{dz} = -j\kappa_{dc}A^-(z) - j\kappa_{ac}^*A^+(z), \tag{5}$$

where the coupling coefficient κ_{dc} is defined by

$$\kappa_{dc} = \frac{2\pi n_{eff}}{c_0}(f - f_B) + \frac{2\pi f}{c_0}\Delta n_{eff,dc} - \frac{1}{2}\frac{d\Phi_{FG}}{dz} \tag{6}$$

f_B :	Bragg frequency
n_{eff} :	Effective refraction index
c_0 :	Velocity of light

The first term denotes the deviation of the frequency from the Bragg frequency

$$f_B = \frac{c_0}{2n_{eff}\Lambda} \tag{7}$$

the second term describes the influence of the refractive index variation (dc part). $1/2 d\Phi_{FG}/dz$ accounts for the chirp of the grating i.e. the variation of the grating period with the grating length. Linear chirp yields

$$\frac{1}{2}\frac{d\Phi_{FG}}{dz} = \frac{4\pi n_{eff} f_B^2 z}{c_0^2}\frac{d\lambda_B}{dz} \tag{8}$$

including chirp parameter $d\lambda_B/dz$, that describes the local modification of bragg wavelength λ_b.

$$\kappa_{ac} = \frac{\pi f}{c_0}\Delta n_{eff,ac} \tag{9}$$

is the coupling coefficient between the forward and backward travelling wave.
A grating of length L showing a spatial variation of the grating period and the refractive index is divided into segments ΔL of equal length. In each segment, the coupling coefficients are assumed to be constant. For a proper description, ΔL should include several grating periods. The transfer matrix method yields for one segment

$$\left(\begin{array}{c} A_i^+ \\ A_i^- \end{array}\right) = \mathrm{T}_i^B \left(\begin{array}{c} A_{i-1}^+ \\ A_{i-1}^- \end{array}\right) \tag{10}$$

where the transfer matrix is given by

$$\mathrm{T}_i^B = \left(\begin{array}{cc} \cosh\left(\kappa_B \Delta L\right) - j\frac{\kappa_{dc}}{\kappa_B}\sinh\left(\kappa_B \Delta L\right) & -j\frac{\kappa_{ac}}{\gamma_B}\sinh\left(\kappa_B \Delta L\right) \\ j\frac{\kappa_{ac}}{\kappa_B}\sinh\left(\kappa_B \Delta L\right) & \cosh\left(\kappa_B \Delta L\right) + j\frac{\kappa_{dc}}{\kappa_B}\sinh\left(\kappa_B \Delta L\right) \end{array}\right) \tag{11}$$

and

$$\kappa_B = \sqrt{\kappa_{ac}^2 - \kappa_{dc}^2}. \tag{12}$$

Multiplying the transfer functions of the different segments results in

$$\left(\begin{array}{c} A^+(0) \\ A^-(0) \end{array}\right) = \mathrm{T}_{ges}^B \left(\begin{array}{c} A^+(L) \\ A^-(L) \end{array}\right), \mathrm{T}_{ges}^B = \prod_{i=M}^{1} \mathrm{T}_i^B \tag{13}$$

Calculating the transfer function for transmission and reflection is straightforward and yields

$$H_{trans}(f) = \frac{A^-(0)}{A^+(0)} = \frac{1}{\mathrm{T}_{ges}^B(1,1)} \tag{14}$$

and

$$H_{refl}(f) = \frac{A^-(0)}{A^+(0)} = \frac{\mathrm{T}_{ges}^B(2,1)}{\mathrm{T}_{ges}^B(1,1)} \tag{15}$$

respectively. Attenuation can be additionally included.
By specifying a noise spectral density, white noise can be added at the FBG input, representing ASE. Hence, a gain flattened EDFA may be simulated using the FBG component.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field (transmission)
Output	2	Optical field (reflection)

Component Parameters

Parameter Name	f0
Description	Design frequency
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	alpha additional
Description	Additional loss
Default Value	0
Parameter Range	0 ≤ alpha additional
Unit	dB

Parameter Name	neff
Description	Effective refractive index (Δn_{eff})
Default Value	1.45
Parameter Range	1 ≤ neff

Parameter Name	del neff dc
Description	Refractive index modulation, dc part
Default Value	0
Parameter Range	0 ≤ del neff dc ≤ 1

Parameter Name	del neff ac
Description	Refractive index modulation, ac part
Default Value	0.0008
Parameter Range	0 ≤ del neff dc ≤ 1

Parameter Name	alpha additional
Description	Additional loss
Default Value	0
Parameter Range	0 ≤ alpha additional
Unit	dB

Parameter Name	Chirp
Description	Include chirp?
Default Value	false
Parameter Range	boolean

Parameter Name	Y
Description	Apodizing function
Default Value	Raised Cosine
Parameter Range	none, Raised Cosine, Gaussian,

Parameter Name	FWHM
Description	FWHM width of apodizing function
Default Value	1e8
Parameter Range	0 < FWHM < 1e12
Unit	nm

Parameter Name	numWL
Description	Number of wavelengths for calculation
Default Value	300
Parameter Range	1 < numWL

Parameter Name	numSect
Description	Number of sections for calculation
Default Value	101
Parameter Range	1 < numSect

Parameter Name	ipol
Description	Spline interpolation
Default Value	Linear
Parameter Range	Linear, Spline, Steps

References

none

7.5.7 Lab Filter

Fundamentals

The lab filter may be used e. g. to resemble measured transmission characteristics. It supports arbitrary transfer functions which are read from input data files (either in amplitude or power domain with linear or logarithmic scale). Phase information can be added with a separate data file. The transfer function is built by linear or cubic spline interpolation of the values contained in the input files. The data values are assumed to be centered around a frequency f_c and will be automatically shifted to the desired center frequency f_0.
Keep in mind that the wavelengths / frequencies in the input file must be in **ascending** order.

Input / Output

	Number	Type of signal
Input	1	Optical field / electrical signal
Output	1	Optical field / electrical signal

Component Parameters

Parameter Name	f0
Description	Center frequency
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	attenuation
Description	filter attenuation
Default Value	5
Parameter Range	-50 ≤attenuation ≤ +50
Unit	dB

Parameter Name	mode
Description	filter mode
Default Value	electrical
Parameter Range	electrical, optical

Parameter Name	dataAxis
Description	x Axis assignment
Default Value	Frequency
Parameter Range	Frequency, Wavelength

Parameter Name	dataType
Description	data Type
Default Value	dB
Parameter Range	dB, Amplitude, Power

Parameter Name	absfile
Description	amplitude response data file
Default Value	none
Parameter Range	valid file path

Parameter Name	usePhase
Description	use phase information
Default Value	false
Parameter Range	boolean

Parameter Name	argfile
Description	phase response data file
Default Value	none
Parameter Range	valid file path

Parameter Name	interMeth
Description	interpolation method
Default Value	Linear
Parameter Range	Linear, Cubic spline , Steps

Parameter Name	fc
Description	Data file carrier frequency
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Input file format

```
// Lab Filter data file
diff_0 = 0.0
diff_n = 0.0

// wavelength [micrometer]  transmission [dB]
1.440   0
1.441   1
...
```

diff_0 and diff_n are the derivatives of the transfer function at the first and last data value (zero by default). The x and y values are listed in two columns which are separated by blanks.

References

none

7.5.8 Mach-Zehnder Filter

Fundamentals

The Mach-Zehnder filter (MZF) is a tunable filter based on the interferometric principle. It consists of two 3dB couplers, which are connected by two waveguides.
The input signal is split into two parts of equal intensity in the first 3dB coupler. Then both optical waves propagate through waveguides of different optical lengths. Thus, a time delay between the two waves is introduced. At the output the waves are combined by the second 3dB coupler.
The scattering matrix of the 3dB coupler is described by

$$[H_{3dB}] = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -j \\ -j & 1 \end{bmatrix} \tag{1}$$

the scattering matrix associated with the waveguide branches is simply given by

$$[H_\tau] = \begin{bmatrix} 1 & 0 \\ 0 & e^{-j\frac{2\pi f}{FSR}} \end{bmatrix} \tag{2}$$

Here the relation

$$FSR = \frac{1}{\Delta\tau} \tag{3}$$

FSR :	Free spectral range
$\Delta\tau$:	Time delay difference

holds.
The overall transfer function $H_{MZF}(f)$ of an ideal single-stage Mach-Zehnder filter is obtained by multiplying the scattering matrices of the different components yielding

$$[H_{MZF}(f)] = [H_{3dB}] \, . \, [H_\tau] \, . \, [H_{3dB}] = \frac{1}{2} \begin{bmatrix} 1 - e^{-j\frac{2\pi f}{FSR}} & -j\left(1 + e^{-j\frac{2\pi f}{FSR}}\right) \\ -j\left(1 + e^{-j\frac{2\pi f}{FSR}}\right) & -\left(1 - e^{-j\frac{2\pi f}{FSR}}\right) \end{bmatrix} \tag{4}$$

An additional attenuation constant has been introduced describing insertion and internal loss.
To ensure that the specified center frequency of port #1 coincides with the maximum of the transmission function obtained at output #1, the FSR and the center frequency are adjusted automatically.
To simulate e.g. an optical pre-amplifier with a gain shape determined by the MZF transfer function, white noise with user-defined PSD can be added at the filter input representing ASE.

The noise will be shaped by the transfer function at the filter output. The noise can be treated numerically by randomly generated noise samples. Alternatively, the noise variance is calculated analytically at the filter output and can be used for further processing.

Input / Output

	Number	Type of signal
Input	1	Optical field
Input	2	Optical field
Output	1	Optical field
Output	2	Complementary Optical field

Component Parameters

Parameter Name	f0
Description	Center frequency
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	FSR
Description	FSR
Default Value	799.59
Parameter Range	0< FSR < Frequency Range
Unit	GHz

Parameter Name	alpha
Description	Additional loss
Default Value	0
Parameter Range	-1e150≤ alpha ≤ 1e150
Unit	dB

References

[1] J.Kastner: “Übertragungsverzerrungen durch Filter und Multiplexern in optischen WDM-Systemen”, Diploma thesis, Chair of High Frequency Technique, University of Dortmund, 1996

[2] D.J.G. Mestdagh: “ Fundamentals of multi-access optical fiber networks”, Artech House, Inc., Norwood/USA, 1995

[3] J. Lenge: “Modellierung optischer Netze mit Wellenlängevermittlung”, Diploma thesis, Chair of High Frequency Technique, University of Dortmund, 1997

7.5.9 MZI Lattice Structure

Fundamentals

The Mach-Zehnder interferometer (MZI) lattice structure shown below is composed of multiple elementary MZI filters. A single MZI consists of two subsequent 3 dB couplers with an additional phase shift applied to one of the two ports between the couplers. The total lattice structure is formed by a symmetric initial MZI with phase shift θ_{start} and a sequence of subsequent stages that consist of one asymmetric and one symmetric MZI each. The MZI structures are modeled as FIR filters.

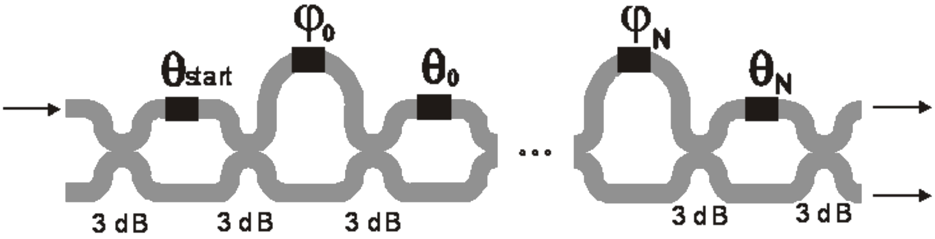


Figure 7.5.2. n-1th order MZI lattice structure

For each stage i, the optical amplitude A_{in} at the two input ports and A_{out} at the corresponding outputs are coupled by the transfer matrix $\mathbf{S_i}$ according to

$$\begin{pmatrix} A_{out,1} \\ A_{out,2} \end{pmatrix} = \mathbf{S_i}(f) \cdot \begin{pmatrix} A_{in,1} \\ A_{in,2} \end{pmatrix} \tag{1}$$

$$\mathbf{S_i} = \frac{1}{2} \begin{bmatrix} \left(e^{-j\theta_i} - 1\right) \cdot e^{-j2\pi(f-f_0)T-j\varphi_i} & -j\left(e^{-j\theta_i} + 1\right) \cdot e^{-j2\pi(f-f_0)T-j\varphi_i} \\ -j\left(e^{-j\theta_i} + 1\right) & -\left(e^{-j\theta_i} - 1\right) \end{bmatrix} \tag{2}$$

with frequency f, center frequency f_0, symmetric MZI phase shift θ_i, asymmetric MZI phase shift φ_i, and asymmetric MZI path delay difference T (identical for all asymmetric MZI filters). The symmetric MZI coupler coefficient k_i follows

$$k_i = -10 \cdot \log_{10}\left(1 - \cos^2\frac{\theta_i}{2}\right) \tag{3}$$

Input / Output

	Number	Type of signal
Input	1 or 2	Optical field
Output	2	Optical field

Component Parameters

Parameter Name	T
Description	asymmetric MZI path delay difference
Default Value	10
Parameter Range	T < Block Time
Unit	ps

Parameter Name	FSR
Description	Free spectral Range ($FSR = 1/T$)
Default Value	100
Parameter Range	FSR < Frequency Range
Unit	GHz

Parameter Name	Stages
Description	Number of stages (filter order)
Default Value	2
Parameter Range	1 ≤ Stages

Parameter Name	f0
Description	Center frequency
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	theta_start
Description	initial MZI phase shift (θ_{start})
Default Value	$\pi/2$
Parameter Range	not limited
Unit	rad

Parameter Name	theta_i
Description	symmetric MZI phase shift in stage i (θ_i)
Default Value	$\pi/2$
Parameter Range	not limited
Unit	rad

Parameter Name	k_start
Description	initial MZI coupler coefficient
Default Value	3
Parameter Range	not limited
Unit	dB

Parameter Name	k_i
Description	symmetric MZI coupler coefficient in stage i
Default Value	3
Parameter Range	not limited
Unit	dB

Parameter Name	phi_i
Description	asymmetric MZI phase shift in stage i (φ_i)
Default Value	$\pi/2$
Parameter Range	not limited
Unit	rad

References

none

7.5.10 SF Filter (Separated Channels)

Fundamentals

The SC Filter can only be used, if the *Separated Channels* method is selected in the *Simulation parameters*. Each WDM-channel is then saved in a separate signal vector.
Using the SC Filter (separated channels filter) one signal is filtered out from a couple of WDM-channels. The approximate center frequency of the desired WDM signals can be defined in the component dialog. The output of the SC Filter always contains only one signal vector. The SC Filter only selects one signal vector, the transfer function for this selected signal is an allpass.
Linear crosstalk can be avoided by placing an SC Filter before a photo diode at the receiver.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	Frequency
Description	Frequency of the desired channel
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

References

none

7.5.11 Transversal Filter

Fundamentals

The transversal filter is a synopsis of a recursive and a transversal digital filter. It is presented in a direct structure.
Exciting the system with a pulse, it passes through the runtime elements and appears - weighted with the particular coefficients - at the output.

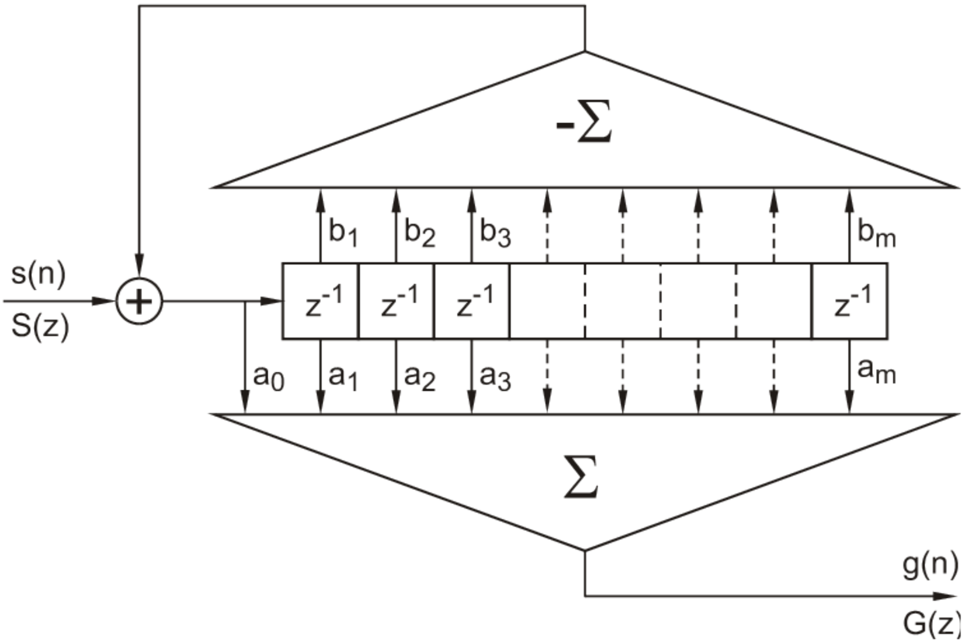


Figure 7.5.3. Direct structure of a universal discrete filter

Transfer function for representation by coefficients

$$H(z) = \frac{\sum\limits_{\nu=0}^{m} a_\nu z^{-\nu}}{1 + \sum\limits_{\mu=1}^{m} b_\mu z^{-\mu}} \tag{1}$$

Transfer function for pole- / null representation

$$H(z) = \alpha_{additional} \frac{\prod\limits_{\nu=1}^{m} (z - z_{0\nu})}{\prod\limits_{\mu=1}^{m} \left(z - z_{\infty\mu}\right)} \tag{2}$$

H :	Filter transfer function, z -domain
$\alpha_{additional}$:	Additional loss
A :	Coefficients of numerator
B :	Coefficients of denominator

where

$$z = \exp(j2\pi (f - f_0) T) \tag{3}$$

f_0 :	Filter center frequency
T :	Sampling time

Input / Output

	Number	Type of signal
Input	1	Optical field / electrical signal
Output	1	Optical field / electrical signal

Component Parameters

Parameter Name	f0
Description	Center frequency
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	T
Description	Time delay
Default Value	25
Parameter Range	0 < T < Block Time
Unit	ps

Parameter Name	Type
Description	Type
Default Value	coffs
Parameter Range	coeffs, null_pole

Parameter Name	NoIIR
Description	Number of IIR coefficients
Default Value	2
Parameter Range	0 ≤ NoIIR

Parameter Name	NoFIR
Description	Number of FIR coefficients
Default Value	2
Parameter Range	0 < NoFIR

Graphs

The complex amplitude transfer function of the loaded data can be visualized in the frequency domain.

References

[1] H. Götz: " Einführung in die digitale Signalverarbeitung", Teubner Studienskripten, 1995

[2] B. Wendland: "Skript zur Vorlesung Nachrichtentechnik I", Universität Dortmund, 1993

[3] H.D. Lüke: Signalübertragung, Springer, 1997

7.6 Multiplexers

7.6.1 Arrayed Waveguide Grating (AWG)

Fundamentals

Arrayed waveguide gratings (AWGs) serve as multiplexers and demultiplexers in WDM applications. They consist of two free propagation zones (FPZs). Both FPZs are connected by arrayed waveguides of increasing length, where ΔL is the length difference between two neighboring waveguides. In the FPZs, the field propagates as a Gaussian beam.

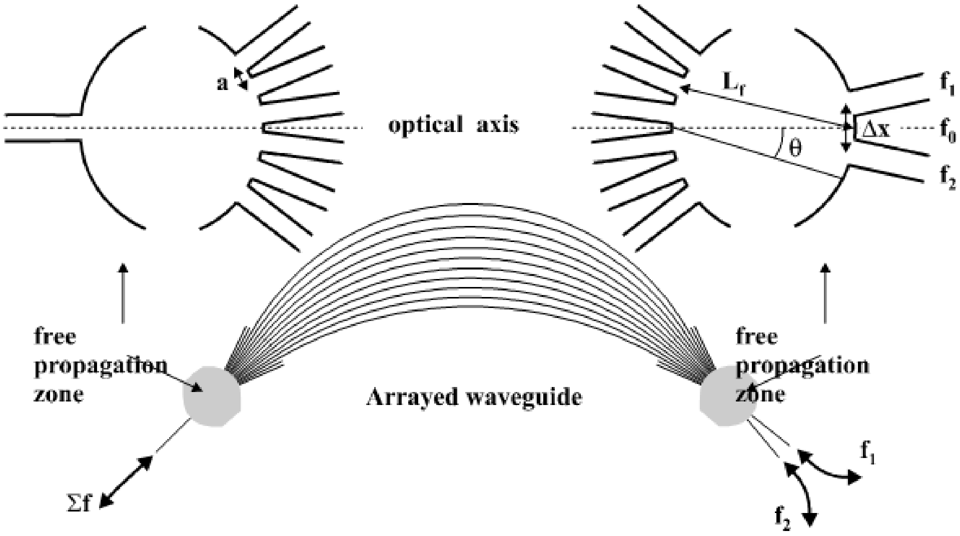


Figure 7.6.1. Basic AWG Structure

Demultiplexing can be described as follows: After passing the first FPZ, the input beam is coupled into the arrayed waveguides. A uniform illumination of the waveguides is assumed. Propagating through the waveguides, the optical waves are subject to different phase shifts. In the second FPZ, the signals interfere constructively or destructively and form a wavelength-dependent interference pattern.
The transfer function of the AWG is periodic. The free spectral range (FSR) denotes the frequency spacing between the maxima of the interference pattern and can be written as

$$\mathrm{FSR} = \mathrm{K}\Delta f \tag{1}$$

K :	Number of channels
Δf :	Channel spacing

f_0 is defined as the passband center frequency on the optical axis ($\theta = 0$) and is given by the constant length difference of the arrayed waveguides

$$\Delta L = m\frac{c_0}{n_c f_0} \tag{2}$$

where

$$m = \frac{f_0}{\mathrm{FSR}} \tag{3}$$

ΔL :	Length difference of arrayed waveguides
m :	Diffraction order
n_c :	Effective refractive index of arrayed waveguides
f_0 :	Center frequency
c_0 :	Velocity of light

denotes the diffraction order. The focal length L_i of the FPZ equals [1]

$$L_f = \frac{\Delta x}{\Delta f} \frac{n_s a f_0^2}{m c_0}. \tag{4}$$

Δx :	Distance between input and output waveguides, respectively
n_s :	Effective refractive index of FPZs
f_0 :	Center frequency
a :	Distance between arrayed waveguides

The coupling coefficients κ_i describing the coupling from the input waveguide to the i$^{\text{th}}$ arrayed waveguide are calculated utilizing the Gaussian beam approach [2]

$$\kappa_i = \frac{w_0}{w(\theta)} \exp\left(-\frac{{L_f}^2 \sin^2\theta}{w(\theta)^2}\right) \tag{5}$$

w :	Spot size of Gaussian beam
w_0 :	Initial spot size
θ :	Angle between waveguide and optical axis

where the spot size is given by

$$w(\theta) = \sqrt{w_0^2 + \left(\frac{c_0 L_f \cos\theta}{f_0 n_s w_0}\right)^2} \tag{6}$$

Due to reciprocity, the coupling between the FPZ and the output waveguides can be described by the same coupling coefficients. The transfer function for an output waveguide on the optical axis can be obtained by summing the contributions of the different arrayed waveguides [2]

$$H_0(f) = \sum_{i=1}^{I} \kappa_i^2 \exp\left(j 2\pi i n_c \Delta L \frac{f}{c_0}\right) \tag{7}$$

Assuming a Gaussian illumination in dependence on θ the transfer function of the output waveguides is weighted according to [3]

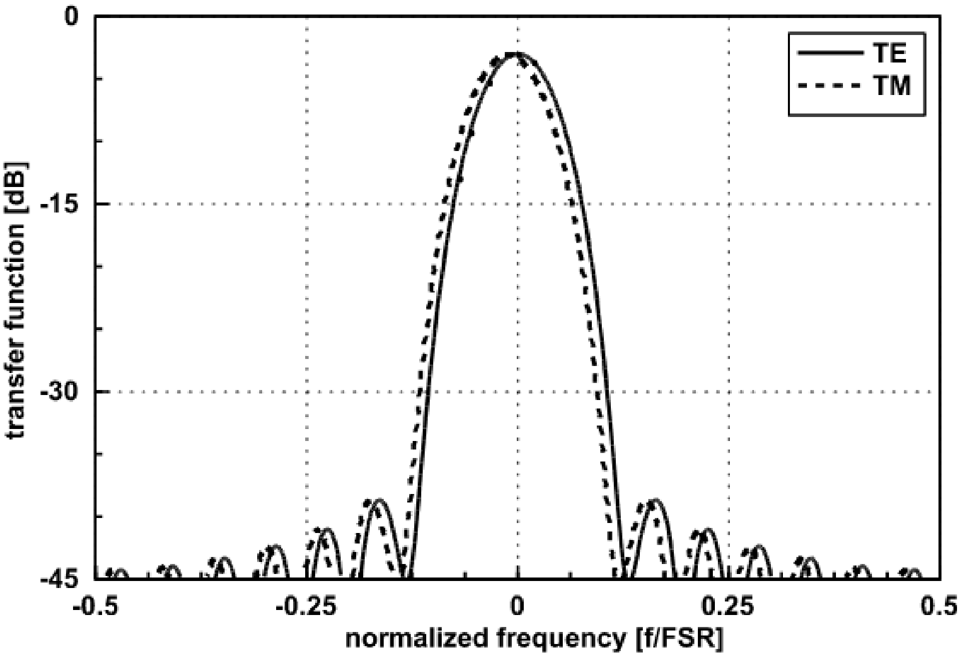


Figure 7.6.2. AWG transfer function (one channel)

$$H_k(f) = H_0(f)\exp\left(-\frac{\theta_k^2}{\theta_0^2}\right) \tag{8}$$

The weighting is performed only if the option “calculate envelope loss” is used. Here we have introduced the normalization constant

$$\theta_0 = \frac{c_0}{f_0 n_s \pi w_0}. \tag{9}$$

The center frequency f_k of channel k and the angle between the waveguide and the optical axis are related by

$$\theta_k = \frac{\Delta x}{\Delta f}\frac{f_k - f_0}{L_f} \tag{10}$$

The polarization dependence of the AWG can be taken into account by using different effective refractive indices for TE and TM modes, while retaining the length difference between the arrayed waveguides constant.
Insertion loss is modeled by means of an effective attenuation constant.

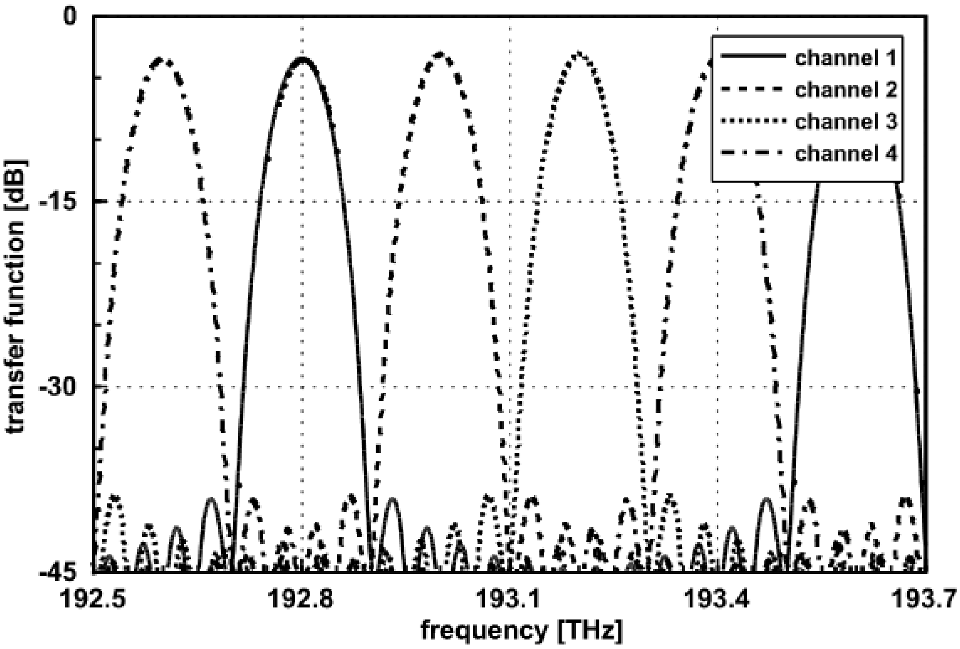


Figure 7.6.3. AWG transfer functions (4 channels)

All channels are spaced equally around the center frequency. The indices of the input and output ports of the AWG grow with increasing frequency, e.g. input/output #1 is assigned to the lowest frequency channel.

A frequency mismatch of the AWG transfer functions can be set for each channel to account for wavelength instabilities or temperature-dependent effects.

To simulate fabrication tolerances, the coupling coefficients κ_i as well as the lengths of the arrayed waveguides ΔL_i can be varied. Zero mean Gaussian random variables with user-defined variances may be added to the coupling coefficients and/or the waveguide lengths if desired. Alternatively, κ_i and ΔL_i can be directly read from data files. Both data files are structured containing one coefficient/waveguide per line.

Example for AWG coupling coefficients file:

```
a.  121
b.  379
c.  1003
d.  ...
```

Example for AWG lengths file:

```
1 0
2 5937
3 11875
4 ...
```

As all other multiplexers, the AWG cannot be bypassed.

Input / Output

	Number	Type of signal
Input	1,2,...	Optical field
Output	1,2,...	Optical field

Component Parameters

Parameter Name	channel spacing
Description	Channel spacing (Δf)
Default Value	200
Parameter Range	0 < channel spacing ≤ Frequency Range
Unit	GHz

Parameter Name	f0
Description	Reference Frequency
Default Value	193.1
Parameter Range	inside the PHOTOSS simulation band
Unit	THz

Parameter Name	loss dB
Description	Loss per channel (α_{dB})
Default Value	3
Parameter Range	0 ≤ loss dB ≤ 10
Unit	dB

Parameter Name	w0
Description	Spot Radius
Default Value	4.5
Parameter Range	1 ≤ w0 ≤ 100
Unit	µm

Parameter Name	d
Description	Distance between arrayed waveguides
Default Value	25
Parameter Range	0 ≤ d ≤ 1000
Unit	µm

Parameter Name	delta-x
Description	Distance between output waveguides (Δx)
Default Value	25
Parameter Range	0.1 ≤ d ≤ 1000
Unit	µm

Parameter Name	num fan
Description	Number of arrayed waveguides
Default Value	16
Parameter Range	0 < num fan

Parameter Name	nc te
Description	Refractive index cladding (TE)
Default Value	1.46146
Parameter Range	0≤ nc te
Unit	µm

Parameter Name	ns te
Description	Refractive index substrate (TE)
Default Value	1.463
Parameter Range	0≤ ns te
Unit	µm

Parameter Name	nc tm
Description	Refractive index cladding (TM)
Default Value	1.46146
Parameter Range	0≤ nc tm
Unit	µm

Parameter Name	ns tm
Description	Refractive index substrate (TM)
Default Value	1.463
Parameter Range	0≤ ns tm
Unit	µm

Parameter Name	rnd_cpl
Description	Variation of coupling coefficients
Default Value	none
Parameter Range	none, Gaussian, Read From File

Parameter Name	file cpl
Description	Input file for coupling coefficients
Default Value	none
Parameter Range	valid file path

Parameter Name	rnd dl
Description	Variation of arrayed waveguide length
Default Value	none
Parameter Range	none, Gaussian, Read From File

Parameter Name	var dl
Description	Input file for variation of AWG length
Default Value	none
Parameter Range	valid file path

Parameter Name	is demux
Description	Multiplexer (true) or Demultiplexer (false)
Default Value	true
Parameter Range	boolean

Parameter Name	envelope
Description	Calculate envelope loss
Default Value	false
Parameter Range	boolean

Parameter Name	map inputs
Description	Automatically order inputs according to their frequency
Default Value	false
Parameter Range	boolean

References

[1] H. Takahashi, K. Oda, T. Toba, Y. Inoue, “Transmission Characteristics of Arrayed Waveguide NxN Wavelength Multiplexer”, J. Lightwave Technol., Vol. 13, Nr. 3, pp. 447-445, 1995

[2] J. Kastner: “Übertragungsverzerrungen durch Filter und Multiplexer in optischen WDM- Systemen”, Diploma thesis, Chair of High Frequency Technique, University of Dortmund, 1996

[3] L. Spiekman, “Compact Integrated Optical Components for Telecommunication Networks”, PhD Thesis, Delft University of Technology, 1996

7.6.2 Coupler

Fundamentals

The coupler component is an ideal optical star coupler that acts as a power splitter, combiner, or a combination of both. N_{in} optical input signals are combined and equally distributed onto N_{out} identical output signals according to

$$A_{out} = \sqrt{\frac{\alpha}{N_{in} \cdot N_{out}}} \cdot \sum_{i=1}^{N_{in}} A_i \tag{1}$$

where A_{out} is the amplitude of all output signals, A_i are the amplitudes of the input signals, and α is an additional coupling loss.
As all other multiplexers, the coupler cannot be bypassed.

Input / Output

	Number	Type of signal
Input	N_{in}	Optical field
Output	N_{out}	Optical field

Component Parameters

Parameter Name	InPorts
Description	Number of input ports
Default Value	1
Parameter Range	1

Parameter Name	OutPorts
Description	Number of output ports
Default Value	1
Parameter Range	1

Parameter Name	Additional_loss
Description	Additional Loss
Default Value	0
Parameter Range	$-100 \leq$ Additional_loss ≤ 100
Unit	db

References

[1] J. Lenge: “Modellierung optischer Netze mit Wellenlängenvermittlung”, diploma thesis at the Chair of High Frequency Technique, University of Dortmund, 1997

7.6.3 3dB-Coupler

Fundamentals

Optical 3 dB couplers are passive devices for combining and splitting optical signals. Two input signals a1 and a2 are related to the output signals b1 and b2 according to

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & j \\ j & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \tag{1}$$

with an optional additional coupling loss.
As all other multiplexers, the 3 dB coupler cannot be bypassed.

Input / Output

	Number	Type of signal
Input	1 or 2	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	InPorts
Description	Number of input ports
Default Value	1
Parameter Range	1

Parameter Name	OutPorts
Description	Number of output ports
Default Value	1
Parameter Range	1

Parameter Name	Additional_loss
Description	Additional Loss
Default Value	0
Parameter Range	-100 ≤ Additional_loss ≤ 100
Unit	db

References

[1] Denis J. G. Mestdagh: Fundamentals of multi-access optical fiber networks, pages 178 and 233, Artech House, Inc., Norwood/USA, 1995

7.6.4 TDM Multiplexer

Fundamentals

The time domain multiplexing (TDM) multiplexer acts as an ideal combiner of multiple optical input signals into a single optical output signal. The bit rate BR_{out} of the output signal is automatically set to

$$BR_{out} = N_{in} \cdot BR_{\max} \tag{1}$$

where N_{in} is the number of input signals, and BR_{max} is the maximum bit rate of all input signals.
Caution: In order to preserve the ability to recover the bit pattern, the pulses of the input signals should not overlap in the time domain.

As all other multiplexers, the coupler cannot be bypassed.

Input / Output

	Number	Type of signal
Input	N	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	InPorts
Description	Number of input ports
Default Value	1
Parameter Range	1 ≤ InPorts
Unit	GHz

References

none

7.7 Polarization Components

7.7.1 Birefringent Element

Fundamentals

The birefringent element (“DGD element”) is an optical element with a Jones matrix that has two frequency independent eigenvectors that are characterized by different refractive indices. The Jones matrix **T** of such an element with the differential group delay (DGD) $\Delta\tau$ and the direction $\vec{s} = \begin{pmatrix} s_1 & s_2 & s_3 \end{pmatrix}^{\mathrm{T}}$ in Stokes space is

$$\mathbf{T}(\omega) = \cos(\Delta\tau\ \omega/2)\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - j\ \sin(\Delta\tau\ \omega/2)\begin{pmatrix} s_1 & s_2 - j\ s_3 \\ s_2 + j\ s_3 & -s_1 \end{pmatrix} \qquad (1)$$

with the imaginary unit j and the angular frequency ω.

For example, a linear birefringent element with the direction $\vec{s} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^{\mathrm{T}}$ in Stokes space (aligned to the x-axis in Jones space) has the Jones matrix

$$\mathbf{T}(\omega) = \begin{pmatrix} \exp\left(-j\frac{\Delta\tau\ \omega}{2}\right) & 0 \\ 0 & \exp\left(j\frac{\Delta\tau\ \omega}{2}\right) \end{pmatrix} \qquad (2)$$

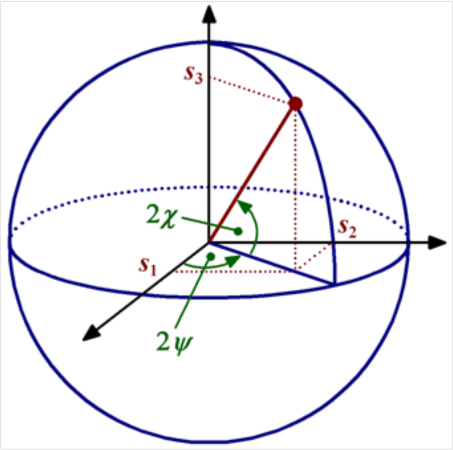


Figure 7.7.1. Poincaré-Sphere

The direction of the element in Stokes space can be set employing the spherical angular coordinates 2ψ and 2χ, too. The relation between the Stokes parameters $s1$, $s2$, $s3$ and the angles 2ψ and 2χ of the Poincaré sphere is given by

$$\begin{aligned} s_1 &= \cos(2\chi)\cdot\cos(2\psi) \\ s_2 &= \cos(2\chi)\cdot\sin(2\psi) \\ s_3 &= \sin(2\chi) \end{aligned} \qquad (3)$$

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	DGD
Description	Differential Group Delay
Default Value	10
Parameter Range	0 < DGD ≤ 10.000
Unit	ps

Parameter Name	s1, s2, s3
Description	Cartesian coordinates si of the slowest polarization state on the Poincaré sphere
Default Value	[1,0,0]
Parameter Range	-1 ≤ s_i ≤ 1
Unit	normalized

Parameter Name	2*Chi
Description	spherical azimuthal coordinate of slowest polarization state on Poincaré sphere
Default Value	0
Parameter Range	-90 ≤ 2*Chi ≤ 90
Unit	degree

Parameter Name	2*Psi
Description	spherical polar coordinate of slowest polarization state on Poincaré sphere
Default Value	0
Parameter Range	0 ≤ 2*Psi ≤ 360
Unit	degree

Parameter Name	Random direction
Description	Random direction of the birefringent element in Stokes space?
Default Value	false
Parameter Range	boolean

References

[1] Damask, Jay N.: Polarization Optics in Telecommunications / Rhodes, William T. (Ed.). New York : Springer, 2005 (Springer Series in Optical Sciences). - ISBN 0-387-22493-9

7.7.2 Bitpolarimeter

Fundamentals

The Bitpolarimeter serves to store the time-dependent state of polarization of a signal into a file. In contrast to the polarimeter, this component provides further ways of selecting certain samples for analysis only. Furthermore additional results are computed: These are the DOP (averaged over the box of all '1' bits or '0' bits) and the standard deviation of the DOP of the individual '1' bits or '0' bits (with regard to the time averaged DOP). If *Auto_save* is set, the file *PolState Name of the Component* created in the simulation file directory. It contains both the Stokes and Jones parameter representation of the polarization state.
The Threshold parameter can be used to define a minimum amplitude level to be regarded in order to ignore intervals with very low signal amplitudes. For $Threshold = 10^{-10}$, the Stokes parameters are set to $s_1 = s_2 = s_3 = 0$ whenever the amplitude drops below 10^{-10} times the maximum amplitude.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output		no output

Component Parameters

Parameter Name	Threshold
Description	While calculating the (time dependent) state of polarization, time-ranges with very low amplitude-values can be ignored
Default Value	0
Parameter Range	not implemented

Parameter Name	Ignore output values below threshold
Description	Skip values below threshold
Default Value	false
Parameter Range	boolean

Parameter Name	Select Samples
Description	Select samples to be written into output file. Bit Center only analyzes the exact center bits. Bit Center Box analyzes all samples in a range of ±10% of the bit slot
Default Value	All
Parameter Range	All, Bit Center, Bit Center Box

Parameter Name	Auto_save (Time)
Description	Creates a file with the (time dependent) state of polarization in the directory of the .pho-file.
Default Value	true
Parameter Range	boolean

Parameter Name	Auto_save (Frequency)
Description	Creates a file with the (frequency dependent) state of polarization in the directory of the .pho-file.
Default Value	true
Parameter Range	boolean

Parameter Name	Auto_save (DOP of all '1'-bits)
Description	Creates a file with the degree of polarization of all '1'-bits in the directory of the .pho-file. The DOP is time-averaged over the box.
Default Value	false
Parameter Range	boolean

Parameter Name	Auto_save (DOP of all '0'-bits)
Description	Creates a file with the degree of polarization of all '0'-bits in the directory of the .pho-file. The DOP is time-averaged over the box.
Default Value	false
Parameter Range	boolean

References

none

7.7.3 PMD Compensator

Fundamentals

The PMD Compensator (PMDC) can be used to compensate for (small) amounts of PMD (DGD and higher order PMD) in an optical network system.
Generally, a PMDC works effectively if the mean DGD of a System is not too big and the bandwidth of the PSPs (Principal States of Polarization) $\langle\Delta\tau\rangle$ is not too small while the bandwidth of the signal

$$\Delta\omega_{PSP} = \frac{(\pi/4)}{\langle\Delta\tau\rangle}$$

is itself limited. Quantitatively, the relation

$$\upsilon = \frac{\omega_{FWHM}}{\Delta\omega_{PSP}} \approx 1 \tag{1}$$

should be satisfied for a PMDC to work effectively. PHOTOSS offers three different modes of operation for the PMDC. Generally, you have to hoist the PMD-flag of all the components you want to be considered during the calculation of the PMD vector and you have to put a Transmission Matrix Analyzer (TMA) at the end of the system of PMD-flag components so as to calculate the PMD vector.

Modes of Operation

3 DOF:

If you use the *3 DOF* mode of operation, the PMDC consists of a polarization controller (PC), a birefringent element including a fixed direction of the vector s_+ of the birefringence and a variable DGD $\Delta\tau$, and a pulse width analyzer (PWA).
The direction can be described by supplying two angles in Stoke space - equal to two degrees of freedom (DOF) - and the element's DGD $\Delta\tau$ can be varied which supplies a total of three DOFs for the complete system.
The feedback signal can be chosen to be either DOP (using the degree of polarization of the signal at the output) or the RMS pulse width (in case of a single pulse transmission). By automatically varying the angels on the Poincaré sphere and the element's DGD with a given precision, the PMDC tries to find an optimal combination of the three DOFs where the pulse width T_{RMS} at the component's output is minimal (and thus the DOP is maximal). In 3 DOF mode, the mean DGD of the whole system may be increased slightly.

2 DOF:

Using the *2 DOF* mode of operation, the PMDC behaves similar to the *3 DOF* mode of operation with one exception: The PMDC consists of a birefringent element which has a **fixed** DGD $\Delta\tau$, thus the PMDC only has 2 DOFs. The constant should be chosen carefully: A value too low reduces the effectiveness of the PMDC while a value too high enlarges the mean DGD of the whole system $\langle\Delta\tau\rangle$, thereby reducing $\Delta\omega_{PSP}$ and increasing the factor υ in equation (1).

1st Order:

In *1st order* mode the PMDC consists of a single birefringent element that allows you to compensate the (first order) PMD vector of the component(s) in your PMD path at the center frequency of the (single) pulse by cancelling out the according transmission matrix.
The *1st order* PMDC does not generate a higher mean DGD $\langle\Delta\tau\rangle$ of the whole system but has several disadvantages:
In time-dependent systems, a permanent measurement of the signal distortion at the system output has to

be carried out while the gathered information has to be transferred **backwards** to the system input where the state of polarization has to be adjusted accordingly. This disadvantage can prove to be problematic when dealing with very fast or instantaneous changes is in the system properties.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	Feedback Signal
Description	type of feedback Signal
Default Value	DOP
Parameter Range	DOP, PulseWidth
Mode of Operation	3 DOF / 2 DOF

Parameter Name	Angle: Precision of Optimization
Description	defines the precision with which the angels are optimized
Default Value	0.1
Parameter Range	0 < Angle < 90
Unit	degrees
Mode of Operation	3 DOF / 2 DOF

Parameter Name	DGD: Precision of Optimization
Description	defines the precision with which the DGD of the birefringent element optimized (only in 2 DOF mode)
Default Value	0.1
Parameter Range	0 < DGD < 1.000.000
Unit	ps
Mode of Operation	2 DOF

References

none

7.7.4 PMD Emulator

Fundamentals

The *PMD Emulator* acts as a source for Polarization Mode Dispersion (PMD). In contrast to the *PMD + PDL Emulator*, the *PMD Emulator* utilizes a different working principle to generate PMD (see below).

Please refer to the component *PMD + PDL Emulator* for a detailed explanation of the PMD phenomenon.

Working principle:
Instead of calculating the DGD and higher order PMD for a concatenation of a fixed number of birefringent elements, the *PMD Emulator directly* creates first and second order PMD by calculating the appropriate Jones transmission matrices. Higher order PMD is **not** emulated. For implementation details please refer to [1].

Modes of Operation

The PMD Emulator offers three basic modes of operation:

1. **DGD an DEP:** You can specify values for the desired and depolarization (DEP) component of the second order PMD which will then automatically be generated at the specified reference frequency of the signal.
2. **DGD an PCD:** You can specify values for the desired DGD and the polarization dependent chromatic dispersion (PCD) component of the second order PMD which will then automatically be generated at the specified reference frequency of the signal.
3. **4 birefringent elements:** This mode of operation emulates a set of four birefringent elements. You can specify the desired DGD, PCD and DEP of this concatenation of elements as well as the reference frequency at which these effects should be considered.

Remarks

To take the settings of the *PMD Emulator* into account when considering a PMD-Path in a network with multiple birefringent elements, you have to hoist the PMD-Flag of the component in order to make its influence on the signal “visible” to the *Transmission Matrix Analyzer*.
The output PMD-vector of the *PMD Emulator* is parallel to the vector (1, 0, 0) in Stokes Space at the specified reference frequency. If you want the output to have a different orientation, you have to put a *Polarization Controller* at the end of the *PMD Emulator*.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	Reference Frequency
Description	The frequency at which the PMD-effects shall be considered.
Default Value	193.1
Parameter Range	withing the PHOTOSS simulation bandwidth
Unit	THz
Mode of Operation	DGD and DEP, DGD and PCD, 4 biref. elements

Parameter Name	DGD
Description	The desired differential group delay.
Default Value	10
Parameter Range	0 ≤ DGD ≤ 1.000.000
Unit	ps
Mode of Operation	DGD and DEP, DGD and PCD, 4 biref. elements

Parameter Name	Depolarization
Description	The desired depolarization (component of the second order PMD) of the output signal.
Default Value	100
Parameter Range	0 ≤ Depolarization ≤ 1.000.000
Unit	ps^2
Mode of Operation	DGD and DEP, 4 biref. elements

Parameter Name	PCD
Description	The desired polarization-dependent chromatic dispersion (component of the second order PMD) of the output signal
Default Value	0
Parameter Range	-1.000.000 ≤ PCD ≤ 1.000.000
Unit	ps^2
Mode of Operation	DGD and PCD, 4 biref. elements

References

[1] P. M. Krummrich, “A novel, easy to use Emulator for deterministic generation of pure first and second order PMD”, Conference on Optical Fiber Communication (OFC 2007), March 25-29, Anaheim, CA, USA, OMH3.

7.7.5 PMD + PDL Emulator

Fundamentals

The *PMD and PDL Emulator* acts as a source for Polarization Mode Dispersion (PMD) and / or Polarization Dependent Loss (PDL) by simulating a concatenation of (birefringent) DGD and / or PDL elements.

Polarization Mode Dispersion:
PMD itself has its origin in the optical birefringence and the random variation of the birefringent axes orientation along the fiber length [1]. Even in an ideal symmetric fiber, two orthogonally polarized modes exist (slow and fast mode). In reality, a certain amount of imperfection due to the manufacturing process and / or mechanical stress after the placement of the fiber is always present and leads to the occurrence of PMD.
In the wave plate model, the fiber is considered to be composed of (many) concatenated sections which each have an individual (but uniform) birefringence and thus an individual (but uniform) birefringence axis. Both the birefringence axes and the magnitudes change randomly along the fiber, rendering PMD a **stochastic process**. The PMD and PDL Emulator mimics these stochastic characteristics of PMD by randomly varying the lengths of the wave plates and (if chosen) the amount of DGD per element as well as the orientation of the Stokes vector on the Poincaré sphere, if the emulator is set to the *Statistical Mode* of operation (see below).

The differential group delay (DGD), $\Delta\tau$, is defined as the group-delay difference between the slow and the fast modes: If a pulse is launched with equal power into the slow and fast axis of a fiber, the output consists of *two* pulses which are separated by the DGD.
Using the principal states model, the relation between PMD vector $\vec{\tau}$ in three dimensional Stokes space and the DGD $\Delta\tau$ can be expressed as follows [1]:

$$\vec{\tau} = \Delta\tau \cdot \vec{q} \tag{1}$$

where $\vec{q}$ denotes the unit Stokes vector in Stokes space aligned with the slow (fast) principal state of polarization (PSP) when considering the right- (left)-circular polarization in Stokes space.
As the fiber PMD vector varies with the optical angular frequency, a Taylor-Series expansion of $\tau(\omega)$ with $\Delta\omega$ about the carrier frequency is often used in combination with larger signal bandwidths. Thus, we have:

$$\vec{\tau}_\omega(\omega_0 + \Delta\omega) = \vec{\tau}(\omega_0) + \vec{\tau}_\omega(\omega_0)\Delta\omega + ... \tag{2}$$

Second-order PMD can be then be described by

$$\vec{\tau}_\omega = \Delta\tau_\omega\vec{q} + \Delta\tau\vec{q}_\omega, \tag{3}$$

where the first summand causes polarization-dependent chromatic dispersion (PCD) and is parallel to $\vec{\tau}$, the second summand causes depolarization (DEP) and is perpendicular to $\vec{\tau}$.
The *PMD and PDL Emulator* of PHOTOSS automatically creates “all order” PMD, by using the Jones matrix expression (for details see the help entry on the component “Birefringent Element” on page 253) for PMD

$$T = \cos(\frac{2 \cdot \pi \cdot f \cdot \Delta\tau}{2}) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - i \sin(\frac{2 \cdot \pi \cdot f \cdot \Delta\tau}{2}) \begin{bmatrix} s_1 & s_2 - i \cdot s_3 \\ s_2 + i \cdot s_3 & -s_1 \end{bmatrix}, \tag{4}$$

where s_i are the components of the Stokes vector on the Poincaré sphere and $\Delta\tau$ denotes the DGD of each individual birefringent element.
By supplying the number of DGD elements and the mean DGD (in ps) per element, the estimated mean DGD of the fiber is calculated. For each DGD element, a relative standard deviation in percent of the mean value can be specified, while the DGD of each element is itself distributed in a Gaussian form.

Generally, you have to hoist the PMD flag for the *PMD and PDL Emulator* if you want to take the simulated effects of PMD and PDL into account. If the *PMD and PDL Emulator* is not part of the PMD path, its generated PMD and PDL will **not** influence the signal.

Polarization Dependent Loss:
The Jones operator of a PDL element (in analogy to a DGD element of the emulator) can be described as

$$T = e^{(-\frac{\alpha}{2})} \cdot \cosh(\frac{\Delta\alpha}{2}) \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \sinh(\frac{\Delta\alpha}{2}) \cdot \begin{bmatrix} s_1 & s_2 - i \cdot s_3 \\ s_2 + i \cdot s_3 & -s_1 \end{bmatrix}, \tag{5}$$

where $\Delta\alpha$ is defined as the (differential) attenuation coefficient. The PMD and PDL automatically calculates the transmission matrix T which is then applied to all components in the following PMD path to include PDL effects.
By setting the number of PDL elements in the concatenation to a value higher than zero, PHOTOSS automatically includes the specified number of PDL elements into the concatenation by distributing the supplied mean PDL per element using a Gaussian form; the mean number of DGD elements between two PDL elements is automatically calculated and shown in the component dialog. You can also manually define the first and / or the last element of the concatenation of PMD and PDL elements to exclusively be a PDL element to closely emulate the system behavior with a higher PDL at the input and output.

Modes of Operation

The *PMD and PDL Emulator* can operate in three different modes:

1. **Statistical:** A complete random system is generated due to the parameters specified in the PMD and PDL Emulator (mean DGD per element, number of elements, etc.).

2. **Semi-Statistical:** The positions, the types and the DGD and PDL values of the emulator's elements are chosen based on the file `PMD_PDL_Emulator_Load.txt` which has to be located in the directory of the current PHOTOSS simulation. The directions of the emulator's elements are determined randomly. Thus, the *PMD and PDL Emulator* produces different overall DGDs for the emulated system each time PHOTOSS is run.

3. **Deterministic:** Based on the file `PMD_PDL_Emulator_Load.txt` the whole set of parameters (the waveplate positions, the types, the DGD and PDL values and the direction of the emulator's elements) is chosen. The file has to be located in the directory of the current PHOTOSS simulation.

In the S*emi-Statistical* and the *Deterministic Mode*, both the DGD and the PDL values which are specified in the `PMD_PDL_Emulator_Load.txt` can be scaled if you check the radio box "Scale DGD" / "Scale PMD". In the *Statistical Mode*, the "Save" radio box can be checked to save all parameters of the emulated concatenation (DGD and PDL value per element, components of the Stokes vector) to the file `PMD_PDL_Emulator_Save.txt`.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	Save
Description	Number of input ports
Default Value	false
Parameter Range	boolean
Mode of Operation	Statistical

Parameter Name	#DGD Elements
Description	Number of the DGD elements (the waveplates)
Default Value	1000
Parameter Range	1 ≤ #DGD Elements ≤ 1.000.000
Mode of Operation	Statistical

Parameter Name	#PDL Elements
Description	Number of the PDL elements
Default Value	0
Parameter Range	0 ≤ #PDL Elements ≤ 1.000.000
Mode of Operation	Statistical

Parameter Name	Mean # DGD Elements
Description	Average value of the number of DGD elements between two PDL elements (read only)
Default Value	automatically derived
Parameter Range	automatically derived
Mode of Operation	Statistical

Parameter Name	Standard Deviation # DGD elements between two PDL elements
Description	Specifies the distance between two PDL elements by supplying the relative standard deviation for a Gaussian distribution in percent of the mean number of DGD elements between two PDL elements
Default Value	10%
Parameter Range	0 < Standard Dev. ≤ 1.000.000
Mode of Operation	Statistical

Parameter Name	First element PDL
Description	Should the first element of the emulator be a PDL element?
Default Value	false
Parameter Range	boolean
Mode of Operation	Statistical

Parameter Name	Last element PDL
Description	Should the last element of the emulator be a PDL element?
Default Value	false
Parameter Range	boolean
Mode of Operation	Statistical

Parameter Name	Mean DGD per Element
Description	Mean DGD for all elements in the concatenation (the individual DGD is Gaussian distributed!)
Default Value	1
Parameter Range	0 ≤ Mean DGD per Element ≤ 1.000.000
Unit	ps
Mode of Operation	Statistical

Parameter Name	Mean PDL per Element
Description	Mean PDL for all elements in the concatenation (the individual PDL is Gaussian distributed)
Default Value	0.3
Parameter Range	0 ≤ Mean PDL per Element ≤ 1.000.000
Unit	dB
Mode of Operation	Statistical

Parameter Name	Standard Deviation DGD
Description	Standard deviation in percent from the mean DGD that each element may have from the Gaussian distribution)
Default Value	30 %
Parameter Range	0 ≤ Standard Deviation DGD ≤ 100
Mode of Operation	Statistical

Parameter Name	Scale DGD
Description	Scale input DGD?
Default Value	false
Parameter Range	boolean
Mode of Operation	Semi-Statistical / Deterministic

Parameter Name	Scale PDL
Description	Scale input PDL?
Default Value	false
Parameter Range	boolean
Mode of Operation	Semi-Statistical / Deterministic

References

[1] Kaminov, Ivan (Ed.): Optical Fiber Telecommunications IVB New York : Elsevier, 2002. - ISBN 0-12-395173-9

[2] Damask, Jay N.: Polarization Optics in Telecommunications / Rhodes, William T. (Ed.). New York : Springer, 2005 (Springer Series in Optical Sciences). - ISBN 0-387-22493-9

7.7.6 Polarimeter

Fundamentals

The Polarimeter serves to store the time dependent state of polarization of a signal into a file. If *Auto_save* is set, the file *PolState* < *Name of the Component*> created in the simulation file directory. It contains both the Stokes and Jones parameter representation of the polarization state.
The *Threshold* parameter can be used to define a minimum amplitude level to be regarded in order to ignore intervals with very low signal amplitudes. For $Threshold = 10^{-10}$, the Stokes parameters are set to $s1 = s2 = s3 = 0$ whenever the amplitude drops below 10^-10 times the maximum amplitude.

Graphs

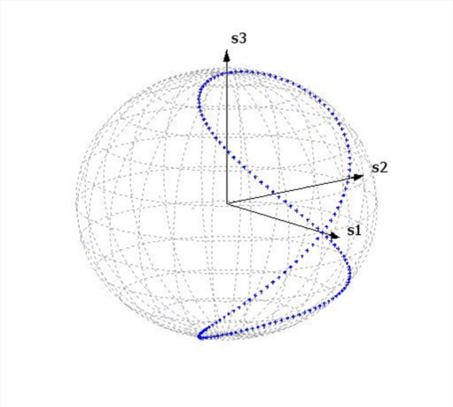


Figure 7.7.2. Poincaré-Sphere

The Polarimeter allows to visualize the signal polarization by plotting a 3D graph with normalized stokes vectors (Poincaré representation) [1, 2]. The positions on the equator represent the linear polarization states, while the poles are left-hand and right-hand circular polarization. The visualizer options allow plotting polarization vs. frequency and vs. time. The option “Intensity-induced transparency” scales the data point transparency with the signal power.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output		no output

Component Parameters

Parameter Name	Threshold
Description	While calculating the (time dependent) state of polarization, time-ranges with very low amplitude-values can be ignored
Default Value	0
Parameter Range	not implemented

Parameter Name	Auto_save
Description	Creates a file with the (time dependent) state of polarization in the directory of the .pho-file.
Default Value	true
Parameter Range	boolean

References

[1] A. Galtarossa, C. R. Menyuk, “Polarization Mode Dispersion”, Springer, 2005, ISBN: 978-0-387-23193-8.

[2] J. P. Gordon, H. Kogelnik, “PMD Fundamentals: Polarization Mode Dispersion in optical fibers ”, Proceedings of the National Academy of Sciences of the United States of America, Vol. 97, No. 9. (Apr 2000), pp. 4541-4550.

7.7.7 Polarizer

Fundamentals

The *Polarizer* acts as a linear polarizer with axis of transmission at Theta. The state of polarization (SOP) is changed by the Jones matrix **J**

$$\mathrm{SOP}_{out} = \mathbf{J} \cdot \mathrm{SOP}_{in} \tag{1}$$

with the Jones matrix being defined as

$$\mathbf{J} = \begin{pmatrix} \cos^2\Theta & -\sin\Theta\cos\Theta \\ -\sin\Theta\cos\Theta & \sin^2\Theta \end{pmatrix} \tag{2}$$

Alternatively you can use the *Polarizer* as a filter with regard to the output PSPs (center frequency) of a PMD-Path. You have to choose at least one fiber/component to be a member of the PMD-Path and you have to put a *Transmission Matrix Analyzer* after the last component of the PMD-Path in order to use this choice.
If you set the parameter *PSP filter* to *Fast PSP*, you use the *Polarizer* as a filter for polarization states which are aligned with the fast output PSP (center frequency of the simulation parameters) of the PMD-Path; the parts of the signal which are aligned with the fast PSP may pass, all other parts of the signal will be blocked.

For this to work correctly, you have to put the *Polarizer* **behind** the birefringent element (fiber, etc.) which is a member of the PMD-Path and after which you want to filter the signal. Normally, the output signal power of a *Polarizer* will be lower than the input signal power, assuming the signal is not completely aligned with the either the fast or slow PSP of the birefringent PMD-Path member in front of the *Polarizer*.

If you want to align the complete signal with any of the possible polarization states, please refer to the polarization component *Polarization Controller* instead.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	Theta
Description	Angle of the axis of transmission
Default Value	0
Parameter Range	-90 ≤ Theta ≤ 90
Unit	degree

Parameter Name	PSP filter
Description	Filter with regard to the output PSPs (PSPs at the center frequency of the simulation parameters) of a PMD-Path? All other parts which are not aligned with the slow (or fast) PSP will be blocked completely.
Default Value	No
Parameter Range	No, Fast PSP Filter, Slow PSP Filter

References

none

7.7.8 Polarization Controller

Fundamentals

The Polarization Controller changes the polarization state of an incident signal. Any well defined state on the Poincaré sphere may be chosen. The component offers two modes of operation:

1. **Set DOP:** You can manually define the state of polarization, thereby ignoring any input State of Polarization (SOP) at the input of the Polarization Controller. Thus, the output SOP exclusively depends on the settings of the Polarization Controller and not the input SOP. There are three different ways to define an SOP:

   First, you can define an SOP by supplying the values for the spherical angular coordinates of the SOP on the Poincaré-Sphere χ and ψ , where

   $$-90° \leq 2 \cdot \chi \leq 90°, \tag{1}$$

   and

   $$0° \leq 2 \cdot \psi < 360°. \tag{2}$$

   Secondly, you can also directly define the Cartesian the values for the SOP on the Poincaré sphere by defining the components of the Stokes vector s_1, s_2 and s_3. If you have instead defined χ and ψ , the Stokes-Vector will be calculated automatically by using

   $$s_1 = \cos(2 \cdot \chi) \cdot \cos(2 \cdot \psi), \tag{3}$$

   $$s_2 = \cos(2 \cdot \chi) \cdot \sin(2 \cdot \psi), \tag{4}$$

   $$s_3 = \sin(2 \cdot \chi), \tag{5}$$

   where

   $$-1 \leq s_i \leq 1, \tag{6}$$

   and

   $${s_1}^2 + {s_2}^2 + {s_3}^2 = 1. \tag{7}$$

   The third way to define a SOP is to directly supply the x- and y-component of the normalized Jones vector in Jones space. The relation between Stokes space and Jones space can be expressed as:

   $$s_1 = {e_x}^2 - {e_y}^2, \tag{8}$$

   $$s_2 = 2 \cdot e_x \cdot e_y \cdot \cos(\delta), \tag{9}$$

   and

   $$s_3 = 2 \cdot e_x \cdot e_y \cdot \sin(\delta), \tag{10}$$

   where

$$0 \leq e_i \leq 1, \tag{11}$$

and

$$e_x{}^2 + e_y{}^2 = 1. \tag{12}$$

In equations (3) to (12), e_x and e_y denote the magnitude of the normalized Jones vector in x- and y-direction and denotes a phase difference between the Jones vector's x- and y-component which is defined in the interval

$$-180^\circ < \delta \leq 180^\circ. \tag{13}$$

Generally, the state of polarization is modified by applying the Jones matrix **J**

$$\mathrm{SOP}_{out} = \mathbf{J} \cdot \mathrm{SOP}_{in} \tag{14}$$

in Jones space. The Jones matrix is defined by:

$$\mathbf{J} = \begin{pmatrix} \cos\Theta \exp\left\{-j\frac{1}{2}\varphi\right\} & -\sin\Theta \exp\left\{-j\frac{1}{2}\varphi\right\} \\ \sin\Theta \exp\left\{+j\frac{1}{2}\varphi\right\} & \cos\Theta \exp\left\{+j\frac{1}{2}\varphi\right\} \end{pmatrix} \tag{15}$$

It is also possible to align the output SOP parallel with the **slow** or the **fast** axis of the Principal State of Polarization (PSP). For this to work correctly, you have to place a Transmission Matrix Analyzer (TMA) at the end of your optical network. You also **have** to hoist the PMD-Flag on **all** components you want to take into consideration for the calculation of the PSPs. Once a simulation is run, the TMA will automatically calculate the correct transmission matrix and the Polarization Controller will align the output SOP parallel to the slow (fast) PSP of all components in the PMD-Path.

You can also choose to define an SOP which is perpendicular to the PSPs by checking the radio box: In the absence of PDL (Polarization Dependent Loss), the system's PSPs are orthogonal (orthogonal in Jones space - 180° in Stokes space). The SOP to be set is chosen to be a linear SOP (SOP on the equator of the Poincaré sphere) perpendicular to the PSPs in that case.

2. **Adjust SOP:** You can modify the input SOP by supplying both Θ and φ. The new Jones matrix for the output SOP of the signal is then calculated by using equation (13). The SOP of each frequency component of the signal is set accordingly.

Remarks

If you want to **filter** your output signal after it has propagated through a birefringent element (fiber, PMD element, etc.) with regard to its state of polarization, you should refer to the use of the polarization component *Polarizer* instead. This component is capable of filtering out (e.g.) any parts of the signal which are not aligned with the slow (or fast) PSP of a system of birefringent elements.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	2*Chi
Description	spherical azimuthal coordinate of SOP on Poincaré sphere
Default Value	0
Parameter Range	-90 ≤ 2*Chi ≤ 90
Unit	degree
Mode of Operation	Set SOP

Parameter Name	2*Psi
Description	spherical polar coordinate of SOP on Poincaré sphere
Default Value	0
Parameter Range	0 ≤ 2*Psi ≤ 360
Unit	degree
Mode of Operation	Set SOP

Parameter Name	s1, s2, s3
Description	Cartesian coordinates of Stokes vector of SOP on Poincaré sphere
Default Value	[1,0,0]
Parameter Range	-1 ≤ s_i ≤ 1
Unit	normalized
Mode of Operation	Set SOP

Parameter Name	e_x, e_y
Description	magnitudes of the normalized Jones vector in x- and y-direction
Default Value	[1,0]
Parameter Range	0 ≤ e_i ≤ 1
Unit	normalized
Mode of Operation	Set SOP

Parameter Name	Delta
Description	phase difference between the Jones vector's x- and y-component
Default Value	0
Parameter Range	180 < e_i ≤ 180
Unit	degree
Mode of Operation	Set SOP

Parameter Name	SOP ||slow PSP
Description	Set SOP parallel to slow PSP?
Default Value	false
Parameter Range	boolean
Mode of Operation	Set SOP

Parameter Name	SOP ||fast PSP
Description	Set SOP parallel to fast PSP?
Default Value	false
Parameter Range	boolean
Mode of Operation	Set SOP

Parameter Name	SOP perpendicular to PSPs
Description	Set SOP perpendicular to PSPs?
Default Value	false
Parameter Range	boolean
Mode of Operation	Set SOP

Parameter Name	Theta
Description	Elevation angle
Default Value	0
Parameter Range	not limited
Mode of Operation	Adjust SOP

Parameter Name	Phi
Description	Ellipsoid angle
Default Value	0
Parameter Range	not limited
Mode of Operation	Adjust SOP

References

none

7.7.9 PDL Element

Fundamentals

The PDL Element is a component with a polarization dependent power transmission T. The least attenuated state of polarization (SOP) has a transmission coefficient $T = T_{max}$, while the SOP that suffers the largest attenuation of all SOPs has a transmission coefficient $T = T_{min}$.
The parameters “PDL [dB]”, “PDL Gamma”, and the differential loss coefficient α are defined as

$$\mathrm{PDL[dB]} = 10\log_{10}\left(\frac{T_{\mathrm{max}}}{T_{\mathrm{min}}}\right), \tag{1}$$

$$\Gamma = \frac{T_{\mathrm{max}} - T_{\mathrm{min}}}{T_{\mathrm{max}} + T_{\mathrm{min}}}, \tag{2}$$

$$\frac{T_{\mathrm{max}}}{T_{\mathrm{min}}} = e^{2\alpha} \tag{3}$$

You may either set the direction of the least attenuated SOP in Stokes space defined by (e_1, e_2, e_3) or let PHOTOSS choose a random direction.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	Tmax
Description	Max. power transmission coefficient
Default Value	1
Parameter Range	0 < Tmin ≤ Tmax
Unit	normalized

Parameter Name	Tmin
Description	Min. power transmission coefficient
Default Value	1
Parameter Range	0 < Tmin ≤ Tmax
Unit	normalized

Parameter Name	PDL[dB]
Description	PDL in dB
Default Value	1
Parameter Range	0 < PDL[dB]
Unit	dB

Parameter Name	Gamma
Description	PDL Γ
Default Value	0.0230218
Parameter Range	0 ≤ Gamma <1
Unit	dB

Parameter Name	alpha
Description	Differential loss coefficient α
Default Value	0.0230259
Parameter Range	0 ≤ alpha

Parameter Name	Tdepol
Description	Transmission coefficient for depolarized light
Default Value	0.977496
Parameter Range	0 < Tdepol ≤1

Parameter Name	s1, s2, s3
Description	Cartesian coordinates of the least attenuated SOP state on the Poincaré sphere
Default Value	[1,0,0]
Parameter Range	-1 ≤ s_i ≤ 1
Unit	normalized

Parameter Name	2*Chi
Description	spherical azimuthal coordinate of the least attenuated SOP state on the Poincaré sphere
Default Value	0
Parameter Range	-90 ≤ 2*Chi ≤ 90
Unit	degree

Parameter Name	2*Psi
Description	spherical polar coordinate of the least attenuated SOP state on the Poincaré sphere
Default Value	0
Parameter Range	0 ≤ 2*Psi ≤ 360
Unit	degree

Parameter Name	Random direction
Description	Do you want PHOTOSS to choose a random direction of the PDL element in Stokes space by itself?
Default Value	false
Parameter Range	boolean

References

[1] N. Gisin, “Statistics of polarization dependent losses”, Opt. Commun., vol. 114, pp. 399-405, 1995

7.7.10 Transmission Matrix Analyzer

Fundamentals

The “Transmission Matrix Analyzer” (TMA) is a powerful tool to analyze the Jones-matrix (transmission matrix) of a concatenation of PMD (and PDL) components. (PMD + PDL emulators, birefringent elements, PDL elements, single mode fibers with random mode coupling, etc.). The TMA must also be present in you optical network if you want to use a Polarization Controller (PC) which should align the State of Polarization (SOP) of the Signal with the slow or fast Principal State of Polarization (PSP).

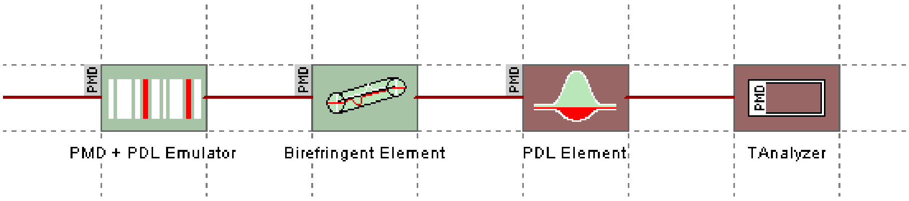


Figure 7.7.3. Path-Setup for a PMD Analysis

If your system consists of one or more of these components and you are interested e. g. in the frequency dependent DGD of the concatenation (or parts of it), you can set the “PMD-flag” of **each** PMD/PDL component to be considered in the calculation. The Transmission Matrix Analyzer takes the transmission matrices of the “PMD-flag-components” and calculates their overall transmission matrix:

$$\mathbf{T} = \mathbf{T}_N \cdots \mathbf{T}_2 \cdot \mathbf{T}_1 \tag{1}$$

Based on this frequency dependent matrix, the Transmission Matrix Analyzer is able to calculate several PMD and PDL quantities as a function of frequency, several higher order quantities and some autocorrelation functions.

The Transmission Matrix Analyzer generates files (one file for each selected AutoSave category) with these quantities as a function of frequency.

There may be situations you are not interested in the entire frequency dependence of the results of the analysis. In that case you can change the storage mode of the Transmission Matrix Analyzer e. g. to save only the quantities at the center frequency (of the pulse generator). Or you can choose to save them separated by one PSP-bandwidth (statistically dependent samples) or by six PSP-bandwidths (statistically independent samples).
(According to the standard PMD theory, the spectral distance of two statistically independent samples is roughly $6 \cdot PSP - bandwidth$.)

If you perform a Monte Carlo Simulation (or a parameter variation), PHOTOSS generates just one single file for each selected AutoSave category. For example, in case of the “6 * PSP-bandwidth” storage mode, PHOTOSS starts with the generation of a file with a subset of the values of the frequency domain. (This subset consists of samples which are 6 * PSP-bandwidth apart, so they can be considered to be statistically independent.)
Running the second and successive simulations of the Monte-Carlo Simulation, PHOTOSS appends the results to the same file(s).

As denoted below, the transmission matrix **T** of the concatenation of “PMD-flag-components” hast to be derived with respect to the angular frequency (which is equivalent to the **frequency slice** in the main parameters dialog) to obtain the PMD-vector. In order to obtain results which are numerically accurate, the frequency slice should be distinctly smaller than the bandwidth of the PSPs [4]. PHOTOSS automatically issues a warning in the TMA component if the bandwidth of the PSPs isn't at least **ten times** the frequency slice. You may want to adjust the frequency slice of your simulation accordingly. Generally, the bandwidth of the PSPs can be calculated by using

$$\Delta\omega = \frac{\pi}{4 \cdot \langle \Delta\tau \rangle}, \tag{2}$$

where $\Delta\tau$ is the mean DGD of the concatenation of the “PMD-flag-components” [4].

Notation

General:

$\mathbf{T}$: Transmission Matrix of the concatenation of “PMD-flag-components”: $\mathbf{T} = \mathbf{T}_N \cdots \mathbf{T}_2 \cdot \mathbf{T}_1$

$\mathbf{T^{-1}}$: Inverse matrix of $\mathbf{T}$

$\frac{\partial \mathbf{T}}{\partial \omega}$ Derivative of $\mathbf{T}$ with respect to the angular frequency ω

$\mathbf{T^0}$: Hermitian conjugate of $\mathbf{T}$

i : Imaginary unit

$\vec{\sigma} = \begin{pmatrix} \sigma_1 & \sigma_2 & \sigma_3 \end{pmatrix} = \left(\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \right)$ 2 x 2 Pauli spin matrices

PMD:

$\vec{\tau}$: (Complex) PMD-Vector: $2i\frac{\partial \mathbf{T}}{\partial \omega}\mathbf{T}^{-1} =: \vec{\tau} \cdot \vec{\sigma} = \tau_1\sigma_1 + \tau_2\sigma_2 + \tau_3\sigma_3$, $\vec{\tau} = \vec{\tau}_r + 1\vec{\tau}_i$

$\vec{\tau}_r = \tau_r\hat{\tau}_r$: Real part of the complex PMD-vector

$\vec{\tau}_i = \tau_i\hat{\tau}_i$: Imaginary part of the complex PMD-vector

$\sphericalangle\,(\hat{\tau}_r, \hat{\tau}_i)$: Angle in degrees between the real and the imaginary part of the complex PMD-vector in Stokes-space

$\Delta\tau$: (Complex) DGD: $\Delta\tau = \sqrt{\tau_1^2 + \tau_2^2 + \tau_3^2}, \quad \tau_i, \tau_i \in \mathbb{C}, \Delta\tau = DGD + iDAS, \quad DGD > 0$

DGD : Real part of the complex DGD

DAS : Imaginary part of the complex DGD: “Differential attenuation slope”

$|\pm\rangle_{\text{in/out}}$: Slow/Fast PSPs in Jones-space:

$$|\pm\rangle_{\text{out}} = \begin{pmatrix} \frac{\tau_1 \pm \sqrt{\tau_1^2 + \tau_2^2 + \tau_3^2}}{\tau_2 + 1\tau_3} \\ 1 \end{pmatrix} \tag{3}$$

$\hat{s}_\pm, \hat{t}_\pm$: Slow/Fast PSPs in Stokes-space (input, output)

$\sphericalangle\left(\hat{t}_g, \hat{t}_h\right)$: Angle in degrees between the output PSPs in Stokes-space

$\sphericalangle\left(\hat{s}_g, \hat{s}_h\right)$: Angle in degrees between the output PSPs in Stokes-space

PDL:

$[p_0, \vec{p}]$: Karlsson's "power transmission quaternion" (input): $\mathbf{T}^0\mathbf{T} =: p_0, +\vec{p} \cdot \vec{\sigma}$

T_{max} : Largest transmission coefficient: $T_{max} = p_0 + \left|\vec{p}\right|$

T_{min} : Smallest transmission coefficient: $T_{min} = p_0 - \left|\vec{p}\right|$

T_{depol} : Transmission coefficient of depolarized light: $T_{depol} = \frac{T_{max}+T_{min}}{2} = p_0$

$\mathrm{PDL_{dB}} := 10 \cdot \log_{10}\left(\frac{T_{max}}{T_{min}}\right) \mathrm{dB}$

$\Delta\alpha$ Differential loss coefficient: $T_{min} =: T_{max}e^{-2\Delta\alpha}$

$\vec{\Gamma}_{in}$: PDL-vector (input): $\vec{\Gamma}_{in} = \vec{p}/p_0$

Γ : PDL$\Gamma = \frac{T_{max}-T_{min}}{T_{max}+T_{min}} = \left|\vec{\Gamma}_{in}\right|, \vec{\Gamma}_{in} = \Gamma \cdot \hat{\Gamma}_{in}$

$\left[t_0, \vec{t}\right]$: "Power transmission quaternion" (output): $\mathbf{T}\mathbf{T}^0 =: t_0, +\vec{t} \cdot \vec{\sigma}$

$\vec{\Gamma}_{out}$: PDL-vector (output): $\vec{\Gamma}_{out} = \vec{t}\Big/t_0, \vec{\Gamma}_{out} = \Gamma \cdot \hat{\Gamma}_{out}$

AutoSave Notation:

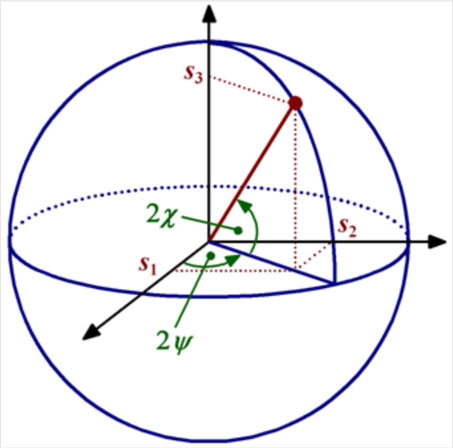


Figure 7.7.4. Poincaré-Sphere

PMD:

PMDvector :$\vec{\tau} = \vec{\tau}_r + i\vec{\tau}_i = \tau_r\hat{\tau}_r + i\tau_i\hat{\tau}_i$

τ_r : `|Re(PMD vector)| [ps]`

τ_i : `|Im(PMD vector)| [ps]`

$\hat{\tau}_r$: `2*Psi Re(PMD vector)[°], 2*Chi Re(PMD vector)[°]`

$\hat{\tau}_i$: `2*Psi Im(PMD vector)[°], 2*Chi Im(PMD vector)[°]`

$\sphericalangle(\hat{\tau}_r, \hat{\tau}_i)$: `Angle PMD vector Re <-> Im [°]`

$\vec{\tau}_r \cdot \vec{\tau}_i$: Re(PMD vector)* Im(PMD vector)[ps^2]

ComplexDGD :$\Delta\tau = DGD + iDAS$

DGD : DGD [ps]

DAS : DAS [ps]

PSPs:

$\hat{t}_g$: 2*Psi PSP+ out [°], 2*Chi PSP+ out [°]

$\hat{t}_h$: 2*Psi PSP- out [°], 2*Chi PSP- out [°]

$\hat{s}_g$: 2*Psi PSP+ in [°], 2*Chi PSP+ in [°]

$\hat{s}_h$: 2*Psi PSP- in [°], 2*Chi PSP- in [°]

$\sphericalangle\left(\hat{t}_g, \hat{t}_h\right)$: Angle PSPs out [°]

$\sphericalangle\left(\hat{s}_g, \hat{s}_h\right)$: Angle PSPs in [°]

PDL:

T_{max} : Tmax
T_{min} : Tmin
T_{depol} : Tdepol
PDL_{dB} : PDLdB [dB]
$\Delta\alpha$: alpha
PDLvector :$\vec{\Gamma}_{in/out} = \Gamma \cdot \hat{\Gamma}_{in/out}$
Γ : Gamma
$\hat{\Gamma}_{in}$: 2*Psi PDL vector in [°], 2*Chi PDL vector in [°]
$\hat{\Gamma}_{out}$: 2*Psi PDL vector out [°], 2*Chi PDL vector out [°]

Higher Order PMD:

SecondOrderPMDvector :$\vec{\tau}_\omega = \frac{\partial}{\partial\omega}\vec{\tau}_r + 1\frac{\partial}{\partial\omega}\vec{\tau}_i$

$\frac{\partial}{\partial\omega}\vec{\tau}_r = \left(\frac{\partial}{\partial\omega}\tau_r\right) \cdot \hat{\tau}_r + \tau_r \cdot \frac{\partial}{\partial\omega}\hat{\tau}_r$

$\frac{\partial}{\partial\omega}\vec{\tau}_i = \left(\frac{\partial}{\partial\omega}\tau_i\right) \cdot \hat{\tau}_i + \tau_i \cdot \frac{\partial}{\partial\omega}\hat{\tau}_i$

$\left|\frac{\partial}{\partial\omega}\vec{\tau}_r\right|$: |Re(SOPMD vector)| [ps^2]

$\left|\frac{\partial}{\partial\omega}\vec{\tau}_i\right|$: |Im(SOPMD vector)| [ps^2]

$\frac{\partial}{\partial\omega}\tau_r$: d/dw |Re(PMD vector)| [ps^2] - the real part of the PCD

$\frac{\partial}{\partial\omega}\tau_i$: d/dw |Im(PMD vector)| [ps^2] - the imaginary part of the PCD

$\left|\frac{\partial}{\partial\omega}\hat{\tau}_r\right|$: |d/dw Normalized Re(PMD vector)| [ps]

$\left|\frac{\partial}{\partial\omega}\hat{\tau}_i\right|$: |d/dw Normalized Im(PMD vector)| [ps]

$\tau_r \left| \frac{\partial}{\partial\omega} \hat{\tau}_r \right|$: |Re(PMD vector)| * |d/dw Normalized Re(PMD vector)| [ps^2] - the real part of the depolarization.

$\tau_i \left| \frac{\partial}{\partial\omega} \hat{\tau}_i \right|$: |Im(PMD vector)| * |d/dw Normalized Im(PMD vector)| [ps^2] - the imaginary part of the depolarization.

$\frac{\partial}{\partial\omega} \sphericalangle (\hat{\tau}_r, \hat{\tau}_i)$: d/dw (Angle PMD vector Re <-> Im)[° ps]

$\frac{\partial}{\partial\omega} (\vec{\tau}_r \cdot \vec{\tau}_i)$: d/dw (Re(PMD vector)* Im(PMD vector))[ps^2]

$\frac{\partial}{\partial\omega}$ComplexDGD

$\frac{\partial}{\partial\omega} DGD$: d/dw DGD [ps^2]

$\frac{\partial}{\partial\omega} DAS$: d/dw DAS [ps^2]

$\frac{\partial}{\partial\omega}$PSPs

$\left| \frac{\partial}{\partial\omega} \hat{t}_g \right|$: |d/dw PSP+ out| [ps]

$\left| \frac{\partial}{\partial\omega} \hat{t}_h \right|$: |d/dw PSP- out| [ps]

$\frac{\partial}{\partial\omega} \sphericalangle \left(\hat{t}_g, \hat{t}_h \right)$ d/dw Angle PSPs out [° ps]

$\frac{\partial}{\partial\omega} \sphericalangle \left(\hat{s}_g, \hat{s}_h \right)$ d/dw Angle PSPs out [° ps]

$\frac{\partial}{\partial\omega} \left(DGD \cdot \hat{t}_g \right) = \left(\frac{\partial}{\partial\omega} DGD \right) \cdot \hat{t}_g + DGD \cdot \frac{\partial}{\partial\omega} \hat{t}_g$

$\left| \frac{\partial}{\partial\omega} \left(DGD \cdot \hat{t}_g \right) \right|$ |d/dw (DGD * PSP+ out)|

$\frac{\partial}{\partial\omega} DGD$: see above

$\left| \frac{\partial}{\partial\omega} \hat{t}_g \right|$: see above

$DGD \cdot \left| \frac{\partial}{\partial\omega} \hat{t}_g \right|$: DGD * |d/dw PSP+ out|

Higher Order PDL:

Higher Order PDL:

$\frac{\partial}{\partial\omega} T_{max}$: d/dw Tmax [ps]

$\frac{\partial}{\partial\omega} T_{min}$: d/dw Tmin [ps]

$\frac{\partial}{\partial\omega} T_{depol}$: d/dw Tdepol [ps]

$\frac{\partial}{\partial\omega} PDL_{dB}$: d/dw PDLdB [dB ps]

SecondOrderPDLvector : $\frac{\partial}{\partial\omega} \vec{\Gamma}_{out} = \left(\frac{\partial}{\partial\omega} \Gamma \right) \cdot \hat{\Gamma}_{out} + \Gamma \cdot \frac{\partial}{\partial\omega} \hat{\Gamma}_{out}$

$\left| \frac{\partial}{\partial\omega} \vec{\Gamma}_{out} \right|$: |SOPDL vector| [ps]

$\frac{\partial}{\partial\omega} \Gamma$: d/dw Gamma [ps]

$\left|\frac{\partial}{\partial\omega}\hat{\Gamma}_{out}\right|$: `|d/dw Normalized PDL vector| [ps]`

$\Gamma\cdot\left|\frac{\partial}{\partial\omega}\hat{\Gamma}_{out}\right|$ `Gamma * |d/dw Normalized PDL vector| [ps]`

Autocorrelation functions:

$\mathrm{ACF}\left(\vec{\tau}_r\right)$ `ACF Re(PMD vector)[ps^2]`
$\mathrm{ACF}\left(\vec{\tau}_i\right)$ `ACF Im(PMD vector)[ps^2]`
$\mathrm{ACF}\left(\tau_r\right)$ `ACF |Re(PMD vector)| [ps^2]`
$\mathrm{ACF}\left(\tau_i\right)$ `ACF |Im(PMD vector)| [ps^2]`
$\mathrm{ACF}(\sphericalangle\hat{\tau}_r,\hat{\tau}_i)$ `ACF Angle PMD vector Re <-> Im [°°]`
$\mathrm{ACF}\left(\vec{\tau}_r\cdot\vec{\tau}_i\right)$ `ACF Re(PMD vector)* Im(PMD vector)[ps^4]`
$\mathrm{ACF}\left(DGD\right)$ `ACF DGD [ps^2]`
$\mathrm{ACF}\left(DAS\right)$ `ACF DAS [ps^2]`
$\mathrm{ACF}\left(\hat{t}_g\right)$ `ACF PSP+`
$\mathrm{ACF}\left(\hat{t}_h\right)$ `ACF PSP-`
$\mathrm{ACF}(\sphericalangle\hat{t}_r,\hat{t}_i)$ `ACF Angle PSPs [°°]`
$\mathrm{ACF}\left(T_{max}\right)$ `ACF Tmax`
$\mathrm{ACF}\left(T_{min}\right)$ `ACF Tmin`
$\mathrm{ACF}\left(T_{depol}\right)$ `ACF Tdepol`
$\mathrm{ACF}\left(\Gamma\right)$ `ACF Gamma`
$\mathrm{ACF}\left(PDL_{dB}\right)$ `ACF PDLdB`
$\mathrm{ACF}\left(\vec{\Gamma}_{out}\right)$ `ACF PDL vector`

Input / Output

	Number	Type of signal
Input	1	Optical field
Output		no output

References

[1] M. Karlsson, M. Petersson, “Quaternion Approach to PMD and PDL Phenomena in Optical Fiber Systems”, Journal of Lightwave Technology, vol. 22, No. 4, pp. 1137-1146, 2004

[2] N. Gisin, B. Huttner, “Combined effects of polarization mode dispersion and polarization dependent losses in optical fibers”, Optics Communications, vol. 142, pp. 119-125, 1997

[3] B. Huttner, C. Geiser, N. Gisin, “Polarization-Induced Distortions in Optical Fiber Networks with Polarization-Mode Dispersion and Polarization-Dependent Losses”, IEEE Journal Of Selected Topics In Quantum Electronics, vol. 6, pp. 317-329, 2000

[4] A. Steinkamp, “Wechselwirkung von Polarisationsmodendispersion und polarisationsabhängiger Dämpfung in faseroptischen Systemen”, Dissertation am Lehrstuhl für Hochfrequenztechnik der TU Dortmund, 07.05.2007.

7.8 Receivers

7.8.1 Bit Error Rate Tester (analytical)

Fundamentals

The analytical bit error rate tester (BERT) does not contain a receiver; it is placed behind a receiver containing a diode and an electrical filter. Thermal noise can be added at the filter input and is treated analytically for the BER calculation.
The algorithms for determining the BER are described in what follows.
The average BER of a binary signal is defined by

$$P_e = P_0 P_{e0} + P_1 P_{e1} = P_0 P(X_0 > S) + P_1 P(X_1 < S) \tag{1}$$

The random variables X_0 and X_1 are the sampled values for the logical “0” and the logical “1” at the decision device. P_0 and P_1 are the probabilities of the symbols, P_{e0} and P_{e1} the corresponding error rates and S is the threshold. With the probability densities $p_{x0}(x_0)$ and $p_{x1}(x_1)$ for the logical “0” and the logical “1” eq. 1) can be written as

$$P_e = P_0 \int_S^{\infty} p_{x0}(x_0)\, dx + P_1 \int_{-\infty}^{S} p_{x1}(x_1)\, dx \tag{2}$$

Assuming Gaussian noise with the standard deviations σ_{N0} and σ_{N1} the BER is given by

$$P_e = P_0 E\left[\frac{1}{2} erfc\left(\frac{S - X_0}{\sqrt{2}\sigma_{N0}}\right)\right] + P_1 E\left[\frac{1}{2} erfc\left(\frac{X_1 - S}{\sqrt{2}\sigma_{N1}}\right)\right] \tag{3}$$

“E[ ]” denotes the expected value of the complementary error function of the random variables X_0 and X_1. These random variable are now the sampled values without the noise described by σ_{N0} and σ_{N1}. The total BER can be calculated from

$$P_e = \frac{M}{N+M} P_{e0} + \frac{N}{N+M} P_{e1} \tag{4}$$

where M is the number of samples for the logical “0” and N is the number of samples for the logical “1” respectively.
For estimating correct BER with the AnalyticalBert, the parameter *noise handling* in the *simulation parameters* dialog, has to be switched to *analytical.*
The analytic approach evaluates the BER by taking a noiseless signal. Therefore no component has to include the noise numerically. The BER is evaluated for each single detected value, whereby the complete BER is calculated by averaging the single BERs.

$$P_e = \frac{1}{N+M}\left(\sum_{i=0}^{M} \frac{1}{2} erfc\left(\frac{X_{1,i} - S}{\sqrt{2}\sigma_{1,i}}\right) + \sum_{i=0}^{N} \frac{1}{2} erfc\left(\frac{S - X_{0,i}}{\sqrt{2}\sigma_{0,i}}\right)\right) \tag{5}$$

The component-induced noise, i. e. the ASE-noise of optical amplifiers is described separately from the signals by a roughly sampled broadband noise spectrum. The noise is furthermore assumed to be Gaussian, so that the following approximation can be applied:

$$\sigma^2 = \sigma_S^2 + \sigma_{ASE-ASE}^2 + \sigma_{Sig-ASE}^2 + \sigma_{Shot-ASE}^2 + \sigma_{th}^2 \tag{6}$$

Shot-Noise, thermal noise and beat noise contributions are covered. The assumption of Gaussian PDFs lead to acceptable results, if the effective bandwidth B_{opt} of the ASE is considerably larger than the single-sided electrical bandwidth B_{el} of the receiver filter and the threshold is optimized [1]. The several noise terms are explained in [1], [2].

For the shot noise, we have:

$$\sigma_S^2 = 2q\left[R\left(X_{,i} + P_{ASE}\right) + I_d\right] B_{el} \tag{7}$$

The part of the dark current I_d is neglected during the calculation of shot noise. The contributions of channel power X_j and ASE power P_{ASE} can be included separately.
The ASE-Beat noise and the ASE-signal beat noise can be formulated as

$$\sigma_{ASE-ASE}^2 = 4R^2 X_{,i}^2 \frac{B_{el}}{B_{opt}} \tag{8}$$

and

$$\sigma_{Sig-ASE}^2 = 4R^2 X_{,i} P_{ASE} \frac{B_{el}}{B_{opt}}, \tag{9}$$

respectively. The ASE-Shot beat niose is

$$\sigma_{Shot-ASE}^2 = 4qRP_{ASE}B_{el} \tag{10}$$

Due to the negligible value of ASE-shot beat noise this term is not implemented in PHOTOSS. The thermal noise can be written as:

$$\sigma_{th}^2 = N_0 B_{el} \tag{11}$$

P_{ASE} denotes so ASE-power of one polarization plane, q is the elementary charge and R the responsivity of the diode.
A direct analytical calculation of the PDFs for monochromatic signals with narrowband Gaussian noise can be found in [1], [2]. The integrated intensity is (assuming rectangular filters and polarized ASE) non-central chi-squared distributed. To include shot noise, this distribution has to be Poisson-transformed resulting in a Laguerre distribution. In combination with thermal noise, a closed solution cannot be obtained.

Input / Output

	Number	Type of signal
Input	1	Electrical signal
Output		no output

Component Parameters

Parameter Name	Store bit lines for constellation diagram
Description	Determines, whether the signal of all simulated blocks is stored for the generation of constellation diagrams.
Default Value	true
Parameter Range	boolean

The Analytical BERT also contains an Eye Analyzer and its parameters (see section 7.8.3 on page 288 for a detailed description). The parameters of the Analytic Performance are described in section 4.1 on page 110.

Results

Parameter Name	Ase_ase noise variance
Description	Variance of the ASE-ASE noise

Parameter Name	Ase channel noise variance
Description	Variance of the ASE channel noise

Parameter Name	Channel shot noise variance
Description	Variance of the channel shot noise

Parameter Name	Variance of the thermal noise
Description	Variance of the thermal noise

Parameter Name	All lines BER
Description	Bit error rate, regarding all lines

Parameter Name	All lines Q
Description	Q factor regarding all lines

Parameter Name	All lines, power4berm9
Description	The power, which is necessary to reach a BER of 10^{-9} regarding all lines

Parameter Name	All lines, power4berm12
Description	The power, which is necessary to reach a BER of 10^{-12} regarding all lines

Parameter Name	All lines, OSNR at BER
Description	The required OSNR value for the given BER regarding all lines

Parameter Name	Box lines BER
Description	Bit error rate, regarding the box lines

Parameter Name	Box lines Q
Description	Q factor regarding the box lines

Parameter Name	Box lines, power4berm9
Description	The power, which is necessary to reach a BER of 10^{-9} regarding the box lines

Parameter Name	Box lines, power4berm12
Description	The power, which is necessary to reach a BER of 10^{-12} regarding the box lines

Parameter Name	Box lines, OSNR at BER
Description	The required OSNR value for the given BER regarding the box lines

Parameter Name	Worst case lines BER
Description	Bit error rate, regarding worst case lines

Parameter Name	Worst case lines Q
Description	Q factor, regarding worst case lines

Parameter Name	Worst case lines, power4berm9
Description	The power, which is necessary to reach a BER of 10^{-9} regarding worst case lines

Parameter Name	Worst case lines, power4berm12
Description	The power, which is necessary to reach a BER of 10^{-12} regarding worst case lines

Parameter Name	Worst case lines, OSNR at BER
Description	The required OSNR value for the given BER regarding worst case lines

Parameter Name	Eye opening penalty (inner eye height)
Description	Vertical inner eye height penalty (based on the inner eye lines)

The Analytical BERT also contains the results of an Eye Analyzer. Please refer to section 7.8.3 on page 291 for further details on the results of the Eye Analyzer.

Graphs

The Analytical BERT contains the following Visualizers:

1. BER and Q factor versus power (on worst case detection)
2. Eye pattern diagram
3. Constellation diagram
4. Histogram of the timing jitter
5. (based on the centroid of the marks)
6. Histogram of the timing jitter
7. (maximum mark value based)
8. Histogram of the ones in the maximum
9. Histogram of the zeros at center time

References

[1] G.-P. Agrawal: “Fibre Optic Communication Systems”, John Willey & Sons, 1992

[2] E. Desurvire: “Erbium-Doped Fiber Amplifiers”, John Willey & Sons, 1994

7.8.2 Avalanche Photodiode Fundamentals

In contrast to a PIN photodiode, an avalanche photo diode (APD) amplifies the generated photocurrent internally due to the avalanche effect and thus shows an improved sensitivity but also a higher quantum noise.

Whereas the number of primarily generated electrons can be described by the Poisson PDF given in the previous section, a complete model must include secondary generated electrons and the excess noise due to the avalanche process.

To a good approximation, Webb, McIntyre and Conradi (WMC) probability density functions (PDF) can be used to describe the photo-electron statistics [1]

$$f_{apd}\left(n_e\right) = \frac{1}{\sqrt{2\pi}\left(1+\frac{x}{\delta}\right)^{3/2}} \exp\left\{\frac{-x^2}{2\left(1+\frac{x}{\delta}\right)}\right\} \tag{1}$$

where

$$-\sqrt{\frac{\langle n_e \rangle}{F_e}} \leq x \leq \infty \tag{2}$$

The variables x,δ , σ^2 and F_e are defined as

$$x = \frac{(n_e - \langle n_e \rangle M_0)}{\sigma}, \tag{3}$$

$$\delta = \frac{\sqrt{\langle n_e \rangle F_e}}{F_e - 1} \tag{4}$$

$$\sigma^2 = \langle n_e \rangle {M_0}^2 F_e, \tag{5}$$

$$F_e = kM_0 + (2 - \frac{1}{M_0})(1-k). \tag{6}$$

k :	Ionization ratio
M_0 :	Avalanche factor
$\langle n_e \rangle$:	Average number of generated carriers

The mean of the photo current is calculated using eqs. (1)-(3) of the “PIN Photodiode” section.

If separated channels is chosen as the method in the simulation parameters-dialog, the power of all signal vectors is summarized to calculate the output current.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Electrical current

Component Parameters

Parameter Name	eta
Description	Quantum efficiency (η)
Default Value	0.88
Parameter Range	0 ≤ eta ≤ 1

Parameter Name	G
Description	Average avalanche gain
Default Value	20
Parameter Range	0 ≤ G ≤ 1e3

Parameter Name	k
Description	Ionization ratio
Default Value	0.1
Parameter Range	0 ≤ k ≤ 1

Parameter Name	Id
Description	Dark current
Default Value	1e-9
Parameter Range	0 ≤ Id ≤ 1e-3

References

[1] Jeruchim, Balaban, Shanmugan: Simulation of Communication Systems, Plenum Press, New York / London, 1994

7.8.3 Eye Analyzer

Fundamentals

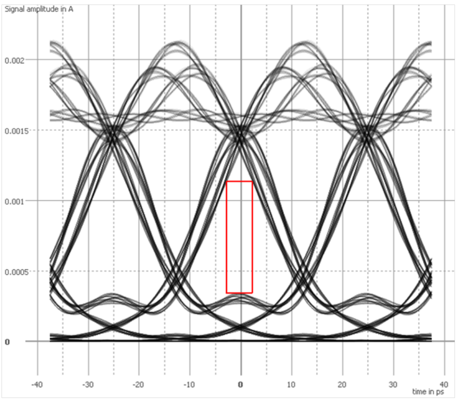


Figure 7.8.1. Eye Diagram of the Eye Analyzer

Using the eye analyzer, an eye pattern of optical or electrical signals can be obtained. The data is visualized in units of $\sqrt{W}$ (optical signal) or V and A for electrical signals, respectively.
An eye analysis is performed where the optimum detection time is evaluated. Additionally, a box is set into the eye diagram. The *search range* defines the range in which the analysis algorithm searches the maximum eye opening. The *number of bits for delay detection* defines the number of bits which are evaluated for finding the optimum detection time. *Number of bits for eye analysis* specifies the number of bits which are collected for the determination of the detected signal values. The optimum detection time is found by fitting a rectangular box in the eye diagram with maximum height. The width of this box is related to the bit time and is chosen in the edit box *Relative box width*. The sampling of the Eye Analyzer is done independently to the definitions in the *simulation parameters* dialog. The number of samples per bit-cell can be defined by Samples per bit. Using the parameter *Ignore first* bits a certain number of bits are ignored at the beginning of the signal of the Eye Analyzer if the convolution method is *cyclic*.
If a huge amount of bits are to be simulated, the eye analysis is very memory consuming. To overcome this problem, the used may switch on the *lean mode*. Here only the samples at the optimum bit time are stored and can be written to file. One results of the Eye Analyzer is the eye opening (EO). The EO is calculated by the height of the box, and additionally by the vertical difference between the worst case mark and space-line, eye opening (inner eye height). With the average value of the signal $\langle s(t) \rangle$, the eye opening penalty (EOP) is roughly estimated by

$$\mathrm{EOP} = \frac{2 \cdot \langle s\,(t) \rangle}{\mathrm{EO}} \tag{1}$$

The value of $2 \cdot \langle s\,(t) \rangle$ denotes the eye opening of an undistorted eye pattern. The EOP is calculate from the EO as well as on the EO (worst case lines). A more accurate way to derive the EOP is to measure the EO directly behind the emitter (back to back).
Additionally, the arrival time of the marks is analyzed. The arrival times are defined as the maximums of the mark-bit-lines. Average value and standard deviation (mue_t and sig_t) are calculated. The standard deviation is commonly called timing jitter.
Furthermore, the detection value of the maximums of the marks are taken to calculated average value and standard deviation (mue_1 and sig_1). The spaces are taken at the eye center (defined by mue_t), and average value and standard deviation (mue_0 and sig_0) are analyzed.

Input / Output

	Number	Type of signal
Input	1	Optical field / Electrical signal
Output		no output

Component Parameters

Parameter Name	Determine delay
Description	the delay is determined by the user (false) or by PHOTOSS (true)
Default Value	true
Parameter Range	boolean

Parameter Name	search range
Description	Search range for the delay detection
Default Value	2
Parameter Range	1 ≤ search range ≤ Number of Bits per Block

Parameter Name	Number of bits for delay detection
Description	Maximum number of bits for delay detection
Default Value	50
Parameter Range	1 ≤ Number of bits ≤ Number of Bits per Block

Parameter Name	User defined delay
Description	Dynamic delay that shall be used for the eye analysis
Default Value	0
Parameter Range	-Block Time ≤ delay ≤ +Block Time
Unit	ps

Parameter Name	Use all bits for eye analysis
Description	number of bits for eye analysis is set to the number of simulated bit (true) or The value of 'Number of bits for eye analysis' is used (false)
Default Value	true
Parameter Range	boolean

Parameter Name	Number of bits for eye analysis
Description	Number of bits for the eye analysis
Default Value	Number of Bits per Block
Parameter Range	bits ≤ Number of Bits per Block

Parameter Name	Relative box width
Description	Relative width of the inner eye box (relative to the bit time)
Default Value	0.2
Parameter Range	1e-6 ≤ Relative box width ≤ Bit Time

Parameter Name	Samples per bit
Description	Number of samples per bit
Default Value	101
Parameter Range	1 ≤ Samples per bit

Parameter Name	Ignore first bits
Description	Number of bits, which should be ignored
Default Value	0
Parameter Range	Ignore first bits < Number of Bits per Block

Parameter Name	Force optical eye
Description	If the Eye Analyzer is placed in the optical domain, the eye analysis is done without an OE-Conversion (true) or the eye is always evaluated in the electrical domain (false)
Default Value	false
Parameter Range	boolean

Parameter Name	Lean Mode
Description	the eye analysis only saves the optimal samples. The rest of the Bitlines will be lost. Results like EO, EOP, are not calculated (true) or the full eye analysis is performed. All results are available (false).
Default Value	false
Parameter Range	boolean

Parameter Name	Detect opt sample
Description	The eye analysis determines the optimal sample time within the eye (true) or the user determines the sample time with regard to the center of the bit (0 ps corresponds to the center of the bit).
Default Value	true
Parameter Range	boolean

Parameter Name	User defined optimal sample
Description	Optimal sample time that shall be used for the eye analysis. 0 ps = center of bit, neg values = left side of bit, pos values = right side of bit.
Default Value	0
Parameter Range	-Block Time ≤ Optimal Sample Time ≤ + Block Time
Unit	ps

Parameter Name	Save variances
Description	Saves the variances of the marks and the spaces into a seperatedfile
Default Value	false
Parameter Range	boolean

Parameter Name	Save pattern and samples
Description	Saves the pattern and the detected samples to a file
Default Value	false
Parameter Range	boolean

Parameter Name	Filename, pattern and samples
Description	Filename for the storing of the pattern and samples
Default Value	none
Parameter Range	valid file path

Parameter Name	Save all samples
Description	Determines, whether all samples of the eye shall be stored in the file or just a fraction
Default Value	false
Parameter Range	boolean

Parameter Name	Number of samples
Description	Determines how many samples per bit shall be stored
Default Value	1
Parameter Range	1 ≤ Samples ≤ Samples Per Bit

Parameter Name	Sampling shift
Description	Determines the time mismatch with regard to the optimal sampling of the eye.
Default Value	0
Parameter Range	not implemented
Unit	ps

Parameter Name	Save zero and one values
Description	Saves the zero and one values, which has been detected at the optimum sample time, in two separated files. The first line in each file is a comment, marked by '%'.
Default Value	false
Parameter Range	boolean

Parameter Name	Store bit lines for constellation diagram
Description	Determines, whether the signal of all simulated blocks is stored for the generation of constellation diagrams.
Default Value	true
Parameter Range	boolean

Results

Parameter Name	Eye opening
Description	Vertical eye width (height of the inner box)

Parameter Name	Eye opening penalty
Description	Vertical eye width penalty (based on the height of the inner box)

Parameter Name	Extinction
Description	Vertical eye width extinction (based on the height of the inner box)

Parameter Name	Mark value
Description	Mark value (based on the inner box)

Parameter Name	Space value
Description	Space value (based on the inner box)

Parameter Name	Eye opening (inner eye height)
Description	Vertical eye width (based on the inner eye lines)

Parameter Name	Eye opening penalty (inner eye height)
Description	Vertical inner eye height penalty (based on the inner eye lines)

Parameter Name	Extinction (inner eye height)
Description	Vertical eye width extinction (based on the inner eye lines)

Parameter Name	Mark value (inner eye)
Description	Mark value (based on the inner eye)

Parameter Name	Space value (inner eye)
Description	Space value (based on the inner eye)

Parameter Name	Mean value
Description	Mean value of the bitlines

Parameter Name	BoxCenterTime
Description	Center time of the inner eye box

Parameter Name	BoxCenterValue
Description	Center value of the inner eye box

Parameter Name	mue_t
Description	Average arrival time of the ones

Parameter Name	sig_t
Description	Standard deviation of the arrival time (timing jitter)

Parameter Name	mue_t_centroid
Description	Average arrival time of the ones (centroid based method)

Parameter Name	sig_t_centroid
Description	Standard deviation of the arrival time time (timing jitter) (centroid based method)

Parameter Name	mue_1
Description	Average value of the ones in the maximum

Parameter Name	Q_1
Description	Average value of the ones divided by standard deviation in the maximum

Parameter Name	exc_1
Description	Maximal excursion of the ones in the maximum

Parameter Name	sig_1
Description	Standard deviation of the ones in the maximum

Parameter Name	mue_0
Description	Average value of the zeros at the eye center

Parameter Name	sig_0
Description	Standard deviation of the zeros at the eye center

Graphs

The Eye Analyzer contains the following Visualizers:

1. Eye pattern diagram
2. Constellation diagram
3. Histogram of the timing jitter (based on the centroid of the marks)
4. Histogram of the timing jitter (maximum mark value based)
5. Histogram of the ones in the maximum
6. Histogram of the zeros at center time
7. Save Null Values (right-click after simulation)
8. Save One Values (right-click after simulation)
9. Save pattern and samples to file (right-click after simulation)

References

none

7.8.4 Bit Error Rate Tester (numerical)

Fundamentals

The numerical bit error rate tester (BERT) is a component for determining the bit error rate of a system. Two algorithms for BER estimation are implemented, the *Monte-Carlo* algorithm and the *tail extrapolation*. The Monte-Carlo method simply counts detection errors. In order to detect BER below 10^{-9}, one has to process 10^{11} bits to find 100 errors. As this would be extremely time consuming, the tail extrapolation method allows a BER approximation based on significantly fewer bits.
For estimating correct BER with the numerical BERT, the parameter *noise handling* in the simulation parameters dialog, has to be switched to *numerical*.
Several switches allow storing information about the analyzing procedure in a file.

The algorithms for determining the BER are described in what follows.
The average BER of a binary signal is defined by

$$P_e = P_0 P_{e0} + P_1 P_{e1} = P_0 P(X_0 > S) + P_1 P(X_1 < S) \tag{1}$$

The random variables X_0 and X_1 are the sampled values for the logical “0” and the logical “1” at the decision device. P_0 and P_1 are the probabilities of the symbols, P_{e0} and P_{e1} the corresponding error rates and S is the threshold. With the probability densities $p_{x0}(x_0)$ and $p_{x1}(x_1)$ for the logical “0” and the logical “1” eq. (1) can be written as

$$P_e = P_0 \int_S^{\infty} p_{x0}(x_0)\, dx + P_1 \int_{-\infty}^{S} p_{x1}(x_1)\, dx \tag{2}$$

Assuming Gaussian noise with the standard deviations σ_{N0} and σ_{N1} the BER is given by

$$P_e = P_0 E\left[\frac{1}{2} erfc\left(\frac{S - X_0}{\sqrt{2}\sigma_{N0}}\right)\right] + P_1 E\left[\frac{1}{2} erfc\left(\frac{X_1 - S}{\sqrt{2}\sigma_{N1}}\right)\right] \tag{3}$$

“E[ ]” denotes the expected value of the complementary error function of the random variables X_0 and X_1. These random variable are now the sampled values without the noise described by σ_{N0} and σ_{N1}. The total BER can be calculated from

$$P_e = \frac{M}{N + M} P_{e0} + \frac{N}{N + M} P_{e1} \tag{4}$$

where M is the number of samples for the logical “0” and N is the number of samples for the logical “1” respectively.
For estimating the correct BER all noise sources in the signal path have to be described numerically.

Tail Extrapolation

The idea behind tail extrapolation is to calculate the actual BER by extrapolation using higher BER values obtained at modified decision thresholds [1,2]. The tails of the PDFs of the sampled values are assumed to have a so-called generalized exponential shape

$$P_{x0}(x0) = \frac{\delta}{\Gamma(1/\delta)\, 2\sqrt{2}\varepsilon} \exp\left(-\left|\left(x - m_{x0}^{(1)}\right)/\sqrt{2}\varepsilon_0\right|^{\delta_0}\right) \tag{5}$$

with the Gamma-function Γ and the fitting parameters δ_0 and ϵ_0. The BER is then given by integration

$$P_{e0} = \int_S^{\infty} p_{x0}(x0)\, dx0 \approx \exp\left(-\left[\frac{S - m_{x0}^{(1)}}{\sqrt{2}\varepsilon_0}\right]^{\delta_0}\right) \tag{6}$$

Equation (5) may be expressed as

$$\ln\left(-\ln P_{e0}\right) = \delta_0 \ln\left(\frac{S - m_{x0}^{(1)}}{\sqrt{2}\varepsilon_0}\right) \tag{7}$$

which is an equation of the form

$$y_0 = a x_0 + b \tag{8}$$

where $y_0 = \ln\left(-\ln P_{e0}\right)$, $x_0 = \ln\left(S - m_{x0}^{(1)}\right)$, $a = \delta_1$,$b = -\delta_0 \ln\left(\sqrt{2}\varepsilon_0\right)$.

Applying the same procedure tor the logical “1” results in

$$y_1 = c x_1 + d \tag{9}$$

with $y_1 = \ln\left(-\ln P_{e1}\right)$, $x_1 = \ln\left(m_{x1}^{(1)} - S\right)$, $c = \delta_1$, $d = -\delta_1 \ln\left(\sqrt{2}\varepsilon_1\right)$

The fitting parameters of these equations are found by a least square fit of the calculated BERs from different so-called pseudo-thresholds S' . Analytically described noise is included by

$$P_{e0,th} = \frac{1}{2} erfc\left(\frac{S' - X_0}{\sqrt{2}\sigma_{Ni}}\right), P_{e1,th} = \frac{1}{2} erfc\left(\frac{X_1 - S'}{\sqrt{2}\sigma_{Ni}}\right) \tag{10}$$

The optimum threshold (minimum BER) may be found from

$$\left(\frac{S_{opt} - m_{x0}^{(1)}}{\sqrt{2}\varepsilon_0}\right)^{\delta_0} = \left(\frac{m_{x1}^{(1)} - S_{opt}}{\sqrt{2}\varepsilon_1}\right)^{\delta_1} \tag{11}$$

Eye Grid

The ’Eye grid’ mode designed to calculate bit error ratios for large number of bits. In contrast to standard mode of operation, the amount of memory used for the calculation does not increase with the number of bits simulated. To achieve this kind of behavior, the bit lines of all simulated blocks are not stored and processed afterwards, but are processed after each block. Since an optimal sampling time and threshold cannot be calculated during the simulation, the bit error ratio is calculated for a number of sampling points within the eye. The sampling points are placed equidistantly in both dimensions of the eye. These sampling points create the eye grid. The number of sampling points along the time axis of the eye is equal to the number of samples per bit determined in the Eyeanalysis Parameters, while the number of sample thresholds may be determined in two ways

1. Only the number of thresholds is determined by the user. PHOTOSS itself will determine the minimum and maximum value.

2. The user determines the number, minimum and maximum value of the threshold.

PHOTOSS will calculate the BER values by determine the optimum column of the eye grid and use each line as a pseudo-thresholds S' as described above.
In order to give the user full control over the generated data, three files may be saved to disk:

1. ’Eye grid values file’: It contains a matrix of the occurred bit errors at the sampling grid points. The first line corresponds to the highest threshold used for the grid.

2. ’Eye grid threshold vector file’: The threshold vector used for the grid is stored in this file.

3. 'Eye grid additional info file': This file contains information on the number of simulated bits, simulated blocks, samples per bit and thresholds.

Due to the nature of this mode of operation, it is not possible to use the 'Eye grid mode' and the 'Lean mode' simultaneously.
With the information in this file the user may perform his own analysis of the data e.g. by means of MATLAB®. A graphical representation of the bit error ration (in 10^{-x}) of an exemplary eye grid may look like figure 7.8.2.

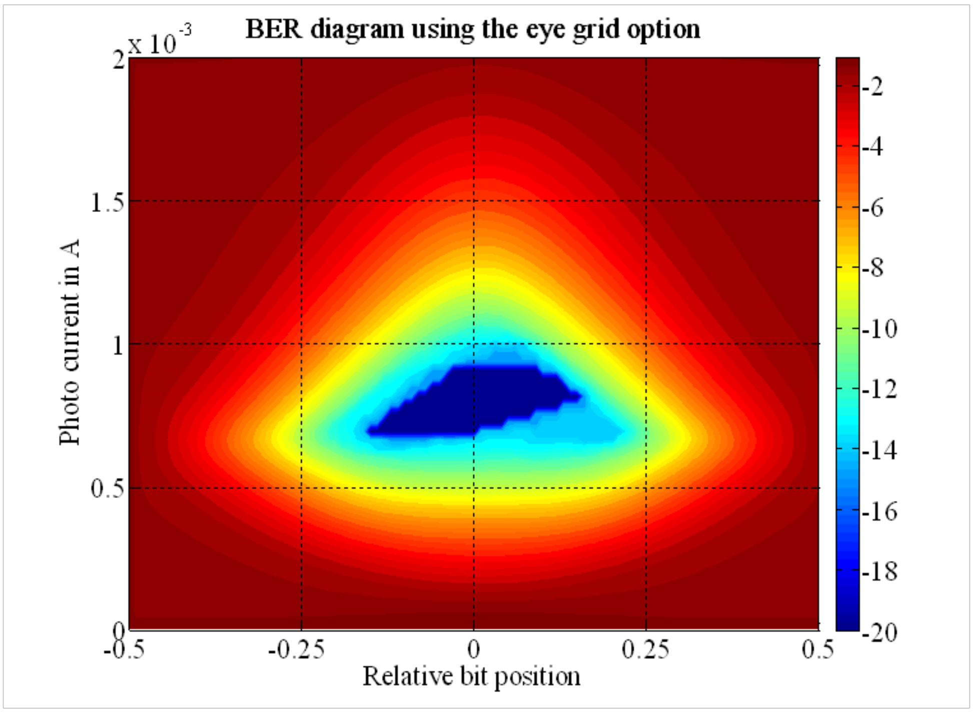


Figure 7.8.2. Eye Grid created with MATLAB®

Input / Output

	Number	Type of signal
Input	1	Electrical signal
Output		no output

Component Parameters

Parameter Name	Always_MC
Description	Always Monte-Carlo, no approximation
Default Value	false
Parameter Range	boolean

Parameter Name	Manual threshold determination
Description	Determines whether the threshold is optimized by PHOTOSS or determined by the user
Default Value	false
Parameter Range	boolean

Parameter Name	Manual threshold
Description	User defined threshold for BER calculations
Default Value	0
Parameter Range	not implemented
Unit	A

Parameter Name	Calculate distorted bit pattern
Description	Determines whether the bit pattern corresponding to the threshold shall be saved to a file
Default Value	false
Parameter Range	boolean

Parameter Name	Bit pattern file
Description	File to save the threshold related bit pattern
Default Value	none
Parameter Range	valid file path

Parameter Name	Store bit lines for constellation diagram
Description	Determines, whether the signal of all simulated blocks is stored for the generation of constellation diagrams.
Default Value	false
Parameter Range	boolean

Parameter Name	Use eye grid
Description	Use the eye grid mode.
Default Value	false
Parameter Range	boolean

Parameter Name	Number of thresholds
Description	Determines the number of threshold used for the eye grid.
Default Value	100
Parameter Range	not implemented

Parameter Name	Set thresholds manually
Description	Determines, whether the threshold for the eye grid is calculated automatically.
Default Value	false
Parameter Range	boolean

Parameter Name	Min. threshold
Description	Determines the minimum threshold of the eye grid, when setting the threshold manually
Default Value	0
Parameter Range	not implemented
Unit	A or W

Parameter Name	Max. threshold
Description	Determines the maximum threshold of the eye grid, when setting the threshold manually
Default Value	0.002
Parameter Range	not implemented
Unit	A or W

Parameter Name	Set sampling time manually
Description	Determines, whether the optimum sampling time for the eye grid is calculated automatically
Default Value	false
Parameter Range	boolean

Parameter Name	Sampling time
Description	Determines the optimum sampling time of the eye grid, when setting the sampling time manually
Default Value	0
Parameter Range	not implemented
Unit	ps

Parameter Name	Save eye grid to file
Description	Determines, whether the data of the eye grid is save to file
Default Value	false
Parameter Range	boolean

Parameter Name	Eye grid values file
Description	Filename for the bit error matrix
Default Value	none
Parameter Range	valid file path

Parameter Name	Eye grid threshold vector file
Description	Filename for the threshold vector
Default Value	none
Parameter Range	valid file path

Parameter Name	Eye grid additional info file
Description	Filename for the additional information on the eye grid
Default Value	none
Parameter Range	valid file path

The Numerical BERT also contains an Eye Analyzer. Please refer to section 7.8.3 on page 288 for further details on its parameters and settings.

Results

Parameter Name	Q-factor
Description	Equivalent Q-factor: $BER = 1/2 * erfc(Q/sqrt(2))$

Parameter Name	Bit Error Rate
Description	Bit error rate probability, in logarithmic units

Parameter Name	Simulated Bits
Description	Number of bits that were used to determine the BER

Parameter Name	Erroneous Bits
Description	Number of erroneous bits that were used to determine the BER (Always MC Mode only)

Parameter Name	Erroneous Bits
Description	Number of erroneous bits that were used to determine the BER (Always MC Mode only)

The Numerical BERT also contains the results of an Eye Analyzer. Please refer to section 7.8.3 on page 291 for further details on the results of the Eye Analyzer.

Graphs

The Numerical BERT contains the following Visualizers:

1. BER versus threshold
2. Q versus threshold

3. Eye pattern diagram
4. Constellation diagram
5. Histogram of the timing jitter
6. (based on the centroid of the marks)
7. Histogram of the timing jitter
8. (maximum mark value based)
9. Histogram of the ones in the maximum
10. Histogram of the zeros at center time

BER versus threshold diagram:
The diagram consists of two different curves. Both curves represent the cumulated BER values for both marks and spaces. The separation into two different curves is used to fit the tail extrapolation in case no real bit error occurs.

References

[1] M.C. Jeruchim, P. Balaban, K.S. Shanmugan, “Simulation of Communication Systems”, Plenum Press, New York / London, 1994.

[2] M.C. Jeruchim: “Techniques for Estimating the Bit Error Rate in the Simulation of Digital Communication Systems”, IEEE J. Select. Areas Commun., vol. 2, no. 1, pp. 153-170, 1984,.

7.8.5 Oscilloscope

Fundamentals

The oscilloscope enables the visualization of electrical as well as optical signals in the time domain. If an optical signal is to be displayed, the two polarization states will be shown, if polarization-dependent calculations are performed. The following units are supported:
The oscilloscope considers and visualizes a couple of *blocks*, therefore long bit sequences can be studied. The oscilloscope further displays the chirp of an arbitrary signal, which is defined as the deviation of the instantaneous frequency with respect to the carrier frequency of the signal. The chirp is calculated by

$$\delta f = \frac{1}{2\pi}\frac{d\varphi}{dt} \tag{1}$$

δf :	chirp
φ :	phase
t :	time

Visualizations

Unit	Amplitude	Power	Phase	Chirp
Electrical	V, A	V^2, A^2	-	-
Optical	$\sqrt{W}$	W	rad	GHz

Input / Output

	Number	Type of signal
Input	1	Optical field / Electrical signal
Output		no output

Component Parameters

Parameter Name	blocks
Description	Number of blocks to visualize
Default Value	1
Parameter Range	blocks ≤ Number of Blocks in Simulation Parameters

Parameter Name	Auto_save
Description	Saves the Oszilloscope data
Default Value	false
Parameter Range	boolean

Results

Parameter Name	power
Description	Average signal power (calculated from sampled signal)
Unit	W

Parameter Name	max. Value
Description	Maximum value

Parameter Name	min. Value
Description	Minimum value

Parameter Name	avg. Value
Description	Averge value

Parameter Name	Std
Description	Standard deviation

Parameter Name	norm. Std
Description	Normalized standard deviation

Parameter Name	mx
Description	Normalized amplitude

Graphs

The Oscilloscope contains the following Visualizers:

1. Oscilloscope, amplitude
2. Oscilloscope, power
3. Oscilloscope, phase
4. Chirp viewer

References

none

7.8.6 Parametric Signal Analyzer

Fundamentals

The Parametric signal Analyzer serves to make all properties of the parametric signal available as a result for the parameter variation.

Input / Output

	Number	Type of signal
Input	1	Optical field / Electrical signal
Output		no output

Results

Parameter Name	SNR
Description	Signal to noise ratio of the worst WDM channel
Unit	dB

Parameter Name	average Power
Description	highest average power of a WDM-channel
Unit	dBm

Parameter Name	minimum Power
Description	minimum WDM channel power
Unit	dBm

Parameter Name	maximum Power
Description	maximum WDM channel power
Unit	dBm

Parameter Name	Extinction
Description	quotient of the one and zero power level

Parameter Name	D*L
Description	Accumulated linear chromatic dispersion
Unit	ps/nm

Parameter Name	S*L
Description	Accumulated dispersion slope
Unit	ps/nm^2

References

none

7.8.7 PIN Photodiode

Fundamentals

PIN photodiodes are the most commonly used type of photodetectors in optical communication systems. Assuming that the photodetector is illuminated by an optical power $P(t)$, the average number of electrons generated by the photons arriving at the detector in a specific time instant Δt is given by

$$\langle n_e \rangle = \frac{\eta}{hf} P(t)\, \Delta t + n_d \Delta t \tag{1}$$

η :	Quantum efficiency
h :	Planck's constant
f :	Frequency
n_d :	Equivalent number of dark electrons

n_d denotes the number of generated charge carriers without optical input power. The dark current I_d of the photodiode is related to n_d by

$$n_d = \frac{I_d}{q} \tag{2}$$

q :	Electron charge

The output current of the diode follows as

$$i(t) = \frac{q n_e}{\Delta t} \tag{3}$$

where n_e denotes the number of electrons generated in the time instant Δt. The PDF of the photocurrent is described by a Poisson process [1]

$$f_{PIN}(n_e) = \frac{\langle n_e \rangle^{n_e}}{n_e!} \exp\{-\langle n_e \rangle\} \tag{4}$$

,

where the mean and variance equal $\langle n_e \rangle$.
If *separated channels* is chosen in the *simulation parameters* dialog, the power of all signal vectors is summarized to calculate the output current.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Electrical current

Component Parameters

Parameter Name	eta
Description	Quantum efficiency (η)
Default Value	0.798598
Parameter Range	0 ≤ eta ≤ 1

Parameter Name	Id
Description	Dark current (I_d)
Default Value	0
Parameter Range	0 ≤ Id ≤ 1e-3
Unit	A

References

[1] Jeruchim, Balaban, Shanmugan: Simulation of Communication Systems, Plenum Press, New York / London, 1994

7.8.8 Single Pulse Analyzer

Fundamentals

The Single Pulse analyzer delivers plenty of data on a single pulse (see results below) of an optical or electrical pulse in the time and in the frequency domain.

As the calculation of the pulse data is only possible with a single non-zero bit, you have to choose *Single Pulse* for the *Pattern generation* parameter of the pulse generator (or use an input file with a single "1" bit and a number of subsequent "0" bits).

Input / Output

	Number	Type of signal
Input	1	Optical field / Electrical signal
Output		no output

Results

Parameter Name	Pulse arrival time
Description	M_1/M_0 where M_1 is the arithmetic mean value

Parameter Name	RMS pulse width
Description	$\sqrt{M_2/M_0 - (M_1/M_0)^2}$ where M_1 is the arithmetic mean value and M_2 is the variance

Parameter Name	Pulse energy
Description	Moment M_0

Parameter Name	Peak power
Description	Maximum value of the power signal

Parameter Name	arrival_time
Description	Arrival time of the signal
Unit	ps

Parameter Name	DOP
Description	Degree of Polarization

Parameter Name	< 2*Psi >
Description	Medium stokes parameter S_1, S_2, S_3 expressed as value on the Poincaré-Sphere
Unit	degrees

Parameter Name	< 2*Chi >
Description	Medium stokes parameter S_1, S_2, S_3 expressed as value on the Poincaré-Sphere
Unit	degrees

References

[1] Karlsson, Magnus ; Sunnerud, Henrik: PMD Impact on Optical Systems: Single- and Multichannel Effects. Galtarossa, Andrea ; Menyuk, Curtis R. (Hrsg.): Polarization Mode Dispersion. New York, NY : Springer, 2004 (Optical and Fiber Communications Reports), S. 198-215 - ISBN 0-387-23193-5

7.8.9 Spectrum Analyzer

Fundamentals

The spectrum analyzer displays the power spectral density (PSD) of an optical or electrical signal. Both linear [mW/THz] and logarithmic [dBm/THz] scaling is possible. An averaging over several blocks is possible.
In the continuous case the single sided PSD of a signal can be calculated applying Parseval's theorem

$$\lim_{T\to\infty}\frac{1}{T}\int_0^T |s(t)|^2 dt = \int_0^\infty \lim_{T\to\infty}\frac{1}{T}|S(f)|^2 df = \int_0^\infty \mathrm{PSD}(f)\,df \tag{1}$$

s :	signal amplitude in the time domain
T :	time duration of one block
S :	signal amplitude in the frequency domain
PSD :	power spectral density of the signal

Using discrete sample points and an FFT algorithm for calculating the signal in the spectral domain, eq. (1) has to be modified to

$$\lim_{T\to\infty}\frac{1}{T}\sum_i |s(t_i)|^2\Delta t = \sum_i \lim_{T\to\infty}\frac{1}{T}\frac{\Delta t}{N\Delta f}|S(f_i)|^2\Delta f = \sum_i \lim_{T\to\infty}\frac{\Delta t^2}{T}|S(f_i)|^2\Delta f = \sum_i \mathrm{PSD}(f_i)\,\Delta f \tag{2}$$

Δt :	time discretization
Δf :	frequency discretization
N :	number of sample points per block

For an optical signal with two orthogonal polarized signal components, $|S(f_i)|^2$ has to be substituted by $|S_x(f_i)|^2 + \left|S_y(f_i)\right|^2$.
The logarithmic scaling of the PSD is performed using the relation

$$\mathrm{PSD}_{\log}(f_i) = \frac{10}{\mathrm{THz}}\log\left[|S(f_i)|^2\cdot\mathrm{THz}\right] \tag{3}$$

Graphs and Visualizations

The Spectrum Analyzer is capable of supplying the following Visualizers

1. Parametric signal
2. Parameterized power density spectrum (linear/logarithmic)
3. Sampled Spectrum (frequency) (linear/logarithmic)
4. Sampled Spectrum (wavelength) (linear/logarithmic)

Unit	PSD, linear scaling	PSD, logarithmic scaling
Electrical	mV/THz, mA/THz	dB(mV/THz), dB(mA/THz)
Optical	mW/THz	db(mW/THz)

	Number	Type of signal
Input	1	Optical field / Electrical signal
Output		no output

Input / Output

Component Parameters

Parameter Name	averaging
Description	Averaging over all blocks
Default Value	false
Parameter Range	boolean

Parameter Name	Sampling
Description	OSA filter characteristics
Default Value	Standard
Parameter Range	Standard, Rectangular

Parameter Name	auto_save
Description	Saves spectrum to file
Default Value	false
Parameter Range	boolean

Results

Parameter Name	Channel power (average)
Description	Average power of all WDM channels

Parameter Name	Channel power (variance)
Description	variance of channel powers

Parameter Name	Channel power (maximum difference)
Description	Difference between power of highest and lowest channel

Parameter Name	OSNR
Description	Optical signal to noise ratio

Parameter Name	Signal bandwidth
Description	Bandwidth of the signal (sigma)

References

none

7.9 Equalizer Components

7.9.1 Adaptive Equalizer

Fundamentals

The adaptive equalizer (AE) is comprised either of a finite impulse response filter structure (FIR) or an infinite impulse response filter structure (IIR) with an odd number of coefficients. Its purpose is to equalize an incoming distorted signal by adjusting the complex filter coefficients ("taps") of the chosen filter structure. The $n = 2 \cdot m + 1$ filter coefficients are spaced using an even spacing (usually selected to be one bit length or one half bit length).

To obtain the optimum set of filter coefficients, the user may choose from a variety of optimization routines described below. For an estimate of the output signal quality, a training sequence (including an undistorted signal) is required to drive the optimizer. The distorted signal has to be connected to port 1 (the upper port) and the undistorted training sequence has to be connected to port 2 (the lower port).

Once the set of optimum coefficients has been determined, the *same* set will be applied to all blocks if you use a multi block simulation; the coefficients will *not* be adapted for each block because it is assumed that only the noise representation varies between the different blocks. Thus, the training sequence will only be evaluated during the first block.

The AE may be used to counteract both phase ripple distortions such as group delay ripples (GDR) and distortions due to polarization mode dispersion (PMD). It may be used in the optical domain. For a more detailed description of the adaptive equalizer please refer to [1, 2].

Error criterion

As a default, the error criterion for the numerical optimizer is automatically calculated using [1, 2]:

$$\gamma = \sum_{i=1}^{\#samples} \vec{\epsilon} = \sum_{i=1}^{\#samples} ||\vec{s}_{i,filtered}|^2 - |\vec{s}_{i,\,training}|^2|, \tag{1}$$

where γ is the sum of the error vector $\vec{\epsilon}$ over all samples and i represents the actual sample index.

Thus, the adaptive equalizer may be used for both direct detection as well as coherent detection modulation formats and in the optical domain without further adjustment.

Numerical optimization algorithms

Currently, the following optimizers are available:

1. A simple random walk algorithm
2. A particle swarm optimizer [3, 4]
3. Threshold Accepting algorithm
4. Genetic Optimization algorithm
5. Trust-Region algorithm
6. Conjugate Gradient algorithm
7. Powell algorithm

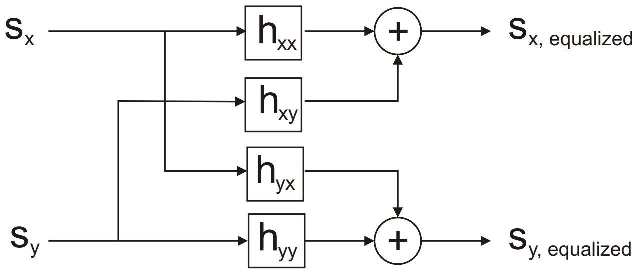


Figure 7.9.1. Butterfly structure for PMD equalization. Each of the h_{ij} subsets consists of a complete FIR or IIR filter structure.

8. Quasi-Newton algorithm

A detailed description of each optimization algorithm can be found on page 389 in section 7.14.1. For some optimizers, the value boundaries for the center coefficient of the FIR or IIR filter may be adjusted independently since the center coefficient often has higher values than the neighboring coefficients. If the parameter "useNeutralSettingForStartup" is set to true, the first particle of the swarm optimizer will automatically be initialized in the way that the FIR or IIR filter structure behaves "neutrally", not affecting the output signal at all. This can help to ensure that the PSO converges to a solution which is in fact *better* than the neutral setting which would always be possible in a real system.

Equalizing systems with two polarization axes

For equalizing systems with two polarization axes, the usage of a butterfly structure [5, 6] has shown to be very effective. This structure consists of four different equalizers which have access to both the information of the x-polarization axis and the y-polarization axis. The four equalizers can be divided into two subsets of two equalizers which have to be adjusted dependently from each other. The structure of a butterfly filter is depicted in figure 7.9.1.

The parameter *butterflyType* specifies whether you want to calculate the equalization for the x- or y-axis only or for both axes at the same time. In the first two cases, the overall computational effort is approximately increased by a factor of two in comparison to a single polarization system. In the latter case, the effort is increased by a factor of four.

Modes of Channel Operation

Currently, the AE only supports *single channel* simulations.

Choice of Tap Spacing and Number of Coefficients

The tap spacing (a.k.a. "taps per bit") defines, how tightly the complex filter coefficients of the AE are spaced in correspondence to the sample points per bits (in other words: the bit length) which can be specified in the simulation parameters.

Only values of the taps per bit which are equal to a power of 2 are allowed as a parameter. The minimum number of taps per bit is 1, the maximum is equal to the numbers of the samples per bit specified in the simulation parameters.

The choice of the optimum tap spacing and the number equalizer coefficients is highly dependent on the distortions you want to equalize and its inherent "memory length" in terms of the bit length. Usually, a tap spacing of 1 or 2 taps per bit and 9-15 equalizer coefficients are a good compromise between convergence speed and performance.

Examples

In your PHOTOSS examples directory, change to the subdirectory named `Adaptive Equalizer`. In there you will find two examples which deal with the equalization of GDR and PMD distortions.

Input / Output

	Number	Type of signal
Input	1	Optical field (distorted)
Input	2	Optical field (undistorted)
Output	1	Optical field (equalized)

Component Parameters

Parameter Name	numberOfTapsFIR
Description	number of taps of the FIR equalizer
Default Value	11
Parameter Range	1 < taps < 1e6

Parameter Name	numberOfTapsIIR
Description	number of taps of the IIR equalizer
Default Value	11
Parameter Range	1 < taps < 1e6

Parameter Name	tapsPerBit
Description	spacing of the taps in terms of the bit length
Default Value	1
Parameter Range	1 ≤ samplesPerBit; must be a power of 2
Unit	1/bit

Parameter Name	numberOfIterations
Description	number of iterations for the chosen optimizer
Default Value	100
Parameter Range	1 ≤ iterations < 1e6

Parameter Name	optimizer Type
Description	available optimizer type(s)
Default Value	Particle Swarm Optimizer
Parameter Range	Random Walk, Particle Swarm Optimizer

Parameter Name	numberOfParticles
Description	number of Particles of the swarm optimizer
Default Value	10
Parameter Range	1≤ particles < 1e6

Parameter Name	useNeutralSettingForStartup
Description	should the neutral equalizer setting be included in the initialization?
Default Value	true
Parameter Range	boolean

Parameter Name	minPosBoundary
Description	minimum allowed value for a non-center coefficient (both for the real and imaginary part)
Default Value	-0.75
Parameter Range	value < 0

Parameter Name	maxPosBoundary
Description	maximum allowed value for a non-center coefficient (both for the real and imaginary part)
Default Value	+0.75
Parameter Range	0 < value

Parameter Name	minCnterPosBoundary
Description	minimum allowed value for the center coefficient (both for the real and imaginary part)
Default Value	-1.2
Parameter Range	value < 0

Parameter Name	maxCenterPosBoundary
Description	maximum allowed value for the center coefficient (both for the real and imaginary part)
Default Value	+1.2
Parameter Range	0 < value

Parameter Name	positionStepwidth
Description	maximum “speed” of a particle for each iteration step
Default Value	0.01
Parameter Range	0 < value

Parameter Name	butterflyType
Description	Type of used butterfly equalizer. The chosen axis/axes will be equalized.
Default Value	Both Axes
Parameter Range	X-Axis, Y-Axis, Both Axes

Results

References

[1] Matthias Westhäuser et al, “Reducing Fiber Bragg-Grating induced group delay ripples in 112 Gbit/s metro networks using generically initialized transversal filters”, ITG-Fachtagung “Photonische Netze”, Leipzig, May 2010.

[2] Matthias Westhäuser et al, “Optimization of Optical Equalization of Group Delay Ripple-Induced Penalties from Fiber Bragg Gratings in 112 Gbit/s Metro Networks”, Optics and Photonics Congress “Advanced Photonics”, Karlsruhe, Germany, June 2010.

[3] James Kennedy and Russell Eberhart, “Particle Swarm Optimization”, Proceedings of the IEEE International Conference on Neural Networks, IV, (Piscataway, NJ: IEEE Service Center, 1995), pp. 1942-1948, 1995.

[4] Ying Zhou et al, “Particle swarm optimization-based approach for optical finite impulse response filter design”, Applied Optics, Vol. 42, No. 8, pp. 1503-1507, 2003.

[5] Seb J. Savory, “Digital filters for coherent optical receivers”, Opt. Express., Vol. 16, No. 2, pp. 804-817, 2008.

[6] Seb J. Savory, “Electronic compensation of chromatic dispersion using a digital coherent receiver”, Opt. Express., Vol. 15, No. 5, pp. 2120-2125, 2007.

7.9.2 Adaptive One Tap Equalizer

Fundamentals

The adaptive one tap equalizer (AOTE) is a simple one-tap equalizer which works in a very similar way as the adaptive equalizer described in section 7.9.1 on page 308. The most important difference is that this equalizer only consists of one tap - the center tap. Therefore, it is most useful for the mitigation of crosstalk effects or simple signal attenuation or amplification.

Numerical optimization algorithms

A detailed description of each optimization algorithm can be found on page 389 in section 7.14.1.

Input / Output

	Number	Type of signal
Input	1	Optical field (distorted)
Input	2	Optical field (undistorted)
Output	1	Optical field (equalized)

Component Parameters

Parameter Name	numberOfTrainingSymbols
Description	number of training symbols which can be used by the equalizer
Default Value	64
Parameter Range	1 < symbols < symbols per block

Parameter Name	trainingSamplesPerBit
Description	number of training samples per bit
Default Value	2
Parameter Range	1 ≤ samplesPerBit; must be a power of 2
Unit	1/bit

Results

References

[1] Matthias Westhäuser et al, “Reducing Fiber Bragg-Grating induced group delay ripples in 112 Gbit/s metro networks using generically initialized transversal filters”, ITG-Fachtagung “Photonische Netze”, Leipzig, May 2010.

[2] Matthias Westhäuser et al, “Optimization of Optical Equalization of Group Delay Ripple-Induced Penalties from Fiber Bragg Gratings in 112 Gbit/s Metro Networks”, Optics and Photonics Congress “Advanced Photonics”, Karlsruhe, Germany, June 2010.

7.9.3 MIMO Equalizer

Fundamentals

The MIMO equalizer is able to equalize linearly distorted signals in multiple interfering channels. For example, the equalizer can be utilized to remove PMD, PDL or coherent crosstalk effects (i.e. induced by the MCF crosstalk emulator) from a signal. Basically, the equalizer has been designed to remove intersymbol interference (ISI) from an incoming signal. To use the MIMO equalizer, set the number of transceivers and receivers according to the number of channels you want to use. You get twice the amount of input ports than the number of transceivers. For each channel, you have one input port for the distorted signal and one input for the undistorted signal (respectively the *training sequence*) which appear in an alternating order. For example, with two transmitters and receivers the configuration of the equalizer is as follows:

	Number	Type of signal
Input	1	Channel 1 Optical field (distorted)
Input	2	Channel 1 Optical field (undistorted)
Input	3	Channel 2 Optical field (distorted)
Input	4	Channel 2 Optical field (undistorted)
Output	1	Channel 1 Optical field (equalized)
Output	2	Channel 2 Optical field (equalized)

The MIMO equalizer will then use one of the numerical optimization algorithms described below to determine the optimum set of complex equalizer coefficients to equalize the signal. The computational effort depends on the type of the chosen algorithm and its parameter settings as well as the length of the training sequence. Please note that the MIMO equalizer only works with two polarization axes, i.e. you have to enable the simulation parameter *Include Polarization Effects*. No additional input ports are necessary to process an optical signal with two distinct polarization (e.g. a polarization multiplexed signal).

For each channel, the principle of the adaptive equalization process is depicted in 7.9.2.

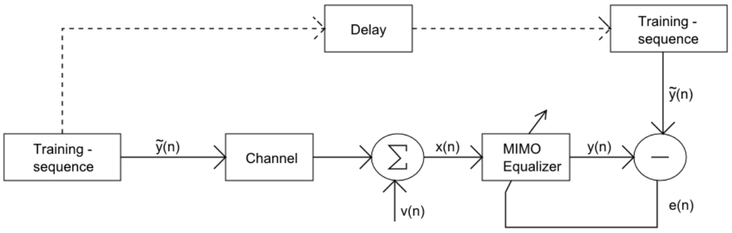


Figure 7.9.2. Schematic of the adaptive MIMO equalization for one channel

Additionally, with a right mouse click on the MIMO equalizer icon on the PHOTOSS interface, you can display the residual error signal *e*(n). After the optimization several results are displayed in the component result windows. The start and end residuum represents the residual error before and at the end of the optimization. The best residuum characterizes the smallest residual error that appears in the whole optimization. Especially in the stochastical algorithms, e.g. 'Random Walk', the best residuum is rarely at the end of the optimization.

For a more detailed description of coherent MIMO equalization please refer to [1]. Information about equalization of inter-channel interference is published in [2].

Noise

When using the MIMO equalizer, you should consider how it will influence noise components of the signal: If the *numerical* noise model (see section 3.9) is used, the equalizer will apply its transfer function directly to both signal and noise components of the sampled signal. Depending on the chosen equalizer coefficients, this may lead to noise enhancement - see [1] for more information on that topic.
When using *analytical* noise, the MIMO equalizer will *not* influence the signals noise components. This assumption is only valid when the noise components are not dominating the signal and the choice of equalizer coefficients will result in a passive behavior of the equalizer, i.e. the total signal is *not* amplified when passing through the equalizer.

Training Sequences

The MIMO equalizer requires that the training sequence for each distinct (distorted) input channel be supplied. In PHOTOSS, you should simply use the undistorted output of the respective pulse generator / transmitter structure to equalize your signal. Keep in mind that the length of the training sequence significantly influences the computational time needed to derive the equalizer coefficients. Usually, the training sequence will have the same length as the distorted signal. If you transmit a large number of symbols per block, you might consider limit the length of the training sequence by using the parameter *limitNumberOfTrainingSymbols*. As a rule of thumb, 512 symbols are usually sufficient for most problems but the exact amount depends on the total memory length defined by the amount of ISI you wish to remove from the signal. If the training sequence becomes too larger, this may result in an “Out of memory” error while deriving the coefficients.

Optimization algorithms

FIR Filter structure:
More information on each optimization algorithm can be found in section 7.14.1 on page 389.

- **Zero Forcing**: The optimizer ’Zero Forcing’ calculates the inverse channel matrix and multiplies it with the received signals. You only have to set the number of transceivers and receivers. The structure of the ’Zero Forcing’ filter is a 1 Tap butterfly structure.

- **Matrix Inversion**: This optimizer calculates the inverse matrix of the received signals and multiplies it with the training sequences. It is possible to increase the amount of filter coefficients to improve the equalization results.

- **Least Mean Square**: The LMS algorithm uses a gradient descent to minimize the error signal and to determine the filter coefficients. Set the value of the step size to a number $0 < stepsize < 2$. The number of iterations reflects the number of times, the algorithm goes through the training sequences to determine the filter coefficients.

- **Normalized Least Mean Square**: The NLMS algorithm is a modified version of the common LMS optimizer. To counteract the fact that a varying energy of incoming symbols amplifies the noise of the gradients, the step size is scaled with the squared magnitude of the incoming symbols.

- **Recursive Least Square**: The RLS is a recursive method to calculate the filter coefficients. You have to set the forgetting factor, which represents the weighting of earlier samples. Experience has shown that useful results are calculated with a forgetting factor between 0.9 and 1.

- **Load from file**: There is also the possibility to load a set of filter coefficients directly from a file.

- Other various optimization routines are explained in more detail on page 389 in section 7.14.1.

IIR Filter structure:

- **Least Mean Square IIR**: For the LMS IIR algorithm, you have to set the amount of feed forward and feedback filter coefficients separately. Set the value of the step size to an arbitrary number ¿ 0 and the number of iterations. The number of iterations reflects the number of times, the algorithm calculates the new signal matrix and goes trough the trainings sequence to approximate the filter coefficients.

Examples

In your PHOTOSS examples directory, change to the subdirectory named `MIMO Equalizer`. In there you will find two examples which deal with the equalization of PMD distortions and crosstalk of an MCF.

Component parameters

Parameter Name	numberOfTransmitters
Description	sets the number of transmitters
Default Value	1
Parameter Range	$1 \leq numberOfTransmitters \leq 100$

Parameter Name	numberOfReceivers
Description	sets the number of receivers
Default Value	1
Parameter Range	$1 \leq numberOfReceivers \leq numberOfTransmitters$

Parameter Name	numberOfTaps
Description	Sets the number of complex filter taps. The larger the number of filter taps, the larger the memory length of the equalizer (the necessary memory length will be given by the amount of ISI you wish to remove from the signal).
Default Value	11
Parameter Range	$1 \leq numberOfTaps \leq 1e6$

Parameter Name	tapsPerBit
Description	sets the number of taps per bit
Default Value	2
Parameter Range	$1 \leq tapsPerBit$ samplesPerBit

Parameter Name	numberOfIterations
Description	determines, how many times the optimizer iterates over the same training sequence
Default Value	100
Parameter Range	$1 \leq numberOfIterations \leq 1e100$

Parameter Name	maxStepsize
Description	sets the maximum step size for the gradient descent, important for convergence speed
Default Value	0.1
Parameter Range	$0 < maxStepsize < 2$

Parameter Name	forgettingFactor
Description	the forgetting factor is the ”‘weight”’ of calculations for earlier samples, with a forgetting factor of 0.1, the RLS does not use older calculations as much as with a forgetting factor of 0.9
Default Value	0.9
Parameter Range	$0 < forgettingFactor < 1$

Parameter Name	numberOfFeedforwardTaps
Description	sets the number of feed forward taps for the IIR Filter
Default Value	1
Parameter Range	$1 \leq numberOfFeedforwardTaps \leq 1e6$

Parameter Name	numberOfFeedbacktaps
Description	sets the number of feedback taps for the IIR Filter
Default Value	1
Parameter Range	$1 \leq numberOfFeedbacktaps \leq 1e6$

Parameter Name	limitNumberOfTrainingSymbols
Description	If set to true, the maximum number of training symbols can be specified. If set to false, all symbols per block will be used as training symbols. Used in all optimizers except 'Least Mean Square' and 'Recursive Least Square'. A larger number of training symbols may result in significantly higher computational effort to derive the equalizer coefficients.
Default Value	false
Parameter Range	boolean

Parameter Name	numberOfTrainingSymbols
Description	Number of used training symbols as a power of 2. Used in all optimizers except 'Least Mean Square' and 'Recursive Least Square'.
Default Value	9
Parameter Range	$0 \leq numberOfTrainingSymbols \leq 100$

Parameter Name	useNeutralSettingForStartup
Description	Should a neutral equalizer setting always be included in the initial startup condition for the optimizer? Used in 'Conjugate-Gradient', 'Quasi-Newton', 'Trust-Region', 'Genetic', 'Treshold-Accepting', 'Particle Swarm', 'Random Walk', 'Normalized Least Mean Square' and 'Powell'
Default Value	true
Parameter Range	boolean

Parameter Name	useStabilityAndPassivity
Description	Guarantees the stability and passivity of an equalizer with one Tap. If more Taps are chosen, each one will have a maximum absolute value, that is equal to one. Used in the 'Random Walk'
Default Value	true
Parameter Range	boolean

Parameter Name	resultTolerance
Description	The allowed difference of the residual error criterion to zero after the optimization is done. Used in 'Conjugate-Gradient', 'Quasi-Newton', 'Trust-Region', 'Genetic' and 'Treshold-Accepting'.
Default Value	0.1
Parameter Range	$0 \leq resultTolerance \leq 1$

Parameter Name	stepsize
Description	The stepsize controls the convergence speed and quality of the NLMS algorithm.
Default Value	0.1
Parameter Range	$1e-7 \leq stepsize < 2$

Parameter Name	useUserOffsetForResampling
Description	Do you want to supply a constant offset for the bit center sample for internal resampling? If set to false, PHOTOSS will attempt to determine the offset automatically.
Default Value	false
Parameter Range	$false < true$

Parameter Name	userResamplingOffset
Description	Offset for signal resampling
Default Value	1
Parameter Range	$0 \leq userResamplingOffset \leq samplesPerBit$

Parameter Name	useUserInitialGuess
Description	Do you want to supply a file containing an initial guess for the equalizer coefficients? The initial guess will be used for the optimization algorithm. If you have chosen the setting 'useNeutralSettingForStartup' it will be overwritten by your custom coefficients.
Default Value	false
Parameter Range	boolean

Parameter Name	probability
Description	The probability gives the 'Random Walk' algorithm the chance to get out of a local minimum. The convergence depends highly on the interplay between the number of iterations and the probability.
Default Value	0
Parameter Range	$0 \leq probability \leq 1$

Parameter Name	minCoeffBoundary
Description	Sets the minimum boundary for the tap coefficient value. Sets the real and imaginary part of the complex value. Used in 'Random Walk', 'Genetic' and 'Treshold Accepting'.
Default Value	-2
Parameter Range	$-1e100 \leq minCoeffBoundary \leq 1e100$

Parameter Name	maxCoeffBoundary
Description	Sets the maximum boundary for the tap coefficient value. Sets the real and imaginary part of the complex value. Used in 'Random Walk', 'Genetic' and 'Treshold Accepting'.
Default Value	2
Parameter Range	$-1e100 \leq maxCoeffBoundary \leq 1e100$

Parameter Name	cgMethod
Description	What for a variant should be used for the Conjugate-Gradient optimization step? The Polak-Ribiere method is slightly better than the Fletcher-Reeves method.
Default Value	Polak-Ribiere
Parameter Range	$Fletcher-Reeves < Polak-Ribiere$

Parameter Name	stepBoundary
Description	The maximum boundary for an decrease in the radius of the model function of the 'Trust Region' optimizer.
Default Value	0.1
Parameter Range	$0 \leq stepBoundary \leq 1e100$

Parameter Name	numberOfIndividuals
Description	The number of individuals of the 'Genetic' optimizer.
Default Value	50
Parameter Range	$1 \leq numberOfIndividuals \leq 1e12$

Parameter Name	tourneySize
Description	The number of individuals who ’fight’ each other in order to be chosen for the next generation (better fitness wins). Used in the ’Genetic’ optimizer.
Default Value	10
Parameter Range	$0 \leq tourneySize \leq 1e100$

Parameter Name	keepBestIndividual
Description	The number of individuals who ’fight’ each other in order to be chosen for the next generation (better fitness wins). Used in the ’Genetic’ optimizer.
Default Value	true
Parameter Range	boolean

Parameter Name	crossoverProbability
Description	Probability of a gene crossover inside an individual. Used in the ’Genetic’ optimizer.
Default Value	0.8
Parameter Range	$0 \leq crossoverProbability \leq 1$

Parameter Name	mutationProbability
Description	Probability of a gene mutation inside an individual. Used in the ’Genetic’ optimizer.
Default Value	0.8
Parameter Range	$0 \leq mutationProbability \leq 1$

Parameter Name	startTemperature
Description	The optimizer starts with this temperature, which is decreasing during the algorithm. Used in the ’Treshold-Accepting’ optimizer.
Default Value	0.3
Parameter Range	$0 \leq startTemperature \leq 1e100$

Parameter Name	minTemperature
Description	The optimizer ends with this temperature, which is commonly zero. Used in the ’Treshold-Accepting’ optimizer.
Default Value	0
Parameter Range	$0 \leq minTemperature \leq 1e100$

Parameter Name	maxStep
Description	The maximum distance from one point to a randomly generated neighbouring point. Used in the ’Treshold-Accepting’ optimizer.
Default Value	0.15
Parameter Range	$0 \leq maxStep \leq 1e100$

Parameter Name	temperatureChange
Description	The decrease of the temperature in each step. Used in the ’Treshold-Accepting’ optimizer.
Default Value	0.005
Parameter Range	$0 \leq temperatureChange \leq 1e100$

References

[1] A.R. Shah, R.C.J. Hsu, A. Tarighat, A.H. Sayed and B. Jalali, “Coherent Optical MIMO (COMIMO)”, Journal of Lightwave Technology, vol. 23, no. 8”, August 2005.

[2] M. Westhäuser, S. Akhtari, M. Finkenbusch, P.M. Krummrich, “Quantification and Reduction of OSNR-Penalties in High Bit Rate Multi Core Fiber Systems”, ITG Fachtagung ”‘Photonische Netze”’, Leipzig, Karlsruhe, Mai 2012.

7.9.4 Constant Modulus Algorithm (CMA) Equalizer Fundamentals

The Constant Modulus Algorithm (CMA) Equalizer is comprised of a finite impulse response filter structure (FIR) with an odd number of coefficients. Its purpose is to equalize an incoming distorted signal by adjusting the complex filter coefficients (“taps”) of the chosen filter structure. The $n = 2 \cdot m + 1$ filter coefficients are spaced using an even spacing (usually selected to be one bit length or one half bit length). More details can be found in [1-2].

The CMA equalizer is a blind equalizer. This means, that no external feedback criterion or training sequence is used to aid the algorithm in finding the optimum set of coefficients for the equalization of the incoming signal. Instead, the CMA equalizer works under the assumption, that the modulus of the constellation points of all symbols in the incoming modulation format is equal to one. It strives to adjust its equalizer taps in such a way, that the constellation points of all outgoing symbols are as close to unity as possible.

The complex tap coefficients will be determined using the following update rules in each iteration [1-2]:

$$s_{x,equalized} = \vec{s}_{x,distorted} * \vec{h}_{xx} + \vec{s}_{y,distorted} * \vec{h}_{xy} \tag{1}$$

$$s_{y,equalized} = \vec{s}_{x,distorted} * \vec{h}_{yx} + \vec{s}_{y,distorted} * \vec{h}_{yy} \tag{2}$$

$$e_x = 1 - (s_{x,equalized} \cdot s^*_{x,equalized}) \tag{3}$$

$$e_y = 1 - (s_{y,equalized} \cdot s^*_{y,equalized}) \tag{4}$$

$$\vec{h}_{xx} = \vec{h}_{xx} + \mu \cdot \vec{s}^*_{x,distorted} \cdot s_{x,equalized} \cdot e_x \tag{5}$$

$$\vec{h}_{xy} = \vec{h}_{xy} + \mu \cdot \vec{s}^*_{y,distorted} \cdot s_{x,equalized} \cdot e_x \tag{6}$$

$$\vec{h}_{yx} = \vec{h}_{yx} + \mu \cdot \vec{s}^*_{x,distorted} \cdot s_{y,equalized} \cdot e_y \tag{7}$$

$$\vec{h}_{yy} = \vec{h}_{yy} + \mu \cdot \vec{s}^*_{y,distorted} \cdot s_{y,equalized} \cdot e_y \tag{8}$$

$*$:	denotes the convolution of two vectors
* :	denotes complex conjugation
$s_{i,equalized}$:	equalized signal value for pol-plane i and center tap
$\vec{s}_{i,distorted}$:	distorted signal vector for pol-plane i
$\vec{h}_{ij}$:	vector containing a set of complex tap coefficients for pol-plane i,j
e_i :	complex residual error for the modulus 1 and equalized signal value
μ	CMA stepwidth factor

However, this also imposes restrictions on the incoming type of signal: If the modulus is not equal to unity for each constellation point (e.g. for the modulation format ’duobinary’ or ’(N)RZ ASK’), the CMA may not converge to a stable solution. The PHOTOSS implementation of the CMA has been created for polarization-multiplexed modulation formats such as POLMUX-QPSK and consists of a butterfly FIR filter structure. Thus, four sets of FIR equalizer coefficients which all have the same number of taps are being adjusted.

The CMA may be used to counteract (e.g.):

- a rotation of the polarization axis of the incoming POLMUX signal
- distortions due to polarization mode dispersion (PMD) effects such as DGD and higher order PMD
- distortions due to linear chromatic dispersion
- distortions due to phase ripple(s) / group delay ripples (GDR)
- various other linear distortion effects

Once the set of optimum coefficients has been determined, the *same* set will be applied to all blocks if you use a multi block simulation; the coefficients will *not* be adapted for each block because it is assumed that only the noise representation varies between the different blocks.

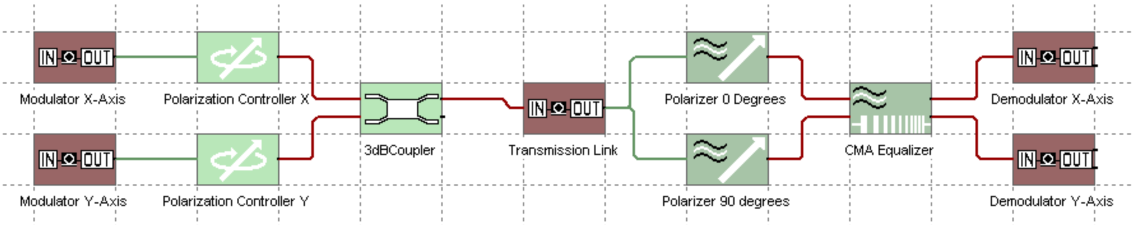


Figure 7.9.3. Setup for the usage of the CMA equalizer

CMA Input

Currently, the CMA equalizer works in the optical domain, only. It is comprised of two input ports to emulate the input of the two polarization-multiplexed tributaries on the x- and y-polarization axis. The general setup for the usage of the CMA equalizer is depicted in figure 7.9.3 (for a working simulation example please refer to the PHOTOSS examples directory).

The setup consists of two separate signal components: The first component is multiplexed on the x-polarization axis, the second one on the y-polarization axis. The 'transmission link' may add any kind of linear (or nonlinear) distortion effects such as PMD, GDR, etc. After the transmission link, two polarizer components will act as a polarization beam splitter (PBS). The first and second polarizer have to be adjusted to an angle of $\vartheta_1 = 0$ degrees and $\vartheta_2 = 90$ degrees, respectively, to match the PHOTOSS world polarization coordinate system for the x- and y-polarization axis.

Please note that this setting does not require that the incoming signal is actually rotated in the same way! The incoming signal before the polarizers may have experienced any kind of rotation in polarization space. However, no additional rotation in polarization space is allowed *after* the PBS. Thus, the incoming signal for in-port 1 of the CMA equalizer only carries signal power on the x-axis and the signal for in-port 2 only carries signal power on the y-axis. If this is not the case, PHOTOSS will issue an error message.

CMA Parameters

The CMA equalizer component consists of several parameters which influence the convergence of the CMA and its general ability to equalize distorted signals. `numberOfTaps` specifies the number of FIR filter taps for the equalizer, while `tapsPerBit` specifies how many taps are placed inside one bit length (the minimum is 1). The `bitRateToSymbolRateRatio` specifies the relation between the bitrate of the simulation parameters and the symbol rate of each polarization multiplexed tributary. The CMA equalizer will carry out `numberOfIterations` iterations, while the step width for each iteration is defined by the `cmaStepwidth`.

Examples

In your PHOTOSS examples directory, change to the subdirectory named `CMA Equalizer`. In there you will find an example which deals with the equalization of PMD-induced distortions for a POLMUX-QPSK signal.

Input / Output

	Number	Type of signal
Input	1	Optical field (power on x-pol-axis)
Input	2	Optical field (power on y-pol axis)
Output	1	Optical field (equalized, x-pol-tributary)
Output	2	Optical field (equalized, y-pol-tributary)

Component Parameters

Parameter Name	numberOfTaps
Description	number of taps of the FIR equalizer
Default Value	9
Parameter Range	1 < taps < 1e6

Parameter Name	tapsPerBit
Description	spacing of the taps in terms of the bit length
Default Value	1
Parameter Range	1 ≤ samplesPerBit; must be a power of 2
Unit	1/bit

Parameter Name	bitRateToSymbolRateRatio
Description	relation between simulation parameters bitrate and symbol rate for each tributary
Default Value	2
Parameter Range	1 ≤ bitRateToSymbolRateRatio
Unit	1/bit

Parameter Name	numberOfIterations
Description	number of iterations for the CMA
Default Value	2000
Parameter Range	1 ≤ iterations < 1e6

Parameter Name	cmaStepwidth
Description	stepwidth for each CMA iteration
Default Value	0.005
Parameter Range	0 < cmaStepwidth

Results

References

[1] Seb J. Savory, “Electronic compensation of chromatic dispersion using a digital coherent receiver”, Opt. Express., Vol. 15, No. 5, pp. 2120-2125, 2007.

[2] Seb J. Savory, “Digital filters for coherent optical receivers”, Opt. Express., Vol. 16, No. 2, pp. 804-817, 2008.

7.9.5 Dispersion Equalizer

Fundamentals

The Dispersion Equalizer (DE) is composed of a finite impulse response filter structure (FIR) which can be used to remove accumulated dispersion from a signal. The symmetrical FIR filter structure is composed of several filter “taps” which are used to influence the time domain signal and are inducing a frequency-dependent delay or acceleration. For a detailed explanation of the working principle, please refer to [1-2].

The values for the complex filter coefficients are determined using the following relation:

$$a_k = \sqrt{\frac{jcT^2}{D\lambda^2 z}} \exp\left(-j\frac{\pi c T^2}{D\lambda^2 z}k^2\right). \tag{1}$$

a_k :	Value of the kth symmetrical complex filter coefficient
N :	Number of complex filter coefficients
k :	Active coefficient index ranging from $-N/2$ to $+N/2$
j :	Complex number
c :	vacuum speed of light in [m/s]
T :	Tap delay in [s]. Given by $t_{bit}/tapsPerBit$
D :	Dispersion in [ps/(nm * km)]
z :	Propagation distance / length of the fiber in [km]

The number of the complex filter coefficients N can be calculated by using

$$N = 2 \cdot \frac{|D|\lambda^2 z}{2cT^2} + 1. \tag{2}$$

Operating Domain

The DE can be selected to operate in the optical domain only. You are not restricted to the use of specific modulation formats.

Modes of Channel Operation

For *single channel* systems, the DE can be used without further adjustments when setting the parameter f_0 of the DE equal to the center frequency of your channel and / or your fiber. However, the effectiveness of the DE is also dependent on the chosen frequency slice and frequency spacing of the simulation parameters. For optimal results, the reference center frequency of the simulation should match the center frequencies of the source, the fiber and the DE.

For *multi channel* systems, the DE should be used after the optical filter. The DE can only mitigate the dispersion for one chosen channel. Please set the center frequency of the DE equal to the center frequency of the source of your chosen channel.

In general, the DE works with both the separated channels and the total field approach.

Usage of Two Polarization Planes

The DE can be used in simulations with one or two polarization axes and also in combination with polarization-multiplexed signals.

Choice of Tap Spacing

The tap spacing (a.k.a. "taps per bit") defines, how tightly the complex filter coefficients of the DE are spaced in correspondence to the sample points per bits which can be specified in the simulation parameters.

Only values of the taps per bit which are equal to a power of 2 are allowed as a parameter. The minimum number of taps per bit is 1, the maximum is equal to the numbers of the samples per bit specified in the simulation parameters.

Keep in mind that a tighter spacing of taps per bit may improve the results of the dispersion equalizer. However, this also increases the number of equalizer taps needed to mitigate a certain amount of dispersion and it may significantly increase its computational time.

Examples

In your PHOTOSS examples directory, change to the subdirectory named `Dispersion Equalizer`. In there you will find an example for the usage of the DE. It demonstrates how the DE can equalize an incoming signal with a fixed dispersion. The delay of the signal due to the FIR equalization is automatically adjusted in the DE.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	channelBitrate
Description	Bitrate of the desired channel
Default Value	40
Parameter Range	0 < channelBitrate
Unit	Gbit/s

Parameter Name	dispersion
Description	Defines the accumulated dispersion which shall be mitigated.
Default Value	100
Parameter Range	$-1e100 \leq$ dispersion $\leq +1e100$
Unit	ps/nm

Parameter Name	f0
Description	Center Frequency of the desired channel
Default Value	193.1
Parameter Range	Inside the simulation bandwidth.
Unit	THz

Parameter Name	tapsPerBit
Description	Defines the spacing of the equalizer taps in means of samples
Default Value	2
Parameter Range	1 ≤ tapsPerBit ≤ samplesPerBit; must be a power of 2
Unit	samples

Results

Parameter Name	numberOfTaps
Description	Number of the required complex filter coefficients N of the equalizer.

References

[1] Seb J. Savory, “Digital filters for coherent optical receivers”, Opt. Express., Vol. 16, No. 2, pp. 804-817, 2008.

[2] Seb J. Savory, “Electronic compensation of chromatic dispersion using a digital coherent receiver”, Opt. Express., Vol. 15, No. 5, pp. 2120-2125, 2007.

7.10 Additional Components

7.10.1 Attenuator

Fundamentals

The attenuator is an idealized component that is not based on an actual physical component. It can be used to manipulate a signal by means of multiplying a constant α (in dB) to the input signal. Since the component is an attenuator, positive values of this parameter will attenuate the signal, while negative ones will cause the signal to gain power. **(The meaning of the sign of the parameter α has been changed from version 4.00 to 4.10).**
If polarization effects are included, each polarization axis may be set to a different attenuation value.

Input / Output

	Number	Type of signal
Input	1	Optical field / Electrical signal
Output	1	Optical field / Electrical signal

Component Parameters

Parameter Name	alpha_1
Description	Loss of the first polarization axis (related to power)
Default Value	5
Parameter Range	not limited
Unit	dB

Parameter Name	alpha_2
Description	Loss of the second polarization axis (related to power)
Default Value	5
Parameter Range	not limited
Unit	dB

References

none

7.10.2 Coax Line

Fundamentals

An electrical coaxial line can be induced in the network. The amplitude transfer function of the coaxial line is given by

$$H(f) = \exp\left(-\sqrt{\frac{f}{f_n}} \cdot L\right) \tag{1}$$

whereby L denotes the length of the coax cable, and f_n the 1 dB amplitude cut-off frequency of a cable with a length of 1 km.

Input / Output

	Number	Type of signal
Input	1	Electrical signal
Output	1	Electrical signal

Component Parameters

Parameter Name	fn
Description	1 dB (Neper cut off frequency)
Default Value	1e7
Parameter Range	1e3 ≤ fn ≤ 1e10
Unit	Hz

Parameter Name	L
Description	Length of the line
Default Value	1
Parameter Range	0 < L ≤ 100
Unit	km

References

none

7.10.3 Delay Component

Fundamentals

This component causes a time delay in the ps-range, which is e. g. useful to observe and compare signals suffering from different delay times with an oscilloscope.
The component can only delay the signal in multiples of the sampling interval.

Input / Output

	Number	Type of signal
Input	1	Optical field / Electrical signal
Output	1	Optical field / Electrical signal

Component Parameters

Parameter Name	Delay
Description	Delay time
Default Value	12.5
Parameter Range	-Block Time < Delay < +Block Time
Unit	ps

References

none

7.10.4 Limiter

Fundamentals

The limiter is an idealized component (i.e. not based on an actual physical component). It can be used to manipulate a signal by means of limiting the input signal to a maximum and minimum value, A_{Min} (in A/V) and A_{Max} (in A/V/W). It is emphasized that the the amplitude value is specified in terms of [W] just as in the component Pulse Generator (see 7.1.3 on page 145).

The average *power* value of the signal will be recalculated by using the simplified relation

$$P_{avg} = \frac{P_{max} - P_{min}}{2}. \tag{1}$$

Please keep in mind that this calculation need not necessarily be accurate, since (e.g.) phase modulation formats have a very different average power level than the mean of the maximum and minimum power. In these cases, be sure to use the simulation mode *combined simulation* and to activate *Use sampled power* in all EDFA components in your simulation (see section 7.4.1 on 197 for further details). Additionally, this approach may produce incorrect results when considering analytical noise - numerical noise should be used instead.

Input / Output

	Number	Type of signal
Input	1	Optical Field / Electrical signal
Output	1	Optical Field / Electrical signal

Component Parameters

Parameter Name	Unit
Description	unit of input and output signal
Default Value	$\sqrt{W}$
Parameter Range	A, V, $\sqrt{W}$
Unit	variable

Parameter Name	A_min
Description	minimum Amplitude (0)
Default Value	0.0
Parameter Range	not implemented
Unit	determined by parameter “Unit”

Parameter Name	A_max
Description	maximum Amplitude (0)
Default Value	0.02
Parameter Range	not implemented
Unit	determined by parameter “Unit”

References

none

7.10.5 Phase Conjugator

Fundamentals

The phase conjugator converts the input signal to a phase-conjugate output in which the signal is inverted with respect to the mirror frequency f [1-4]. An ideal phase conjugation is performed neglecting influences of noise:

$$A_{out} \propto A_{in}^{*} \tag{1}$$

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	f
Description	Mirror frequency
Default Value	193.1
Parameter Range	within the PHOTOSS simulation band width
Unit	THz

References

[1] R. Ramaswami, K. N. Sivarajan, “Optical Networks - A Practical Perspective”, 2nd ed., Morgan Kaufmann, 2002.

[2] T. Merker, P. Meissner, U. Feiste, “High Bit-Rate OTDM Transmission Over Standard Fiber Using Mid-Span Spectral Inversion and Its Limitations”, IEEE J. Sel. Top. In Quant. Electron., vol. 6, no. 2, April 2000.

[3] S. Watanabe, M. Shirasaki, “Exact compensation for both chromatic dispersion and Kerr effect in a transmission fiber using optical phase conjugation”, J. Lightw. Technol., vol. 16, no. 3, March 1996.

[4] A. Yariv, D. Fekete, D. M. Pepper, “Compensation for channel dispersion by nonlinear optical phase conjugation”, Optics Letters, vol. 4, no. 2, February 1979.

7.10.6 Phase Ripple Emulator

Fundamentals

The Phase Ripple Emulator (PRE) is designed to create sinusoidal single or multi-component phase ripples across the incoming sampled signal spectrum. Such a phase ripple might in reality be created (e.g.) due to the usage of Chirped Fiber Bragg Gratings (CFBGs) which have a group delay ripple (GDR). The group delay may be expressed as:

$$\tau_g(\lambda) = -\frac{\lambda^2}{2\pi c} \cdot \frac{d\varphi(\lambda)}{d\lambda} \tag{1}$$

where $\varphi(\lambda)$ is the wavelength-dependent phase response of the CFBG, and λ is the wavelength of the current channel. By integrating (1) we can obtain the phase response (material dispersion will not be considered):

$$\varphi(\lambda) = -2\pi c \int\limits_{\lambda_{min}}^{\lambda_{max}} \frac{\tau_g(\lambda)}{\lambda^2} d\lambda. \tag{2}$$

A Fourier analysis is often used to examine, which frequency components of the group delay ripple play the most dominant parts in creating the phase ripple. The frequencies of these components are denoted as “ripple periods” and have the unit [Hz]. In a first order approximation, the phase response for each distinct frequency $\varphi(f)$ can be approximated with a sum of sine functions in the frequency domain [1-3]:

$$\varphi(f) = -\sum_{i=1}^{N} \Delta\tau_{G,i}(f) \cdot f_{r,i} \cdot \sin\left(\frac{2\pi f}{f_{r,i}} + \varphi_{offset,i}\right). \tag{3}$$

$\Delta\tau_{G,i}(f)$ represents the phase ripple amplitude for each of the N ripple frequency components with frequency $f_{r,i}$ (also denoted as “ripple period”). $\varphi_{offset,i}$ describes a phase offset for each frequency component and f is the current frequency for which the ripple shall be derived.

Sometimes, it may be beneficial to emulate phase ripples from a point of view which is not related to GDR. In this case, you may want to consider phase ripple amplitudes in [rad] instead of GDR amplitudes. Thus, equation (3) has to be modified slightly:

$$\varphi(f) = -\sum_{i=1}^{N} a_{r,i} \cdot \sin\left(\frac{2\pi f}{f_{r,i}} + \varphi_{offset,i}\right). \tag{4}$$

Here, $a_{r,i}$ denotes the phase ripple amplitude in [rad] for each distinct phase ripple period.

The PRE may be used in two different modes: You may either generate a phase ripple by specifying the GDR amplitude(s) or by specifying the phase ripple amplitude(s). For each mode, you can either specify a single period ripple or a ripple with multiple period components. However, currently you may only import multiple period components by loading them from an external file.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	numberOfRippleComponents
Description	Number of phase ripple components
Default Value	1
Parameter Range	$1 \leq$ number $\leq$ 1e6
Unit	

Parameter Name	ripplePeriods
Description	vector containing the ripple period(s) for each distinct ripple component
Default Value	100
Parameter Range	0 < period
Unit	GHz

Parameter Name	GDRAmplitudes
Description	vector containing the ripple GDR amplitude(s) for each distinct ripple component
Default Value	5
Parameter Range	0 < GDRAmplitudes
Unit	ps

Parameter Name	phaseAmplitudes
Description	vector containing the ripple phase amplitude(s) for each distinct ripple component
Default Value	5
Parameter Range	no restrictions
Unit	rad

Parameter Name	rippleOffsets
Description	vector containing the ripple offset(s) for each distinct ripple component
Default Value	0
Parameter Range	no restrictions
Unit	rad

Parameter Name	f0
Description	center frequency for the ripple creation
Default Value	193.1
Parameter Range	within simulation bandwidth
Unit	THz

Parameter Name	importFromFile
Description	shall the ripple data be imported from an external file?
Default Value	false
Parameter Range	boolean

Parameter Name	rippleComponentInputFIle
Description	file from which the ripple data shall be loaded
Default Value	none
Parameter Range	valid file path

Parameter Name	saveToFile
Description	shall the ripple data be saved to an external file?
Default Value	false
Parameter Range	boolean

Parameter Name	rippleComponentOutputFIle
Description	file to which the ripple data shall be saved
Default Value	none
Parameter Range	valid file path

References

[1] J. Mietzner and S. Otte, “Optimal Equalization of Distortions due to Group Delay Ripples of Chirped Fiber Bragg Gratings (CFBG)”, Intern. J. of Electronics and Communications, Vol. 56, No. 3, pp. 187-192, 2002.

[2] Matthias Westhäuser et al, “Reducing Fiber Bragg-Grating induced group delay ripples in 112 Gbit/s metro networks using generically initialized transversal filters”, ITG-Fachtagung “Photonische Netze”, Leipzig, Mai 2010.

[3] Matthias Westhäuser et al, “Optimization of Optical Equalization of Group Delay Ripple-Induced Penalties from Fiber Bragg Gratings in 112 Gbit/s Metro Networks”, Optics and Photonics Congress “Advanced Photonics”, Karlsruhe, Germany, June 2010.

7.10.7 Phase Shifter

Fundamentals

The phase shifter adds a constant phase Δp to the input signal $A_{in}(t)$. All signals are assumed to be in a complex baseband representation. The output signal is given by

$$A_{out}(t) = A_{in}(t) \exp\left[j \frac{2\pi}{360} \Delta p\right] \tag{1}$$

A_{out} :	shifted output signal
A_{in} :	input signal
Δp :	range of phase sift

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	ps
Description	range of phase shift
Default Value	45
Parameter Range	not limited
Unit	degree

References

none

7.10.8 Phase Synchronizer

Fundamentals

The Phase Synchronizer is a component which allows adapting the phase of a target signal to the phase of a source signal. For the phase evaluation the Viterbi & Viterbi algorithm [1,2] is applied.
The functionality is helpful, especially when creating a coherent receiver. In a coherent receiver the data signal is superimposed with a local oscillator (LO). For optimum sensitivity both signals should be in phase. This can be arranged using the Phase Synchronizer by connecting the LO to the upper input port and the data signal to the lower input port. The output will return the data signal shifted by the phase difference between LO and data signal. Now the two signals can be superimposed e.g. in a 3dB coupler.

Input / Output

	Number	Type of signal
Input	1	Optical field
Input	2	Optical field
Output	1	Optical field (Phase Synchronized)

Component Parameters

Parameter Name	Phase States
Description	Bits per symbol in phase coded signals (2 for BPSK, 4 for QPSK, . . .)
Default Value	4
Parameter Range	0 < Phase States

Parameter Name	Pol Axis
Description	Polarization axis for measurement of phase difference. (0 = horizontal polarization, 1 = vertical polarization)
Default Value	0
Parameter Range	0, 1

References

[1] A. J. Viterbi, A.M. Viterbi, “Nonlinear Estimation of PSK-Modulated Carrier Phase with Application to Burst Digital Transmission,” IEEE Trans. Info. Theory, vol. 29, 543-551, 1983.

[2] J.C. Rasmussen, T. Hoshida, H. Nakashima, “Digital Coherent Receiver Technology for 100-Gb/s Optical Transport Systems,” Fujitsu Sci. Tech. J., Vol. 46, No. 1, pp. 63-71, 2010.

7.10.9 Wavelength Converter

Fundamentals

The wavelength converter ideally shifts the carrier frequency of the input signal by fs.

Input / Output

	Number	Type of signal
Input	1	Optical field
Output	1	Optical field

Component Parameters

Parameter Name	fs
Description	frequency shift
Default Value	0.2
Parameter Range	0 ≤ fs ≤ Frequency Range
Unit	THz

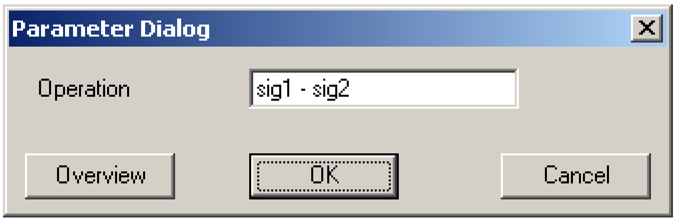
References

none

7.11 Programming

7.11.1 Calculator

Fundamentals



Parameter Dialog

Operation: sig1 - sig2

Figure 7.11.1. Calculator Parameter Dialog

The calculator component serves to perform universal mathematical operations with the input signals. Structures such as balanced receivers may easily be implemented with this component. According to the number of input ports, the input signals are labeled sig1, sig2, sig3, ... Derivations of the input signals are labeled dsig1, dsig2, dsig3, . . . The operations are specified with the dialog in figure 7.11.1.

Input / Output

	Number	Type of signal
Input	1	Optical Field / Electrical signal
Output	1	Optical Field / Electrical signal

Component Parameters

Parameter Name	Operation
Description	Operation that the component will perform on the input signals
Default Value	not applicaple
Parameter Range	not applicaple

References

none

7.11.2 Command Line Interface

Fundamentals

PHOTOSS includes a Command Line Interface for the integration of user-defined algorithms. The Command Line Interface allows executing arbitrary programs from a command line. It complements the MATLAB® interface to interact with any user-defined program. The interaction between PHOTOSS is enabled by two files, which export the sampled data from PHOTOSS and read the modified contents afterwards.

Component Parameters

In the *Command line* parameters an arbitrary command can be given, which is executed after the file is written. The *Write to File* parameter contains the file, which contains the exported data from PHOTOSS. The data contained in the file may be modified by the external program and written back to another file. This file (defined by the parameter *Read from file*) is imported into PHOTOSS afterwards. The *Wait* button determines, whether PHOTOSS shall pause until the command line program has terminated or not.
The syntax of the data file can be set by the parameter *File Format*. PHOTOSS means, that the File-Loader/-Saver format of PHOTOSS-Versions prior to 4.10 is used. When the parameter is set to TEXT, the new simplified text format is used.
A text format data file consists of name-value-pairs, vectors, matrices and comments.

Name-value-pairs are written as:

```
Name=Value
```

Vectors are written as:

```
Bit Pattern Vector    % Name of vector
Size=64          % Size of vector
0         % Elements of vector (one per line)
0
0
```

Matrices are written as:

```
Sampled Signal Matrix   % Name of matrix
Number of Rows=4096     % number of rows
Number of Columns=3     % number of columns
% Sample Values (Time, Abs of Pol-Axis 1, Arg of Pol-Axis 1, Abs of Pol-
    Axis 2, Arg of Pol-Axis 2)  % Comment containing column headers (
    optional)
0 0 0     % Elements of matrix (one row per line)
0.390625  0 0
0.78125 0 0
1.171875  0 0
1.5625  0 0
```

The astute reader may have noticed that comments are denoted by a preceding %.

	Number	Type of signal
Input	1	Optical Field / Electrical Signal
Output	1	Optical Field / Electrical Signal

Input / Output

Component Parameters

Parameter Name	Command line parameterized
Description	Command to be executed during parameterized run
Default Value	none
Parameter Range	valid CMLI command

Parameter Name	Command line sampled
Description	Command to be executed during sampled run
Default Value	none
Parameter Range	valid CMLI command

Parameter Name	Write to file
Description	File containing sampled data
Default Value	none
Parameter Range	valid file path

Parameter Name	Read from file
Description	File containing modified data to be imported
Default Value	none
Parameter Range	valid file path

Parameter Name	Wait
Description	Wait for command line process to terminate
Default Value	true
Parameter Range	boolean

Parameter Name	File Format
Description	Determines the file format that is used for the data file
Default Value	.txt

Results

On the result page of the Command Line Interface the user may specify results. These can be used, to pass data from the command line application to PHOTOSS.
Assuming a result with the name *myResult* is specified, PHOTOSS does the following:
When the data file is read, PHOTOSS searches the file for a name-value-pair with the name *myResult*. More specifically, it searches for a line that contains $myResult = X$, where X is the value of the result. If a result is specified but cannot be found in the data file, PHOTOSS will print a warning to its logging window.

References

none

7.11.3 DPSK Decoder

Fundamentals

The DPSK decoder adjusts the bit pattern of the parametric signal representation at the receiver if a DPSK signal was employed. In this case, different bit patterns are present in the channel so that the parameterized representation has to be modified at the receiver. The DPSK decoder conducts the operation depicted in figure 7.11.2 on the incoming bit pattern.

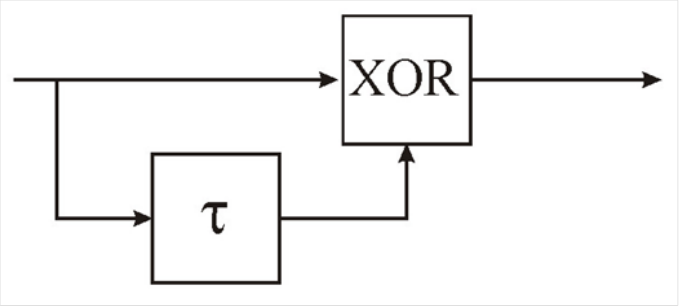


Figure 7.11.2. DSPK decoder block diagram

In contrast to the parametric signal representation, the sampled signal remains unchanged.

Input / Output

	Number	Type of signal
Input	1	Optical Field / Electrical signal
Output	1	Optical Field / Electrical signal

References

none

7.11.4 MATLAB®

Fundamentals

PHOTOSS includes a MATLAB® programming interface for the integration of user-defined algorithms. MATLAB® is a software package for numerical calculations that is developed and distributed by The MathWorks (http://www.mathworks.com). MATLAB® is not part of PHOTOSS and is not included in the PHOTOSS package. In order to use the programming interface, you must have MATLAB® installed on your computer. Please refer to the MATLAB® documentation for details on the installation of MATLAB®. PHOTOSS currently supports the MATLAB® versions which are included in The MathWorks Releases 2006a to 2013a. PHOTOSS will automatically detect and use the latest version of MATLAB® that is installed on your computer.

⚠ **You need to run PHOTOSS with administrator privileges in order to use it with MATLAB® 7.0 prior to Service Pack 1. This is caused by MATLAB® and does not apply to 6.x versions. The issue is fixed with MATLAB® 7.0 Service Pack 1.**

In rare cases you may receive an error message saying that `PHOTOSS cannot run MATLAB` even though it is installed. Please check if the MATLAB® directory is contained in the list of default search paths. You can display the path list e. g. by entering *path* in a command line window. If the MATLAB® directory is not listed, your installation of MATLAB® is probably broken or incomplete. Re-install MATLAB® to fix the problem.
This problem also occurs if MATLAB® has not been registered during its installation. You can manually do this by typing `matlab /regserver` on the command line.

Component Parameters

Parameter Name	component parameter file
Description	MATLAB® file containing the model parameters
Call	called before the parameterized analysis initialization

Parameter Name	power budget initialize file
Description	MATLAB® file with data that needs to be initialized before the parameterized run
Call	called before the parameterized analysis

Parameter Name	power budget run file
Description	MATLAB® file with data that needs to be initialized during the parameterized run
Call	performs the parameterized analysis

Parameter Name	sample initialize file
Description	MATLAB® file with data that needs to be initialized before the sampled run
Call	called before the sampled analysis

Parameter Name	sample run file
Description	MATLAB® file with data that needs to be initialized during the sampled run
Call	performs the sampled analysis (repeatedly called for each signal block)

Parameter Name	stop on MATLAB® error
Description	If set, an error that occurs while running the MATLAB® script (such as using an undefined variable) will cause a PHOTOSS error that results in stopping the simulation

Parameter Name	halt engine
Description	Can be set to prevent the MATLAB® engine from automatically terminating and freeing its memory when the simulation is finished. This is especially helpful for reviewing MATLAB® figures generated during the simulation, as these will be deleted otherwise.

Parameter Name	InPorts / OutPorts
Description	Number of input and output ports of your MATLAB® component.
Parameter Range	0 ≤ Ports

Custom parameters page:

Here you may define custom parameters, which are passed to MATLAB® before your scripts are run. You may add a variable by clicking the Add parameter-Button. To change the name of a parameter, just click on the column Name of the according parameter. Value and type of a parameter may be changed in the same way. Let the name of your parameter be `My_Parameter`. It can be accessed in MATLAB® by the same name `My_Parameter`. Note that the custom parameters can only be used for passing data from PHOTOSS to MATLAB®. If you want to transfer information from MATLAB® to PHOTOSS, then use the results of the MATLAB® component, which are explained in the next paragraph.

Results page:

You may define your own results, which are transferred from MATLAB® to PHOTOSS after your scripts have finished. The results may be used like results generated by other components (see section *Parameter Variation* on page 71). You can access a result named `My_Result` by typing `Results.My_Result` in MATLAB®. The results are not available when you use the MATLAB® Interface component as part of a receiver in a path analysis simulation (see section Path Analysis).

Integration of MATLAB® Components into PHOTOSS

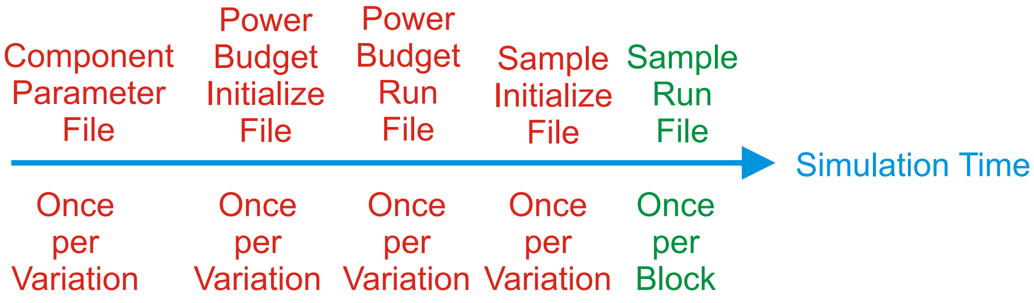


Figure 7.11.3. Time Diagram of a PHOTOSS simulation run with a MATLAB® component.

The following steps are carried out during simulations that involve MATLAB® components (compare fig. 7.11.3):

1. PHOTOSS opens a MATLAB® engine and passes general data to MATLAB®, e. g. simulation parameters, global variables, and some information about the input signals.
2. The component parameters are loaded from the *component parameter file*.
3. PHOTOSS passes some more data. Afterwards, the MATLAB® engine executes the *power budget initialize file*.

4. The MATLAB® engine executes the *power budget run file* in order to calculate the parameterized output signals.

5. PHOTOSS passes some more data. Afterwards, the MATLAB® engine executes the *sample initialize file*.

6. The MATLAB® engine executes the *sample run file* in order to calculate the sampled output signals. This step is repeated for each signal block of the simulation.

When the MATLAB® component is used as part of a path analysis receiver, steps 4 and 6 are also repeated for each path step.

Programming MATLAB® Components

Step 1 - Your Parameters

Just like the built-in PHOTOSS components, your MATLAB® based components may have a specific set of customizable parameters. There are two ways to specify these parameters. You may use a *component parameter file*. This is a MATLAB® script, which just introduces your variables. It is executed once per simulation, at the beginning of the first calculation of the component. Write this file as a MATLAB® script like the following example.

```
% Component parameter file example:
global inputs % Make the variable available to all MATLAB functions
inputs = 1; % Will set the number of inputs
global bandwidth
bandwidth = 2.024; % Component specific bandwidth
global txt
txt = ’Hello Word’ % My comment
global myarray
myarray = [1 , 23 ,0.5 , 0] % Constants
```

As all MATLAB®(.m) files are executed in the same engine, it is not mandatory to declare your variables global, even when you initialize them in subroutines. However, we recommend declaring them global in order to clarify their respective scope.

The second way to specify your parameters is to use the *Custom Parameters* property page. To get to this property page, just double click the MATLAB® component.

Step 2 - The Initialization Functions

MATLAB® based components involve two initialization functions, the *power budget initialize file* and the *sample initialize file*. The first is intended to initialize parameters and settings for the parametric signal analysis, the second is intended do initialize the sampled signal analysis. As the *sample initialize file* is executed only once per simulation, you should use it to reserve memory (if needed), and to calculate values that remain constant for the entire simulation including all signal blocks, e. g. static properties such as transfer functions.

Step 3 - The Run Functions

The *power budget run file* is intended to perform the parametric signal analysis. The corresponding input and output signals are described in section *Simulation Parameters* (*channel parameters paragraph*). The *sample run file* is intended to perform the sampled signal analysis. It is repeatedly called for each signal block.

Access Parameters from MATLAB®

While the MATLAB® Workspace provides comfortable means to review the data that PHOTOSS passes to MATLAB®, some information is still missing (e. g. the units of the data values). Therefore, the following section provides an overview on the variables that you may access in user defined MATLAB® files along with their respective units and descriptions.

current_block_no = 42;	Number of the current signal block (0 = first block)
PHOTOSS_IM	internal parameter (must not be overwritten)
PHOTOSS_RE	internal parameter (must not be overwritten)
PHOTOSS_SIZE	internal parameter (must not be overwritten)
i	internal parameter (must not be overwritten)

The *main_parameters* structure contains the simulation parameters. While you can change the values in your MATLAB® files, this will not affect any subsequent components. The following list includes MATLAB® variables, their respective default values, and the corresponding PHOTOSS simulation parameters.

DoPowerBudget = 1;	Perform parameterized analysis (0 = no, 1 = yes)
DoSampledCalculation = 1;	Perform sampled signal analysis (0 = no, 1 = yes)
runParameterizedAndSampledTogether = 0;	Perform a combined simulation run (0 = no, 1 = yes)
BidirectionalFlow = 0;	(reserved for future use)
convolution(:) = ’CYCLIC’;	Convolution method (’LINEAR’, ’CYCLIC’)
RandomGenerator(:) = ’STATISTICAL_RNG’;	Initialization of the random generator
	(’DETERMINISTIC_RNG’, ’STATISTICAL_RNG’)
method(:) = ’M_TOTAL_FIELD’;	Analysis method:
	’M_TOTAL_FIELD’ = Total field
	’M_SEPARATE_CHANNELS’ = Separated channels
polPlanes = 1;	Number of polarization planes (1 or 2)
Number_Blocks = 1;	Number of blocks
BlockSize = 4096;	Samples per block
bandwidth = 2.56;	Frequency range [THz]
f0 = 193.1;	Center frequency [THz]
fstep = 0.000625;	Frequency slice [THz]
ref_bitrate = 40	Reference bitrate [Gbit/s]
signalBandwidth = 0.0125;	Signal bandwidth (for OSNR calculation) [THz]
tstep = 0.390625;	Sampling time [ps]
noiseBandwidth = 8;	Noise bandwidth [THz]
noise_fstep = 0.004;	Noise slice [THz]
noise_threshold_dBm = -58;	Noise threshold [dBm]
BlockTime = 1600;	Block time [ps]
noisehandling(:) = ’ANALYTIC’;	Noise handling (’NUMERIC’, ’ANALYTIC’)
noiseSamples = 2000;	Noise samples

The *Global_Parameter_List* vector contains the global variables. While you can change the values in your MATLAB® files, this will not affect any subsequent components. Global variables are composed of their *name*, *unit*, and *value*. If being varied in a parameter variation simulation, an additional *data_vector* element contains the corresponding variation values. As an example the command *Global_Parameter_List(1)* might return:

```
name: 'r'
unit: ''
value: 0.8000
data_vector: [0.4000 0.5000 0.6000 0.7000 0.8000]
```

The above output means that the variable r has no unit, its value is varied from 0.4 to 0.8 with a step size of 0.1, and the current value is 0.8 (so we are now in the last iteration a parameter variation simulation).

MultiSignal is an array of structures that contains the input signals of the component. For a component with n inports, the structure is `MultiSignal(1,n)` These variables and default values are included:

name(:) = 'component output signal';	Signal name (component names the preceeding component which generated this input signal)
parameterizedisReady = 1;	Parameterized analysis is already done (0 = no, 1 = yes)
sc_signals(1,m)	Structure of sc_signals parameters (see below) for *each* channel m: number of channels (= 1 in Total field mode)
channelData	Structure of channelData parameters (see below) There is only one channelData structure which holds the data of *all* channels

sc_signals is an array of structures that contains the sampled signals of a *MultiSignal*. Each channel has a separate substructure and can be accessed by addressing the second dimension of the array.
E.g. `test = MultiSignal(1,1).sc_signals(1,3)` would access the third channel substructure. The following variables and default values are included in this structure:

band_width = 2.56;	Frequency range [THz]
block_size = 4096	Number of samples per block
comp(n,m) = Real(Sample) + i * Im(Sample);	Complex signal values [sqrt(W)] Available only during the sampled run n: Number of polarization plane (1 or 2). m: Number of samples Example: Addressing the 5th sample of the 2nd polarization plane for channel 3: `test = MultiSignal.sc_signals(1,3).comp (2,5);`
domain = 'TIME';	Time or frequency domain ('TIME', 'FREQ')
f0 = 193.1;	Center frequency [THz]
frequ_step = 0.000625;	Frequency slice [THz]
pol_planes = 1;	Number of polarization planes (1 or 2)
time_step = 0.390625;	Sampling time [ps]
unit(:) = 'SQRT_WATT';	Signal unit ('AMPERE', 'VOLT', 'SQRT_WATT', 'WATT')

channelData is a structure that contains the parametric signals of a *MultiSignal*. *channelData* and its substructures are only available during the parameterized run and each substructure contains the data for each individual channel. When using the analytical performance evaluation, there is only one analytical noise vector for all channels, since the vector spans over the whole frequency band of the simulation bandwidth. Presently, only one noise vector is used for the representation of noise for *both* polarization axes, please refer to chapter Noise Models on page 56 for more details. The following variables and default values are included:

channels(1,n)	Structure of channels parameters (see below)
  has_noise = 0;	n = 1 for Total field, n = number of channels for separated channels Consider noise (0 = no, 1 = yes)
noise_vec(1,m) = 0;	Noise vector [W / THz] identical for all channels m: Number of noise samples

channels is an array of structures that contains the data for each individual parametric signal channel. These variables and default values are included:

ana_per_info	Structure of ana_per_info parameters (see below) n: number of channels
data	Structure of data parameters (see below)
modulation_pointer = 1.88532e+007	memory pointer (for internal storage, do not change!)
delay = 0;	Signal delay [ps]
freq = 193.1;	Channel carrier frequency [THz]
inverted = 0;	Inverted bit sequence (0 = no, 1 = yes)
p_max = 2.505e-4;	Maximum power [W]
p_avg = 1.2e-4	Average power [W]
p_min = 0;	Minimum power [W]
unit(:) = 'OPTICAL';	Unit ('ELECTRICAL', 'OPTICAL')
DL = 0;	Accumulated linear dispersion ($D\cdot$ fiber length)
SL = 0;	Accumulated parabolic dispersion ($S\cdot$ fiber length)
Theta = 0;	Polarization angle [degree]

ana_per_info is a structure that contains data related to the analytic performance evaluation. These variables and default values are included:

g_edfa = 1;	Amplifier gain [no unit]
elec_rec_gain = 1;	Electrical gain of the receiver [no unit]
f_step = 0;	frequency step [THz]
F_z = 1;	Additional noise figure [no unit]
i_dark = 0;	Dark current of photo diode [A]
isinit_elec_filter = 0;	Is electrical filter initialized? (0 = no, 1 = yes)
M_0 = 0;	Avalanche factor [no unit]
neb_elec = 0;	Noise equivalent bandwidth [Hz]
noise_isinit_diode = 0;	Is diode noise initialized (0 = no, 1 = yes)
noise_isinit_int = 0;	Is noise initialized (0 = no, 1 = yes)
noise_isinit_preamp = 0 ;	Is preamplifier noise initialized? (0 = no, 1 = yes)
optical_filter_isinit = 0 ;	Is optical filter noise initialized? (0 = no, 1 = yes)
par_isinit_diode = 0 ;	Are diode parameters initialized? (0 = no, 1 = yes)
par_isinit_pre_amp = 0 ;	Are preamplifier parameters initialized? (0 = no, 1 = yes)
pre_amp_receiver = 0 ;	Pre-amplified receiver? (0 = no, 1 = yes)
R = 1;	Responsivity of the diode [A/V]
status = 0;	Signal type:
	0 = optical signal
	1 = signal after pin diode or APD
	2 = signal after electrical filter
	>2 = signal after more than one electrical filter
thermal_noise = 0;	Thermal noise [A^2]

data is a structure that contains signal modulation parameters. These variables and default values are included:

cw_signal = 0;	Continuous wave? (0 = no, 1 = yes)
AmplitudeMode = 1;	Amplitude mode:
	0 = sqrt(Amplitude), 1 = Amplitude, 2 = Power
BitCode = 0;	Bit coding:
	0 = none, 1 = duobinary, 2 = AMI,
	3 = phase switch, 4 = amplitude switch
bitrate = 40;	Bitrate [Gbit / s]
duty_cycle = 1;	Duty cycle (0...1)
Methode = 0;	Pulse shaping method (0 = direct, 1 = filter)
ModFormat=0;	Modulation format (0 = ASK, 1 = DPSK)
pattern(23) = 0;	Bit pattern
PulseCode = 1;	Pulse code format (0 = RZ, 1 = NRZ)
Shape = 1;	Pulse shape:
	0 = rectangle, 1 = cos2, 2 = sin, 3 = sech (soliton),
	4 = Gaussian, 5 = triangle, 6 = saw up, 7 = saw down

There is an additional message variable, which allows you to pass message strings from MATLAB® to PHOTOSS. If a string is assigned to this variable, it is printed to the output window in PHOTOSS.

message	is printed to output window

Path Analysis Parameters:

These parameters are available in a path analysis simulation:

AbsolutePathPosition	Current position within the path [km]
LastPathPosition	Length of the path [km]
Additional_Parameter_List.PathFrequency	Frequency to be analyzed [THz]

Examples how to use the variables:

main_parameters.signalBandwidth = 0.26	Changes the frequency range.
MultiSignal(1,2).sc_signals(1,10).comp(1,32) = 0	Sets the 32th sample of the 10th signal of the second output for the first polarization plane to zero.
MultiSignal(1,1).channelData.channels(1,3).freq	Frequency of the 3rd channel of the 1st signal.

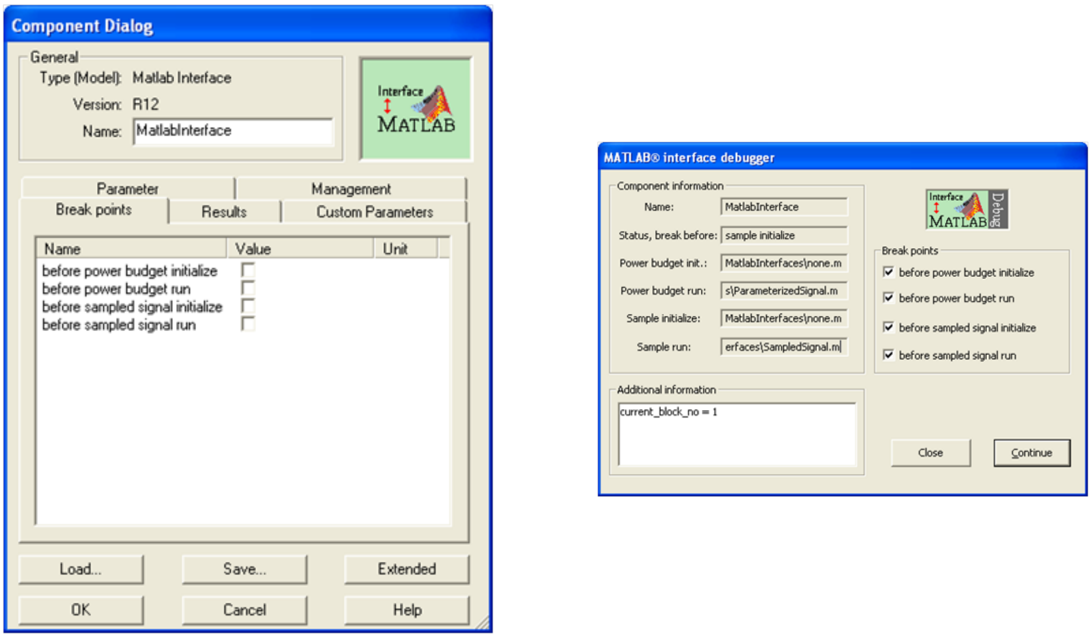
Important Remarks

⚠ **Most data is transferred from PHOTOSS to MATLAB® more than once. This implies that if you change affected data, you may need to redo your changes after each transfer as the data will be overwritten with the respective original values.**

⚠ **Prior to the execution of the power budget initialize file (step 3), MultiSignal is transferred including the channelData, but excluding the sampled signal (comp). The same data is also transferred before the execution of the power buget run file (step 4).**

Prior to the execution of the sample initialize file (step 5), MultiSignal is transferred including the sampled data (comp), but excluding channelData. The same data is also transferred before the execution of the sample run file (step 6).

Debugging MATLAB® Functions



Component Dialog

General
Type (Model): Matlab Interface
Version: R12
Name: MatlabInterface

Name	Value	Unit
before power budget initialize	□	
before power budget run	□	
before sampled signal initialize	□	
before sampled signal run	□	

MATLAB® interface debugger

Component information
Name: MatlabInterface
Status, break before: sample initialize
Power budget init.: MatlabInterfaces\none.m
Power budget run: s\ParameterizedSignal.m
Sample initialize: MatlabInterfaces\none.m
Sample run: erfaces\SampledSignal.m

Break points
☑ before power budget initialize
☑ before power budget run
☑ before sampled signal initialize
☑ before sampled signal run

Additional information
current_block_no = 1

Figure 7.11.4. MATLAB® Component Dialog (left) and MATLAB® Interface Debugger (right)

User defined MATLAB® functions are most effectively debugged directly in MATLAB®. As this approach requires access to same environment that is present during a PHOTOSS simulation, the debugging is carried out from within a running simulation.

To set up a debug simulation run, open the Break points page of the component parameter dialog and set the breakpoint value for one or more of the displayed MATLAB® files to true. This will cause the simulation to pause and open a MATLAB®-Interface Debugger dialog just before the corresponding MATLAB® file is executed.

This dialog provides an overview of which component and setting caused the break.

The Additional information section shows the exact position in the current simulation. The Break points section lets you modify the breakpoint settings for the current MATLAB® component. However, these changes are temporary and will only last until the current simulation is finished.

In order to debug the current MATLAB® function, you may now run it manually from a MATLAB® command window while the PHOTOSS simulation is paused.

For using the MATLAB® Debugger, the function must be opened with the same MATLAB® engine that is used to run the function.

If more than one MATLAB® engine is active, compare the *Additional information* with the corresponding values in the active engines to find out which one belongs to the current component.

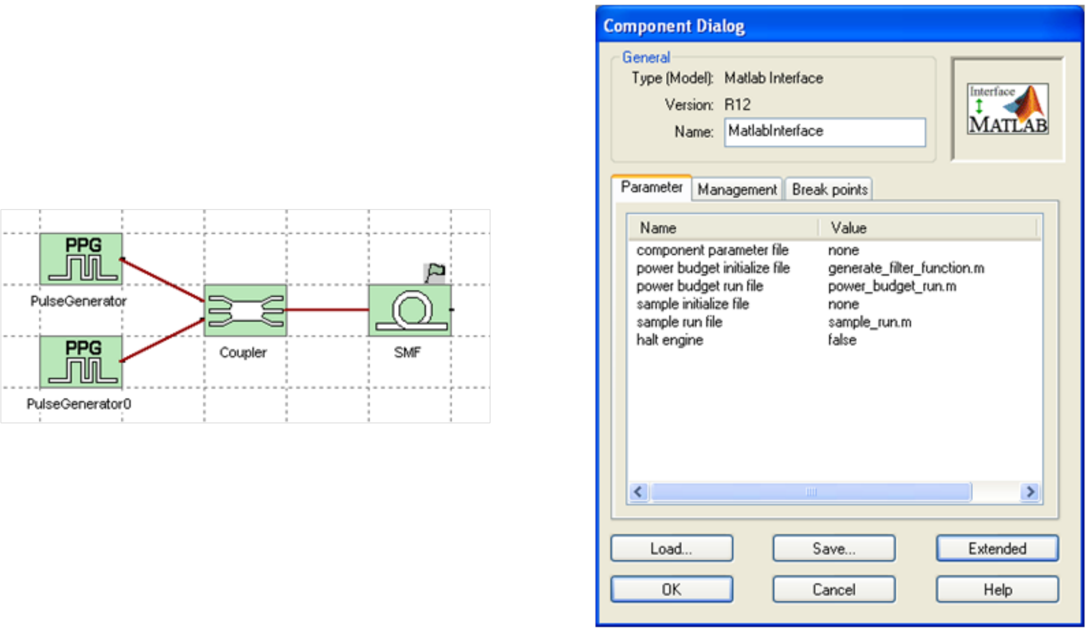
After debugging is finished, press *Continue* to resume the simulation until it reaches the next breakpoint. You may also close the dialog with *Close* and then continue or abort the simulation via the corresponding options in the menu or the simulation toolbar.

Additional Remarks:

- During the testing and debugging of your MATLAB® functions, the original data that PHOTOSS transferred to MATLAB® might get corrupted, so that you cannot continue debugging. The PHOTOSS Examples folder contains some useful functions to recover from such a situation. Use *make_copy.m* to generate a copy from the data, and *get_copy.m* to restore the data from your copy.
- Before running one of the following functions from the PHOTOSS Examples folder, you need to define a variable *cn* and set it to the number of channels to be used:
  - *transform_to_frequency.m* transforms the selected sampled signal from the time domain to the frequency domain.
  - *transform_to_time.m transforms* the selected sampled signal from the frequency domain to the time domain.
  - *plot_sampled_signal.m* plots the selected sampled signal

You may copy these functions to your function folder to make your debugging easier. Feel free to adapt them to your needs.

Example Simulation



Type (Model): Matlab Interface
Version: R12
Name: MatlabInterface

Name	Value
component parameter file	none
power budget initialize file	generate_filter_function.m
power budget run file	power_budget_run.m
sample initialize file	none
sample run file	sample_run.m
halt engine	false

Figure 7.11.5. Example Simulation Overview (left) and MATLAB® Component Dialog (right)

This section discusses an example simulation in order to prevent some common mistakes and misleading interpretation of results. The example is taken from the PHOTOSS Examples folder:

path_analysis_type_d_const_power.pho.

It involves a path analysis simulation using a type D receiver with an EDFA preamplifier (see section *Path Analysis* on page 107 for details).

The MATLAB® based receiver component implements an optical filter for separating the signals of the two sources. As the filter transfer function will remain constant throughout the simulation, it is calculated in the *power budget initialize file* (because this function will be called only once).

The following source code abridgment illustrates how to generate a filter for each frequency of interest by using the *PathFrequency* parameter.

```
% power budget initialization file:
f0 = Additional_Parameter_List.PathFrequency; % center frequency [Thz]
delta_f = 0.082; % set 3 dB bandwidth to 82 GHz
f_min = main_parameters.f0-main_parameters.bandwidth/2;
f_step = main_parameters.fstep;
f_max = f_min+(main_parameters.BlockSize-1)*f_step;
H = gaussian_filter(f_min,f_step,f_max,f0,delta_f);
```

In a path analysis simulation, the *power budget run file* is called for each path step. Therefore, you need to save copies of the *channelData* information that you'll require in the subsequent sampled signal calculation in order to prevent it from being overwritten upon the next call of the *power budget run file*. You can do so e. g. like this:

```
% power budget run file:
%Determine the vector position of result to save
if (AbsolutePathPosition == 0)
  counter = 1; %first call of the funciton
else
  counter = counter+1;
end;
results(counter,2) = param_pow;
```

In order to save your results in the last run of the *sample run file*, you can use the *LastPathPosition* parameter, e. g. like this:

```
% sample run file:
if (AbsolutePathPosition == LastPathPosition) %Last function call
  %Save results
  save(['matlab_results_'
      num2str(Additional_Parameter_List.PathFrequency) '.txt'],
      'results', '-ASCII');
end;
```

Figure 7.11.6 to the right shows the power of both the sampled (blue) and the parameterized (red) signal representations. Note that the average power of the sampled signal is not exactly 6.6 mW as predicted by the parameterized analysis. This is because of the phase interference between the two transmitted channels, which is ignored by the parameterized approach but included in the sampled signal analysis.

Keep in mind that the parameterized analysis yields approximate results that may or may not be accurate enough for use in the sampled calculations. Check the assumptions made by the parameterized analysis whenever you use *channelData* values.

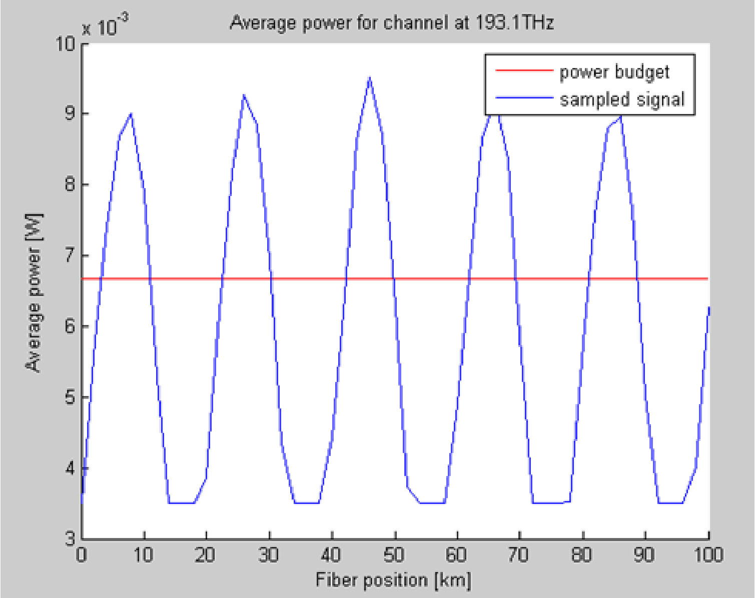


Figure 7.11.6. MATLAB® Example Graph

References

none

7.11.5 Pattern Generator Fundamentals

In contrast to a simple amplitude on-off-keying scheme, some modulation formats need a certain amount of preprocessing. In differential modulation schemes -like the DQPSK for example- the bit pattern needs to be differentially precoded.
To be able to cope with these kinds of modulation schemes the pattern generator is designed to create bit sequences that are suitable for a variety of such formats. For all encoding schemes the component writes the original and precoded bit patterns to according files. These files can then be loaded into the pulse generator and pattern modifier.
Since this preprocessing of bit patterns is only the first part of the signal generation, some of the transmitters may consist of a variety of other components. Examples of the most common implementations are stored in the user defined models.

Supported Formats

DQPSK - PRBS:

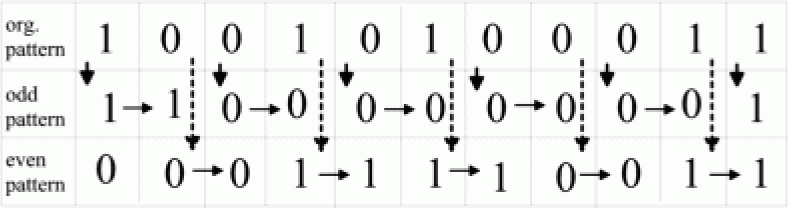


Figure 7.11.7. Separation in odd and even bit patterns

By choosing the DQPSK format, an initial single bit pattern is generated. This pattern is then splitted in two parts. One bit pattern consists of all the odd bits from the original pattern and the other one of its even bits. The scheme for this operation is shown in Figure 7.11.7.
These two bit patterns are used as the input for the actual precoder, as can be seen in the figure below. The splitting in odd and even bits guarantees that at the input only one signal may change at a time.

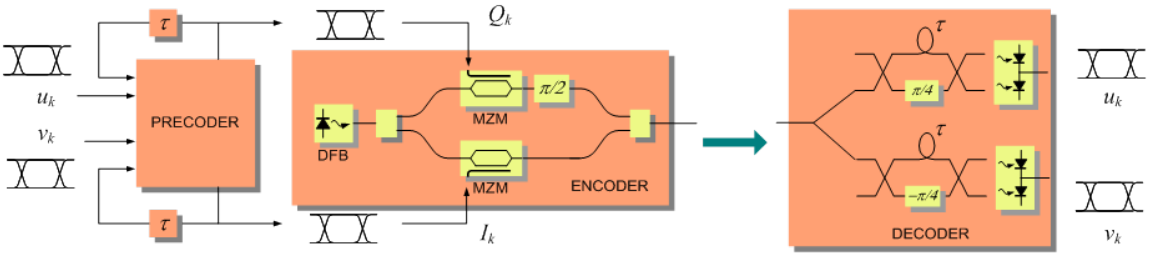


Figure 7.11.8. Schematic of the DQPSK precoder, transmitter and receiver 1

The precoder uses an algorithm to link the two different patterns. The employed algorithm is shown below.

$$\begin{aligned} I_k &= (Q_{k-1} \oplus I_{k-1})\ (v_k \oplus I_{k-1}) + (Q_{k-1} \oplus I_{k-1})\ (u_k \oplus I_{k-1}) \\ Q_k &= (Q_{k-1} \oplus I_{k-1})\ (u_k \oplus I_{k-1}) + (Q_{k-1} \oplus I_{k-1})\ (v_k \oplus I_{k-1}) \end{aligned} \qquad (1)$$

Here I_k and Q_k represent the precoded inphase and quadrature channel bit patterns of the DQPSK signal, I_{k-1} and Q_{k-1} the precedent precoded bits and u_k and v_k the original bit patterns.

DQPSK - deBruijn:

In this mode the pattern generator will create a deBruijn sequence with an alphabet of $m = 4$ symbols (0, 1, 2 and 3). With this alphabet a unique pattern is generated. Every possible symbol combination for a given pattern length n has to be part of this unique pattern. The combination of m and n determines the length of this unique sequence. Since $m = 4$ PHOTOSS has to determine the maximal value of n to create an optimal deBruijn sequence. The length of a deBruijn sequence is equal to m to the power of n. Therefore unique deBruijn sequences in PHOTOSS are powers of four (16, 64, 256,...). If the block length differs from that, PHOTOSS will generate shorter unique sequences and concatenate them to match the block length.
The first digit of each symbol of the alphabet corresponds to the I channel and the last one to the Q channel of the encoded DQPSK signal. From this encoded bit sequences the unencoded sequences are then calculated by applying

$$v_k = \left(\left(\overline{I_{k-1}} \cap \overline{Q_{k-1}} \cap \overline{I_k}\right) \cup \left(\overline{I_{k-1}} \cap Q_{k-1} \cap Q_k\right)\right) \cup \left((I_{k-1} \cap Q_{k-1} \cap I_k) \cup \left(I_{k-1} \cap \overline{Q_{k-1}} \cap \overline{Q_k}\right)\right) \qquad (2)$$

$$u_k = \left(\left(\overline{I_{k-1}} \cap \overline{Q_{k-1}} \cap \overline{Q_k}\right) \cup \left(\overline{I_{k-1}} \cap \overline{Q_{k-1}} \cap \overline{I_k}\right)\right) \cup \left((I_{k-1} \cap Q_{k-1} \cap Q_k) \cup \left(I_{k-1} \cap \overline{Q_{k-1}} \cap I_k\right)\right) \qquad (3)$$

The fact that the encoded bit sequence is directly generated gives a better control over the properties of the signals within the fiber.

DUOBINARY:

In order to generate a duobinary signal, the bit pattern has to be precoded, following a certain logic This algorithm can be seen in Figure 7.11.9, where $d(k)$ represents the original bit pattern and $b(k)$ the precoded one.

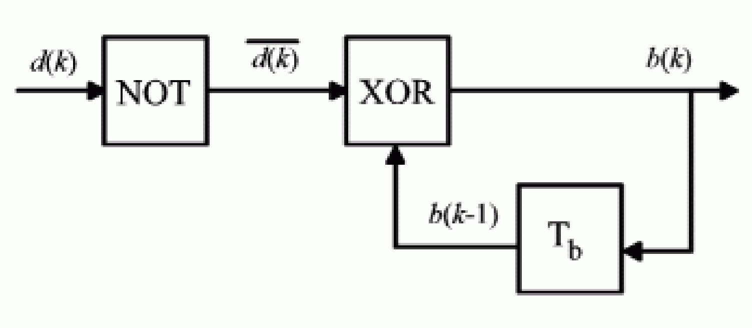


Figure 7.11.9. Duobinary precoder logic

Parameters of the bit pattern generation

Bit sequence:

- *Bit_Rate*: The bit rate f_{Bit} in Gbit/s. The bit duration is derived by $T_{Bit} = 1/f_{Bit}$.
- *Pattern_generation:* Determines whether a new random bit sequence is generated or if the bit sequence is read from an existing bitfile that contains a user defined bit pattern of arbitrary length. In the latter case, the *Input File Name* parameter must specify the corresponding filename.
- *PRBS_length:* If the *PRBS_length* value N is chosen to be different then zero, a $(2^N - 1)$ PRBS (*Pseudo Random Bit Sequence*) pattern is generated. If set to zero, the bit pattern is generated using the built-in random number generator instead of the PRBS algorithm. The *PRBS_length* parameter has no effect if an input bitfile is used.

- *Calc. bit shift:* If more than one pattern generator is used during the simulation, the bit sequence will be shifted, to minimize a correlation between the bits sequences, the sequences will be shifted. For example, a simulation which continues 3 pattern generators and a sequence contains 30 bits, the second generator will shift 10 and the third will shift 20 bits. If the value is set to *false*, the user can specify an own shift in *Bit shift*.
- *Loop_Mode:* If the input bitfile specified by the *Input File Name* parameter contains less bits than required for the simulation, the *Loop_Mode* parameter lets you choose to extend the bit sequence by appending zero bits, continuously repeating the last bit, or periodically repeating the complete bit pattern. The *loop* parameter has no effect if a random or PRBS bit pattern is used (PRBS patterns are always periodically repeated).
- *Input File Name:* Specifies the filename for the input bitfile. A bitfile is a plain ASCII text file that contains exactly one bit value (0 or 1) per line. The *Input File Name* has no effect if a random or PRBS bit pattern is used.
- Enables or disables the creation of an output bitfile that contains the generated bit sequence. Even if a user defined bit sequence is used via an input bitfile, the generated bit sequence may differ due to internal or user defined modifications (see *Loop_Mode*, *NoPaddingBits*, *Bit_Code*).
- *Output File Name:* Specifies the filename for the output bitfile. The file format is the same as used for the input bitfile. The *outBitName* has no effect if *file_out* is disabled.
- *NoPaddingBits:* If the *cyclic convolution* mode is selected in the *simulation parameters* dialog, each signal block is considered periodically continued and therefore should not exhibit a signal discontinuity when repeated in order to avoid the creation of artificial frequency components. For non-chirping noise free signals, a periodic behavior is achieved by automatically adjusting the pulse edges at the block boundaries without changing the bit sequence or specified edge geometry. In some cases it can be helpful to enforce that each block is bounded by a certain number of padding bits of the same value both at the beginning and the end of the block in order to avoid (or minimize) signal discontinuities when the block is periodically repeated.
  If *NoPaddingBits* is chosen to be different than zero, **the specified number of padding bits is inserted into the bit sequence at the beginning and the end of each signal block**. Note that inserting padding bits will affect the bit sequence and therefore may yield a different bit pattern than specified by the input bitfile or the PRBS algorithm.
  The insertion of padding bits will not be performed if the linear convolution mode is chosen in the simulation parameters dialog, or if bit coding is applied (see *Bit_Code*). In either case, the *NoPaddingBits* value is ignored.
- *Padding_Bit*: Specifies the value of the padding bits (0 or 1). Has no effect if *NoPaddingBits* equals zero, or if no bit padding is performed due to other parameters (see *NoPaddingBits*).

Input / Output

	Number	Type of signal
Input		No input
Output		No output

Component Parameters

Parameter Name	Modulation scheme
Description	Determines the modulation scheme of the encoded bit patterns
Default Value	DQPSK
Parameter Range	DQPSK, Duobinary, DPSK

Parameter Name	Pattern generation
Description	Algorithm for bit pattern generation
Default Value	Create random sequence
Parameter Range	create random sequence, read from a file, single pulse, sequence of 1's, alternating sequence

Parameter Name	Input file
Description	name name of input file of the bit pattern
Default Value	none
Parameter Range	valid file path

Parameter Name	Bitrate
Description	signal bit rate
Default Value	40
Parameter Range	$0 <$ Bitrate
Unit	Gbit/s

Parameter Name	Loop mode
Description	loop mode (for input bit file only)
Default Value	Repeat all bits
Parameter Range	Repeat all bits, repeat last bit, append zeros

Parameter Name	PRBS length
Description	length of the PRBS pattern, $2^x - 1$, $0 =$ built-in random number generator
Default Value	6
Parameter Range	$0 \leq$ PRBS length

Parameter Name	Sequence
Description	Determines which pattern generation scheme should be used
Default Value	PRBS
Parameter Range	PRBS, deBruijn

Parameter Name	Calc bit shift
Description	Determines whether a maximal bit shift between different Pattern generators should be calculated
Default Value	false
Parameter Range	boolean

Parameter Name	NoPaddingBits
Description	number of padding bits (beginning/end of every block, cyclic convolution mode only)
Default Value	0
Parameter Range	$0 \leq$ NoPaddingBits

Parameter Name	Output file
Description	Determines whether the original bit pattern is saved in a ASCII file
Default Value	false
Parameter Range	boolean

Parameter Name	Output file name
Description	output file name of the original bit pattern
Default Value	none
Parameter Range	valid file path

Parameter Name	Output file name(I channel)
Description	output file name of the inverted bit pattern of the inphase channel
Default Value	none
Parameter Range	valid file path

Parameter Name	Output file name (Q channel)
Description	output file name of the bit pattern of the quadrature channel
Default Value	none
Parameter Range	valid file path

Parameter Name	Output file name (diff I channel)
Description	output file name of the bit pattern of the precoded inphase channel
Default Value	none
Parameter Range	valid file path

Parameter Name	Output file name (diff Q channel)
Description	output file name of the bit pattern of the precoded quadrature channel
Default Value	none
Parameter Range	valid file path

References

[1] R.A. Griffin and A.C. Carter, “Optical Differential Quadrature Phase-Shift Key (oDQPSK) for High Capacity Optical Transmission”, Wednesday session, WX6, Atlanta, OFC 2002

[2] W. Kaiser, T. Wuth, M. Wichers and W. Rosenkranz, “Reduced Complexity Optical Duobinary 10-Gb/s Transmitter Setup Resulting in an Increased Transmission Distance”, IEEE Photon. Technol. Lett., vol. 13, pp. 884-886, Aug. 2001

7.11.6 Pattern Modifier

Fundamentals

The pattern modifier component does not represent a physical model. Its purpose is to offer a flexible tool for adjustments of the parametic signal representation, e. g. in setups with unusual simulation settings that cause formally incorrect parametic signals. This allows to account for very special and innovative approaches which are often dealt with in research facilities.

Input / Output

	Number	Type of signal
Input	1	Optical Field / Electrical signal
Output	1	Optical Field / Electrical signal

Component Parameters

Parameter Name	Channel number
Description	Determines the number of channels that are to be modified
Default Value	1
Parameter Range	1 ≤ Channel number

Parameter Name	Ch_x delete
Description	Determines whether channel x is to be deleted from the parametric signal
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Edit pattern
Description	Determines whether the bit pattern of channel x is to be substituted by a different one
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Bit pattern source
Description	Determines the source of the bit pattern
Default Value	File
Parameter Range	File, Other Channel

Parameter Name	Ch_x Filename
Description	Filename that contains the bit pattern
Default Value	none
Parameter Range	valid file path

Parameter Name	Ch_x source channel
Description	Channel number of the bit pattern source
Default Value	1
Parameter Range	not implemented

Parameter Name	Ch_x Edit power
Description	Determines whether the min and max power of channel x are to be modified
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Min power
Description	Power of a space
Default Value	0.0
Parameter Range	not implemented
Unit	W

Parameter Name	Ch_x Avg power
Description	Average power of the signal
Default Value	0.0005
Parameter Range	not implemented
Unit	W

Parameter Name	Ch_x Max power
Description	Power of a mark
Default Value	0.001
Parameter Range	not implemented
Unit	W

Parameter Name	Ch_x Edit delay
Description	Determines whether the delay of channel x is to be modified
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Delay
Description	Additional delay of channel x (may be positive or negative)
Default Value	0
Parameter Range	not implemented
Unit	ps

Parameter Name	Ch_x Edit frequency
Description	Determines whether the center frequency of channel x is to be modified
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Frequency
Description	Center frequency of channel x
Default Value	193.1
Parameter Range	not implemented
Unit	THz

Parameter Name	Ch_x Edit inversion
Description	Determines whether the inversion of channel x is to be modified
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Inversion
Description	Determines whether channel x is to be inverted
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Edit bitrate
Description	Determines whether the bitrate of channel x is to be modified
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Bitrate
Description	Bitrate of channel x
Default Value	40
Parameter Range	not implemented
Unit	Gbit/s

Parameter Name	Ch_x Edit bit code
Description	Determines whether the bit code of channel x is to be modified
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Bit code
Description	Determines the bit code of channel x.
Default Value	none
Parameter Range	none, Duobinary, AMI, Phase switch, Amplitude switch

Parameter Name	Ch_x Edit pulse code
Description	Determines whether the pulse code of channel x is to be modified
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Pulse code
Description	Determines the pulse code of channel x.
Default Value	NRZ
Parameter Range	NRZ, RZ

Parameter Name	Ch_x Edit bit shape
Description	Determines whether the bit shape of channel x is to be modified
Default Value	false
Parameter Range	boolean

Parameter Name	Ch_x Bit shape
Description	Determines the bit shape of channel x.
Default Value	Rectangle
Parameter Range	Rectangle, Squared cosine, Sine, Sech, Gaussian, Triangle, Saw up, Saw down

References

none

7.11.7 Result Importer

Fundamentals

The Result Importer component can be used to import a result from a given text file format input file into the PHOTOSS result space. The result may be either from the data type `double`, `int` or `bool` (Results of the data type `string` can not be imported).

Your text file format input file may also contain a vector of results. In this case, the default behavior of PHOTOSS is to select the data in the *first* line. You can also decide to import the data from the last line of the file, too, by checking the “readFromLastLine” option.

However, if your text file includes string phrases, such as e.g. “\” or “&”, the imported result value will set to “0”, since `string` results are not permitted in PHOTOSS.

The Result Importer can be selected to operate either in the optical or electrical domain.

Input / Output

	Number	Type of signal
Input	1	Optical Field / Electrical signal

Component Parameters

Parameter Name	operatingDomain
Description	Operating domain of the Result Importer
Default Value	optical
Parameter Range	may be either optical or electrical

Parameter Name	readFromLastLine
Description	If checked, the result will be imported from the last line of the input file. If unchecked, it will be imported from the first line.
Default Value	false
Parameter Range	boolean

Parameter Name	resultInputFile
Description	path to the input file containing the result
Default Value	none
Parameter Range	valid input file path

Parameter Name	resultUnit
Description	unit of the user-defined, imported result
Default Value	none
Parameter Range	any valid string expression

Results

Parameter Name	result
Description	user-defined and imported result

References

none

7.11.8 Signal Comparator

Fundamentals

The signal comparator is designed to quantify the difference between two signals. The power difference of the signals is integrated over the full signal duration. Finally the result is normalized to the integrated power of the first signal:

$$e = \frac{\left| \int P_1(t) - P_2(t)\, dt \right|}{\int P_1(t)\, dt} \tag{1}$$

The resulting dimensionless value e represents a measure for the difference of the signals.

Input / Output

	Number	Type of signal
Input	1	Optical Field / Electrical signal
Output	1	Optical Field / Electrical signal

Results

Parameter Name	e
Description	Error / difference of the signals

References

none

7.11.9 Transient Importer

Fundamentals

The Transient Importer component provides the interface to PHOTOSS *Transient*. The Transient Importer replaces the EDFA component and imports the spectral characteristics of the signal channels (and noise bins) at the desired time instances from PHOTOSS *Transient*.

Component Parameters

The component requires that a filename is specified. This file is created by the Bit Sim Exporter component in PHOTOSS Transient. It contains the spectrally resolved power levels calculated in the transient simulation at user specified time instances. The incoming signals are amplified to the (average) power levels defined in the file. All signal frequencies used in PHOTOSS must be contained in the file. However, not all frequencies defined in the file need to be used in PHOTOSS. If analytical noise is selected in the main parameters, it is also possible to import the noise bins from the file. For this purpose the spectral width and the center frequencies of the noise bins of the imported file and the actual simulation must be identical.

Please note that the transient importer only works correctly in separated channels (SC) mode in the WDM case.

This enables the user to calculate the important multi-channel effects (e.g. SRS-tilt and EDFA spectral characteristics) in the time efficient PHOTOSS *Transient* simulation and to compute the PHOTOSS simulation only for a subset of frequencies to estimate the impact of the nonlinear fiber effects.
Because it is possible to define several time instances, which are of interest to the user (e.g. the maximal under- or overshoots), also the Time Instance Index has to be defined, which selects the desired column from the file.

Input / Output

	Number	Type of signal
Input	1	Optical Field
Output	1	Optical Field

Component Parameters

Parameter Name	Read from file
Description	File containing data to be imported
Default Value	none
Parameter Range	valid file path

Parameter Name	Time Instance Index
Description	Column of the desired time instance to be imported
Default Value	1
Parameter Range	not implemented

References

none

7.12 Management

7.12.1 File Loader

Fundamentals

The file loader is a helpful component for simulating complex network structures. The network can be subdivided into several smaller units which can be simulated sequentially. The output signals of each part can be read from the disk into the file loader.

You may also use the file loader during a parameter variation, to load the signal for each variation from a different file. Simply create a global variable as described in the section Parameter Variation on page 71. Assuming x is your global variable write something like *myFile_Parse*{x} in the file name parameter of the file loader.

If the simulation parameters in the loader's file differ from the current simulation parameters the file loader tries to adjust the current simulation parameters and writes a warning message to the output window. If the adjustment fails (e.g. when there is more than one file loader with different simulation parameters) the simulations stops and an error message is written to the output window.

The parameter File Format determines whether the input file is written in either TEXT, PHOTOSS or MATLAB® format by the filesaver. If the parameter Convert noise is checked, the file loader uses the analytical noise density vector of the input file to create numerical noise with the same spectral properties. This numerical noise is then added to the sampled signal.

Input / Output

	Number	Type of signal
Input	1	no Input
Output		Optical Field / Electrical Signal

Component Parameters

Parameter Name	filename
Description	Name of the file
Default Value	none
Parameter Range	valid file path

Parameter Name	File Format
Description	Determines the input file format
Default Value	Text
Parameter Range	Text, PHOTOSS, MATLAB

Parameter Name	adjustSimParams
Description	Determines, whether simulation parameters from the signal file should override current simulation parameters
Default Value	No
Parameter Range	No, Yes

Parameter Name	Convert Noise
Description	Determines, whether the noise representation is converted to numerical
Default Value	false
Parameter Range	boolean

References

none

7.12.2 File Saver

Fundamentals

The file saver is a helpful component for simulating complex network structures. The network can be subdivided into several smaller units which can be simulated sequentially. The output signals of each part can be stored to disk using the file saver.

You may also use the file saver during a parameter variation to store the results of each variation in a different file. Simply create a global variable as described in the section Parameter Variation on page 71. Assuming x is your global variable write something like $myFile\_Parse\{x\}$ in the file name parameter of the file saver.

It is also possible to save the signal only to a separate file for MATLAB® import. (This option requires parameters “Signal file” *and* “filename” to be specified.)

The option *FileFormat* determines the output format of the file. The format used in 3.xx versions if PHOTOSS is called PHOTOSS.
TEXT is a new and simplified format. It is also possible to export into the .mat format, if MATLAB® is chosen. Compression of the file can be enabled by checking the *Compressed* box. This option is also possible in combination with .mat format. If MATLAB® is chosen as format two structures called *Multi_Signal* and *Multi_Signal_Param* are created which are identical to the workspaces created by the PHOTOSS component MATLAB® interface after the parameterized and sampled runs, respectively.

Input / Output

	Number	Type of signal
Input	1	Optical Field / Electrical Signal
Output		no Output

Component Parameters

Parameter Name	filename
Description	Name of the file
Default Value	none
Parameter Range	valid file path

Parameter Name	Save signal to separate file
Description	Creates output file containing the signal only (for MATLAB® import)
Default Value	false
Parameter Range	boolean

Parameter Name	Signal File
Description	Name of file containing the signal only
Default Value	none
Parameter Range	valid file path

Parameter Name	File Format
Description	Determines the output file format
Default Value	Text
Parameter Range	Text, PHOTOSS, MATLAB

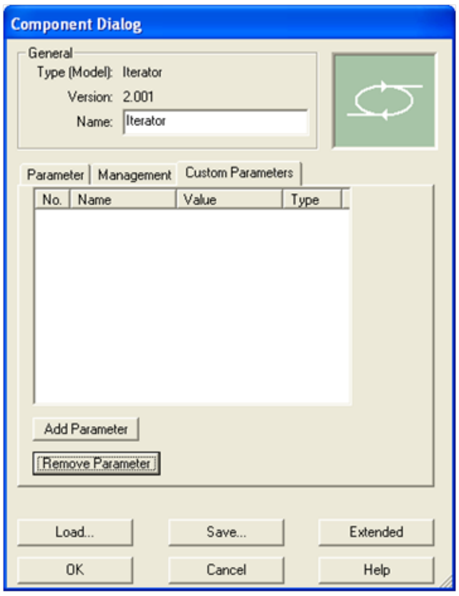
Parameter Name	Compressed
Description	Enables compression
Default Value	false
Parameter Range	boolean

References

none

7.12.3 Iterator

Fundamentals



Component Dialog

General

Type (Model): Iterator

Version: 2.001

Name: Iterator

Figure 7.12.1. Iterator Component Dialog

The iterator works like a network. Its subnetwork can be expanded by pressing the expand- button or by right-clicking the component and choosing “open iterator” from the context menu. The iterator further allows to repeat the signal operation of the included network in vertical or horizontal direction. As an example, a horizontal iteration is useful, if a couple of fiber-spans are to be defined, whereby the component parameters of the fibers are the same in each span (serial networks). A vertical iterator might be useful in designing a transmitter or receiver array with multiple WDM channels (parallel networks).

For example, a horizontal Iterator with the size n will always have one input and output port; If it is used in the vertical mode instead, it will have n input and output ports. The component parameter dialog of the sub-network is opened by right-clicking the iterator. In this dialog, the number of input- and output-ports of the sub-network can be defined.
The number of input- and output-ports of the sub-network and the number of iterations yield the total number of input- and output-ports of the iterator.

In order to encapsulate the function of an iterator, an iterator may have its own set of variables that is only accessible by components stored within. These components may access both Global variables and network variables. Variables of the iterator are accessed using the prefix ’net.’ followed by the name of the variable.

If an iterator is cascaded within another iterator or network, the variables of the parental network/iterator are inherited to the child iterator and can be accessed by using ’net.net.variable_name’, while variables of the child iterator itself may still be accessed via ’net.variablename’. Additionally, the iteration index can be accessed via the keyword *iteratorname_ItrNo*. The keyword *iteratorname_ItrSize* returns the size of the iterator. For example, when setting up a WDM system with 50 GHz channel spacing, a vertical iterator containing a Pulse Generator can be used to generate the signal for each channel. We name the Iterator “WDM_Transmitter”. In this case the center frequency f0 may be defined by $193.1 + WDM\_Transmitter\_ItrNo * 0.05$. This creates one channel at 193.15 THz, one at 193.2 THz, etc.

Input / Output

	Number	Type of signal
Input	variable	Optical Field / Electrical Signal
Output	variable	Optical Field / Electrical Signal

Component Parameters

Parameter Name	Size
Description	Number of iterations
Default Value	1
Parameter Range	$1 \leq$ Size

Parameter Name	Mode
Description	Direction of the iteration
Default Value	Horizontal
Parameter Range	Horizontal, Vertical

References

none

7.12.4 Network

Fundamentals

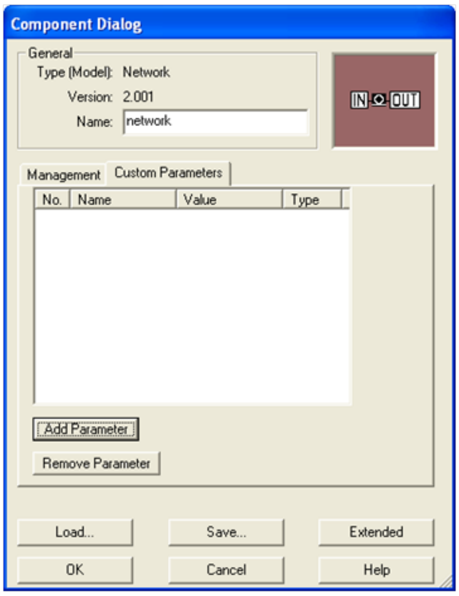


Figure 7.12.2. Network Component Dialog

A network acts as a component, which can include multiple sub components. A network can be opened by pressing the expand-button in the toolbar or by right-clicking the component and choosing “open network” from the context menu. Afterwards a new grid appears where components can be dragged and dropped, as well as linked together.
The network offers the possibility to design networks with a very plain view, even if a couple of components are used.

In order to encapsulate the function of a network, a network may have its own set of variables that is only accessible by components stored within. These components may access both *Global variables* and network variables. Variables of the network are accessed by using the prefix ’net.’ And the name of the variable.
If a network is cascade within another network or iterator, the variables of the parental network/iterator are inherited to the child network and can be accessed by using ’net.net.variable_name’, while variables of the child network itself may still be accessed via ’net.variablename’.

Input / Output

	Number	Type of signal
Input	variable	Optical Field / Electrical Signal
Output	variable	Optical Field / Electrical Signal

Component Parameters

Parameter Name	InPorts / Outports
Description	Number of Inports and Outports
Default Value	1
Parameter Range	$0 \leq$ Ports

References

none

7.12.5 Sample Exporter

Fundamentals

The *Sample Exporter* can be used to write optical samples to a file for processing in third-party software or user scripts. Unlike the *File Saver* (section 7.12.2) this component does not export the whole MultiSignal but only the complex entries from the vectors contained in *comp* (see section 3.7). Each polarization plane is written to an individual file. If *Include polarization effects* is activated in the Built-In Simulation Parameters (section 3.3.2) both parameters *outputFilename x* and *outputFilename y* have to be set to different paths. The output file format matches the input file format of the *Sample Replacer* (section 7.12.6).

The combination of *Sample Exporter* and *Sample Replacer* allows for reusing the sampled signal from one simulation in another while retaining that simulation's parameterized signal. Please note that such manipulations can result in inconsistent MultiSignals which in turn can lead to unexpected behaviour of subsequent components.

This component can not be used in *Separated Channels* mode.

Input / Output

	Number	Type of signal
Input	1	Optical Field
Output		no Output

Component Parameters

Parameter Name	outputFilename x
Description	Filename for samples from polarization plane *x*
Default Value	none
Parameter Range	valid file path

Parameter Name	outputFilename y
Description	Filename for samples from polarization plane *y*
Default Value	none
Parameter Range	valid file path

References

none

7.12.6 Sample Replacer

Fundamentals

The *Sample Replacer* can be used to load samples from a file into the input signal. Unlike the *File Loader* (section 7.12.1) this component does not initialize a complete MultiSignal from file. It requires an input signal in which the entries of the vectors contained in *comp* (see section 3.7) are replaced with values from the given input files. Each polarization plane is read from an individual file. The input file format matches the output file format of the *Sample Exporter* (section 7.12.5).

The combination of *Sample Exporter* and *Sample Replacer* allows for reusing the sampled signal from one simulation in another while retaining that simulation's parameterized signal. Please note that such manipulations can result in inconsistent MultiSignals which in turn can lead to unexpected behaviour of subsequent components.

This component can not be used in *Separated Channels* mode.

Input / Output

	Number	Type of signal
Input	1	Optical Field
Output	1	Optical Field

Component Parameters

Parameter Name	inputFilename x
Description	Filename for samples of polarization plane x
Default Value	none
Parameter Range	valid file path

Parameter Name	inputFilename y
Description	Filename for samples of polarization plane y
Default Value	none
Parameter Range	valid file path

References

none

7.13 Multimode and Multicore

7.13.1 Crosstalk Emulator

Fundamentals

The Crosstalk Emulator can be used to simulate the exchange of amplitude and phase information of electric fields which will occur when two or more signal spatial "channels" are interacting together. One possible filed of application for such a device would be the simulation of modal crosstalk when considering spatial multiplexing in multimode fibers (MMF) or the crosstalk which occurs due to interactions of two or more neighboring channels in a multicore fiber (MCF).

This crosstalk emulation, however, should not be confused with a simulation of inter channel effects like cross phase modulation (XPM) of four wave-mixing (FWM) due to nonlinear effects (please refer to the component *Single Mode Fiber* on page 180 to cover these topics!). Instead, the emulator is designed to create *spatial* crosstalk effects only.

The emulator allows you to define any number of input (and output) channels for your device. As stated above, a channel representation can be a modal channel in spatial multiplexing or even a single fiber core in an MCF. From a functional point of view, the emulator just needs two input parameters: The amount of crosstalk which will occur between the spatial channels (related to the signal amplitude or power) and the phase relation or phase shift between the distinct spatial channels.

Crosstalk relations can be defined either in means of

- a crosstalk factor f (e.g. $s_{1,new} = s_{1,old} \cdot f$),
- an attenuation in dB (e.g. $s_{1,new} = s_{1,old} \cdot \alpha_{dB}$),
- or a crosstalk percentage γ (e.g. $s_{1,new} = s_{1,old} \cdot \gamma_{\%}$).

Internally, however, your specified value is *always* converted to a crosstalk factor.

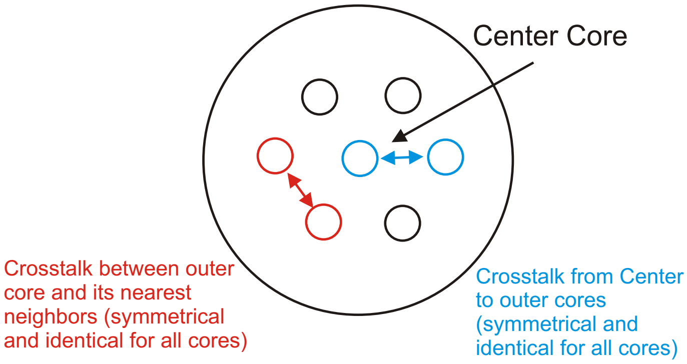

Center Core
Crosstalk between outer core and its nearest neighbors (symmetrical and identical for all cores)
Crosstalk from Center to outer cores (symmetrical and identical for all cores)

Figure 7.13.1. Geometry for the "SevenCoreSimple" mode.

Generally, the emulator offers three different modes of operation:

1. DualCoreSimple. This mode assumes that you have two spatial channels and a symmetric crosstalk from channel 01 to channel 02. The amount of crosstalk can be specified by using the parameter *"crosstalkValueDualCore"*. This mode assumes that the crosstalk value for the x- and y-polarization is identical. If you choose this mode of operation, the loss for each individual channel due to crosstalk effects will be calculated automatically. You can also specify whether the crosstalk value is related to the signal in terms of electrical field amplitudes (unit $\sqrt{\text{W}}$) or power (unit W) by switching the parameter "*unitRelationOfCrosstalk*".

2. SevenCoreSimple. This mode assumes that one center channel ("fiber core") and six outer channels ("outer cores") are present in the spatial geometry. The geometry for this setup is depicted in figure 7.13.1. The crosstalk between the center channel and the six remaining channels is symmetric. It is specified by the parameter *"crosstalkValueCenterToOuterCores"*. The crosstalk between one of the outer channels and its two nearest neighboring channels is defined by the parameter *"crosstalkValueNeighboringOuterCores"*. It is assumed that only the two nearest neighbors of the outer cores influence each other. This mode assumes that the crosstalk values for the x- and y-polarization are identical. If you choose this mode of operation, the loss for each individual channel due to crosstalk effects will be calculated automatically. You can also specify whether the crosstalk values are related to the signal in terms of electrical field amplitudes (unit $\sqrt{\mathrm{W}}$) or power (unit W) by switching the parameter "*unitRelationOfCrosstalk*".

3. CrosstalkMatrix. This is the more general working mode for the crosstalk emulator. You have to specify a "crosstalk matrix input file" in order to define the crosstalk values from each individual channel to each of the remaining other channels. Any number of channels may be defined for this working mode. Asymmetrical crosstalk (e.g. crosstalk from channel 01 to channel 02 is greater than crosstalk from channel 02 to channel 01) is also possible. You can also define distinct crosstalk values for each polarization. You can also specify whether the crosstalk values in your matrix are related to the signal in terms of electrical field amplitudes (unit $\sqrt{\mathrm{W}}$) or power (unit W) by switching the parameter "*unitRelationOfCrosstalk*".

Crosstalk Matrix

All crosstalk values for the "CrosstalkMatrix" mode can be fed into the emulator by specifying the so-called "crosstalk matrix input file". In this file, you can state the amount of crosstalk that occurs between each individual set of channels.

You can create an example file by activating the parameter "*createExampleFile*" and running the simulation to obtain a "blueprint" matrix for your purpose. You can then modify the output matrix file (don't forget to uncheck the parameter or your new file will be overwritten!) or copy it.

Please keep in mind that you can specify the crosstalk relation between the two polarization axes individually. In reality, most models of crosstalk assume that the crosstalk between polarization axes is symmetrical, but you can nonetheless define any asymmetrical relation you wish.

The crosstalk matrix itself can be read as follows. If - for example - we have the following matrix (for reasons of simplicity let us assume that the values listed in the matrix are *crosstalk factors*):

$$\begin{matrix} 0.35 & 0.43 & 0.21 & 0 \\ 0.1 & 0.17 & 0.21 & 0.88 \\ 0.45 & 0.1 & 0.37 & 0.1 \\ 0.1 & 0.3 & 0.21 & 0.02 \end{matrix}$$

This matrix represents the interaction of 4 channels. The jth column of the matrix describes the amount of crosstalk that the channel of the ith row receives from the jth channel.

In the matrix stated above, channel 1 would retain a factor of 0.35 of its original power (or amplitude) level and it would receive a factor of 0.43 of channel 2, a factor of 0.21 of channel 3 and no power from channel 4. You can also see that the remaining factor 0.65 of the power of channel 1 is distributed unevenly to channel 2 (factor 0.1), channel 3 (factor 0.45) and channel 4 (factor 0.1).

You can easily notice that the matrix above states a system where the *total* energy of all channels remains constant - there is no *overall* gain or loss of the crosstalk element. In real physical components, however, there may be a net loss (or even gain) for the whole component. The net gain or loss is always specified by the entries $i = j$ in the matrix; they describe the "crosstalk" of each channel to itself. If we consider crosstalk factors (as stated above), a "zero" entry would mean that the original channel information is completely lost (although some of it may be distributed to *other* channels nonetheless). A "one" entry would mean that the channel is completely preserved (although it may also receive some additional crosstalk from other channels).

It is also possible to let PHOTOSS automatically calculate the index entries $i = j$ in the matrix from all entries with $i \neq j$ by checking the parameter "*calcChannelAttenuationAutomatically*" to achieve a sum of 1.0 for all elements for each column and to represent a crosstalk element with no net loss or gain. If you activate the parameter "*checkForTotalPower*", PHOTOSS will issue a warning if the sum of all matrix elements of one column is > 1 and the component would produce an overall net gain.

Phaseshift Matrix

By activating "*usePhaseShiftMatrix*" you can also assign a special phaseshift factor for the crosstalk of each channel. E.g. for a system with three channels, the resulting field amplitude $s_{1,new}$ for channel 1 would be:

$$s_{1,new} = s_{1,old} \cdot f_{11} \cdot \exp(i \cdot p_{11}) + s_{2,old} \cdot f_{12} \cdot \exp(i \cdot p_{12}) + s_{3,old} \cdot f_{13} \cdot \exp(i \cdot p_{13}), \tag{1}$$

where f_{ij} denotes the crosstalk factor from channel j to channel i and p_{ij} is the phase shift between channel j and i. If the parameter "*useRandomPhaseShifts*" is checked, PHOTOSS will assign an evenly distributed random variable for each p_{ij} in the range of $[0, 2\pi]$ instead.

Type of Crosstalk You can specify the "type" of crosstalk by switching the parameter "*unitRelationOfCrosstalk*". When opting for the choice $\sqrt{\text{W}}$, the crosstalk will be emulated using interaction of the complex field (including both amplitude and phase information). This kind of crosstalk is *coherent* - e.g. the different crosstalk parts may interfere with each other. Choosing W instead, results in a crosstalk which is only attributed to an exchange of signal power - this crosstalk type is *not* coherent and no interference between the channels will appear.

Input / Output

	Number	Type of signal
Input	variable	Optical field
Output	identical to number of input channels	Optical field

Component Parameters

Parameter Name	emulatorMode
Description	Selects the working mode of the emulator.
Default Value	CrosstalkMatrix
Parameter Range	CrosstalkMatrix, DualCoreSimple, SevenCoreSimple

Parameter Name	calcChannelAttenuationAutomatically
Description	Automatically derive matrix indices f_{ij} to ensure that the emulator conserves its total energy. Appears only in “CrosstalkMatrix” mode.
Default Value	false
Parameter Range	boolean

Parameter Name	checkForTotalPower
Description	Issues a warning if current channel crosstalk creates a net gain in the component. Appears only in “CrosstalkMatrix” mode.
Default Value	true
Parameter Range	boolean

Parameter Name	createCrosstalkExampleFile
Description	Creates an example file holding an empty crosstalk matrix for you to modify
Default Value	false
Parameter Range	boolean

Parameter Name	createPhaseshiftExampleFile
Description	Creates an example file holding an empty phaseshift matrix for you to modify
Default Value	false
Parameter Range	boolean

Parameter Name	crosstalkMatrixOutputFile
Description	File to which the (example) output matrix will be saved
Default Value	none
Parameter Range	valid file name

Parameter Name	phaseshiftMatrixOutputFile
Description	File to which the (example) output phaseshift matrix will be saved
Default Value	none
Parameter Range	valid file name

Parameter Name	crosstalkMatrixInputFile
Description	File from which the crosstalk matrix will be loaded.
Default Value	none
Parameter Range	valid file name

Parameter Name	phaseshiftMatrixInputFile
Description	File from which the phaseshift matrix will be loaded.
Default Value	none
Parameter Range	valid file name

Parameter Name	crosstalkFormat
Description	Specifies the format of crosstalk.
Default Value	Attenuation in dB
Parameter Range	Attenuation in dB, Crosstalk in Percent, Crosstalk Factor

Parameter Name	numberOfChannels
Description	Number of Input and Output Channels.
Default Value	2
Parameter Range	from 2 to 2000

Parameter Name	unitRelationOfCrosstalk
Description	Do the crosstalk values relate to signal amplitudes or signal power?
Default Value	[W]
Parameter Range	[W], [$\sqrt{W}$]
Parameter Name	usePhaseShiftMatrix
Description	If checked, the entries in the phase shift matrix will be used
Default Value	false
Parameter Range	boolean

Parameter Name	useRandomPhaseShifts
Description	If checked, the each channel will have a random phase shift in the interval $[0, 2 \cdot \pi]$ with uniform distribution
Default Value	false
Parameter Range	boolean

Parameter Name	crosstalkValueDualCore
Description	Amount of crosstalk between two cores. Only appears in "DualCoreSimple" mode.
Default Value	0
Parameter Range	not limited

Parameter Name	crosstalkValueCenterToOuterCores
Description	Amount of crosstalk between center and each outer core. Only appears in "SevenCoreSimple" mode.
Default Value	0
Parameter Range	not limited

Parameter Name	crosstalkValueNeighboringOuterCores
Description	Amount of crosstalk between neighboring outer cores. Only appears in "SevenCoreSimple" mode.
Default Value	0
Parameter Range	not limited

References

[1] Matthias Westhäuser et al, "Mitigation of combined PMD- and crosstalk-induced signal distortions in spatially-multiplexed multi-core fiber networks", Signal Processing in Photonics Communications (SPPCom), Colorado Springs, USA, June 2012.

[2] Matthias Westhäuser et al, "Reduction of crosstalk-induced OSNR penalties in high bit rate optical spatially multiplexed systems", Signal Processing in Photonics Communications (SPPCom), Colorado Springs, USA, June 2012.

[3] Matthias Westhäuser et al, "Quantifizierung und Reduktiuon von OSNR-Einbußen in hochbitratigen Mehrkernfasersystemen", ITG Fachtagung Photonische Netze, Leipzig, Germany, May 2012.

7.14 Testing Equipment

7.14.1 Optimization Tester

Fundamentals

The Optimization Tester (OT) is a component which has been created specifically for testing the built-in optimization routines of PHOTOSS. This component does not serve any physical purpose but it can be used to obtain a feel for the parameters and abilities of the built-in optimization routines. The component emulates a number of different 3D surfaces with a varying number of local minima and maxima. The goal for each optimization routine is to find the global minimum of the chosen 3D surface by varying the coordinates X and Z in the parameter space. So, generally this procedure corresponds to a two-dimensional parameter space optimization.

The global minimum of each test surface is known to the PHOTOSS environment and will be displayed in the console output once the simulation is run (*"Theoretical function minimum"*). You can also observe the newly obtained coordinates for the alleged global minimum which has been derived by the optimizer as well as its value. The closer this value is to the theoretical minimum, the better the result of the optimization routine.

It also very useful to take a look at the plot of the absolute residual error between the alleged minimum and the true minimum of the function. You can do this by right-clicking on the component and choosing the visualization "Optimization Convergence". The visualizer will show you how the residual error changes due to an increasing number of iterations of the optimizer.

The residual error is defined as:

$$\epsilon_{res} = \mathrm{abs}(min_{true} - min_{alleged}) \tag{1}$$

Please also keep in mind that the choice of initial starting point, etc. of the optimizers and their parameters may greatly influence their capabilities. Chosing a "wrong" set of parameters for a specific kind of problem may naturally result in very inaccurate results. Use the OT to get a feel for the individual parameters and their influences and you will be able to extrapolate how the optimizers might be used in your kind of optimization problem in PHOTOSS.

Components which also use the optimization routines in PHOTOSS are (e.g.) the equalizer components (Adaptive, Adaptive One Tap, MIMO, etc.). You can select the appropriate optimization routine and its parameters in the specific components.

Optimization Algorithms

The OT supports the testing of various optimization algorithms which are used in other PHOTOSS components. Here, you can find a short description of each implemented algorithm:

- **Conjugate-Gradient**: This numerical optimization algorithm [1] based on the possibility to compute the gradient of a function at the position x and not only the value of the function. In each iteration the algorithm will span a set of conjugate gradients in the search space and will compute a new minimum with the line minimizing method. It is potentially faster than the 'Powell' method. Additionally the 'Polak-Ribiere' variant outperforms the 'Fletcher-Reeves' method.
- **Genetic**: This stochastic optimization algorithm imitates genetic processes. The main paradigm of this method is called 'Survival of the fittest'. An adjustable amount of individuals represents sets of filter coefficients. At the start of each iteration the residual error criterion will be computed. It is possible to keep the best individual in every case. In the next step the individuals will be mutated or passed trough a crossover. In a mutation step the filter coefficients will be chosen randomly with a

given probability and also the values of the coefficients will be determined randomly in adjustable upper and lower boundaries. In a crossover step a randomly determined amount of coefficients of two individuals will be interchanged. In the end step of an iteration an adjustable amount of individuals will be chosen and battle each other in a tourney. The individual with the better fitness wins.

- **Particle Swarm Optimizer**: The particle swarm optimizer (PSO) is described in more detail in [2, 3]. The general idea is to use a number of m “particles” to explore the n-dimensional parameter space for the coefficient values. In this case, every particle represents an individual choice of filter coefficients. In contrast to the simple random walk routine, the PSO tries to avoid local optima by taking into account the “knowledge” of the best performance of each particle and a “global knowledge” which monitors the best overall achieved performance of all particles.

  The larger the number of particles for the PSO, the greater the chance to avoid local optima. However, increasing the number of particles by a factor of p also increases the total computational time needed for a fixed number of iterations by a factor of p. To speed up the convergence of the algorithm, minimum and maximum boundaries for the coefficient values must be specified by the user. Generally, the default values should be sufficient to solve most distortion problems but these values might have to be increased or decreased depending on your specific optimization problem.

  The parameter *positionStepwidth* is crucial for the convergence of the PSO optimization routine, it represents the particle’s “speed” in the n-dimensional optimization procedure and it describes, how large changes in the value(s) of a coefficient in each iteration step may be. A step width which is too small results in a very fine sampling of the result space but may require a much larger number of iterations to converge to a globally optimal solution. On the other hand, a very large step width may also increase the required number of iterations to reach a stable solution since the sampling of the result space is too coarse. Usually, a position step size between 0.1 and 0.01 is a good choice.

- **Powell**: The ’Powell’ algorithm [1] is a numerical optimization technique, that is based on the concept of conjugate directions. In each iteration the algorithm will span a set of conjugate directions in the search space and will compute a new minimum with the line minimizing method. ’Powell’ is an improved variant of the concept of conjugate directions that counteract the problem of linearly dependency of the generated directions at an increasing amount of iterations.

- **Quasi-Newton**: This numerical algorithm [1] is similar to ’Conjugate-Gradient’ and ’Powell’. It also based on line minimizing to derive an exact minimum of the quadratic form of a Taylor series of a starting point.

  $$f(x) \approx c - bx + \frac{1}{2}x^T Ax \tag{2}$$

  With $c = f(a)$, $b = -\nabla f$ and $A_{ij} = \frac{\delta^2 f}{\delta x_i \delta x_j}$.

  The additional idea is to compute iteratively a good approximation of the inverse Hessian matrix A^{-1}. But the Newton direction ist only a descent direction, if the Hessian matrix A will be positive definite.

- **Random Walk**: The Random walk algorithm executes an n-dimensional search in the complex parameter space where n is equivalent to the number of filter coefficients times two, since the coefficients contain both a real and an imaginary part which must both be determined. The search is carried out randomly in each dimension of the parameter space. After a new set of coefficients is determined, the optimizer checks whether the new set achieves a better solution than all previously tested sets. If the new set shows improvement, it is accepted and the search continues until all iterations have been carried out. Generally, a random walk algorithm is very fast, but has the disadvantage of converging to local optima.

- **Threshold-Accepting**: The 'Treshold-Accepting' [1] is a stochastical optimization algorithm based on 'Simulated Annealing'. From a given temperature $T > 0$ the algorithm starts with a slight change of the start set of filter coefficients. After the change the difference of quality is derived from the old and the new set. If the difference is greater than $-T$ the new set will be chosen. That is the main fact, that differs from 'Simulated Annealing'. If the difference is smaller than $-T$ the old set persists and a new iteration will begin. If there are too many iterations at the given temperature, T will be decreased and the algorithm goes on with the new temperature. If $T = 0$ the optimization ends and the current set of filter coefficients will be returned.

- **Trust-Region**: Another numerical optimization algorithm is 'Trust-Region' [1]. Based on the starting point the algorithm generates a new point with the aid of a model function. In a definite region around the old point, it will be trusted, that the model function is a good approximation of the real function. If the region is found, the function can be minimized in this region and this will solve a part of the whole problem. If the new point is not acceptable, the size of the region will be decreased and the direction of the next step will be changed.

Test Surfaces

You may choose any of the test surfaces depicted below. The "easiest" surface for an optimizer should be represented by the surface named "Deep Dip 2D" (see figure 7.14.1). It can be calculated by using:

$$Z = \sqrt{\log(X^2 + Y^2 + 1)}; \tag{3}$$

The surface "Easy Surface 2D" represents a slightly more complex challenge, since it also consists of areas which do not include a gradient in the combined parameter and result space (see figure 7.14.2). Its results can be calculated by using:

$$Z = real(Y \cdot exp(-X^{2.1} - Y^{2.1})); \tag{4}$$

Finally, the surface "Medium Surface 2D" includes a complex topology which consists of several local minima and maxima and areas which do not include a gradient in the results space (see figure 7.14.3). Its results are calculated by using:

$$Z = real(Y \cdot exp(-X^{1.55} - Y^{1.8})); \tag{5}$$

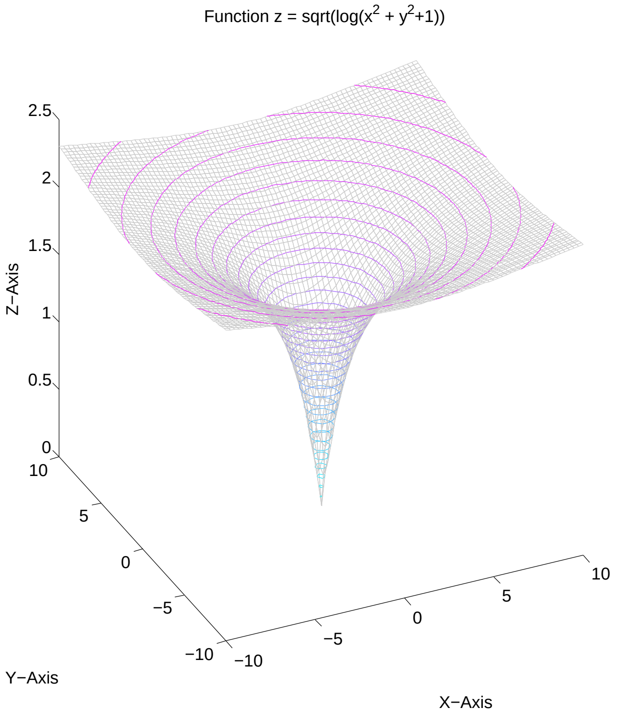


Figure 7.14.1. Test surface “Deep Dip”

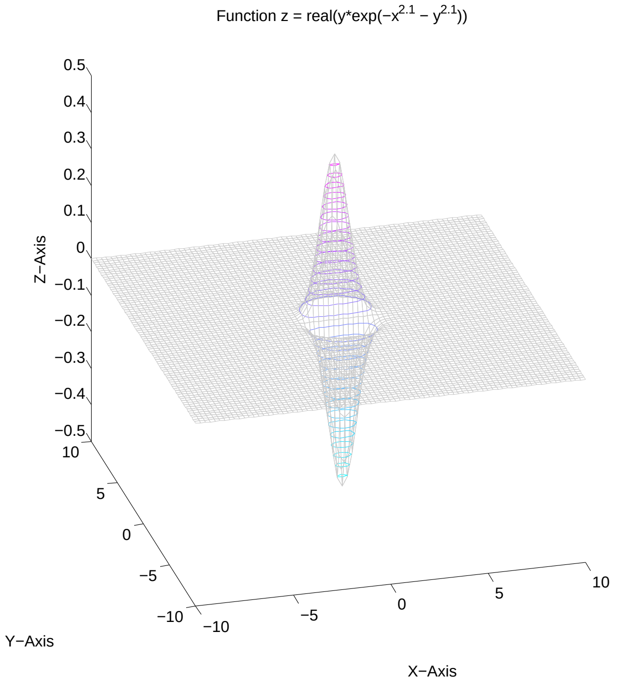


Figure 7.14.2. Test surface “Easy Surface 2D”

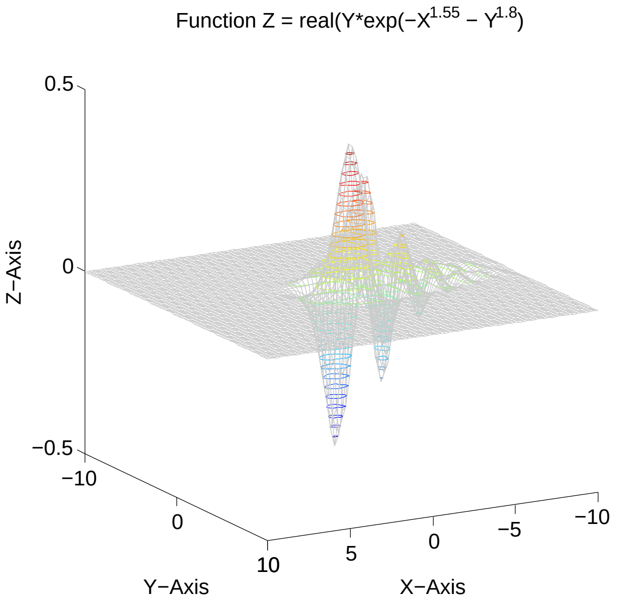


Figure 7.14.3. Test surface “Medium Surface 2D”

Input / Output

none

Component Parameters

Parameter Name	testSurface
Description	chosen test surface
Default Value	Deep Dip 2D

Additional parameters are dependent on the chosen type of optimization routine and are explained by tooltips.

References

[1] William H. Press et al, “Numerical Recipes”, Third Edition, Cambridge University Press, 2007.

[2] James Kennedy and Russell Eberhart, “Particle Swarm Optimization”, Proceedings of the IEEE International Conference on Neural Networks, IV, (Piscataway, NJ: IEEE Service Center, 1995), pp. 1942-1948, 1995.

[3] Ying Zhou et al, “Particle swarm optimization-based approach for optical finite impulse response filter design”, Applied Optics, Vol. 42, No. 8, pp. 1503-1507, 2003.

7.15 User Defined Models

7.15.1 10 Gb/s CS-RZ Modulator

Fundamentals

This user-defined model represents a carrier-suppressed RZ modulator. The signal is generated by a Mach-Zehnder modulator, which modulates a cw laser. The MZ modulator is driven by an electrical pulse generator and creates an amplitude modulated signal. In order to produce a carrier-suppressed signal, the signals phase has to be modulated as well. Therefore, it is altered by a phase modulator that is driven by a pulse generator that produces an alternating sequence. This CS-RZ signal is filtered by a bandpass filter to restrict the signal's bandwidth.

Component settings

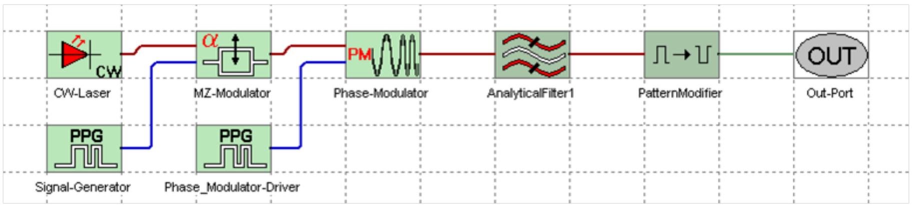


Figure 7.15.1. Simulation Setup

The component 'Signal-Generator' produces an ordinary electrical RZ signal with a minimum voltage of 1.2V and a maximum voltage of 4.6V. The shape of the pulses is a squared cosine with a duty cycle of 0.5.
The 'CW-Laser' generates a 1 mW signal with at a carrier frequency of 193.1 THz.

The 'MZ-Modulator' operates with a conductor loss of 0.55 db/cm $\sqrt{GHz}$, an optical refractive index of 2.17 an electrical index of 2.6 and an insertion loss of 5 dB. Its switching voltage is set to 3.4V and the bias voltage to 1.2 V. The sensitivities of the first and second path are set to 89 1/(Vm) and 12.711 1/(Vm), respectively.
The 'Phase_Modulator-Driver' generates an amplitude modulated electrical NRZ signal with a minimum value of 0 mV and a maximum value of 2mV. The pulses are squared cosine shaped with an alpha roll-off factor of 0.4.
'AnalyticalFilter1' is a 2nd-order Gaussian filter with a bandwidth of 50 GHz.

Characteristics

The output spectrum of the model is depicted in figure 7.15.2.

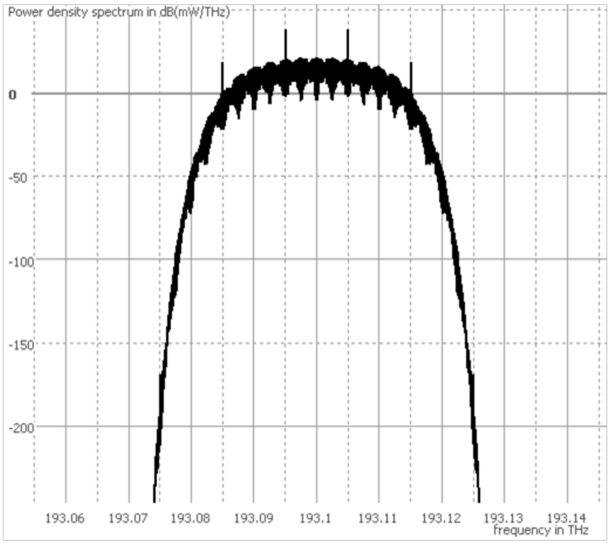


Figure 7.15.2. Spectrum of CSRZ Model

Input / Output

	Number	Type of signal
Input		no Input
Output	1	Optical Field

Component Parameters

Parameter Name	f0
Description	Center frequency of the signal
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	BitRate
Description	Bitrate
Default Value	10
Parameter Range	0 < BitRate
Unit	GBit/s

Parameter Name	avgPower
Description	Average signal power
Default Value	0.001
Parameter Range	not implemented
Unit	W

Parameter Name	PRBS_length
Description	Length of the PRBS sequence ($2^x - 1$)
Default Value	6
Parameter Range	must be a power of 2

7.15.2 10 Gb/s Duobinary Modulator

Fundamentals

This user-defined model represents a duobinary modulator. The signal is generated by a Mach-Zehnder modulator, which modulates a cw laser. The MZ modulator is driven by an electrical pulse generator and creates an amplitude and phase modulated signal. In order to produce a duobinary signal, the electrical signal is low pass filtered with a FWHM bandwidth of 0.6 of the bit rate (here: 6 GHz)

Component settings

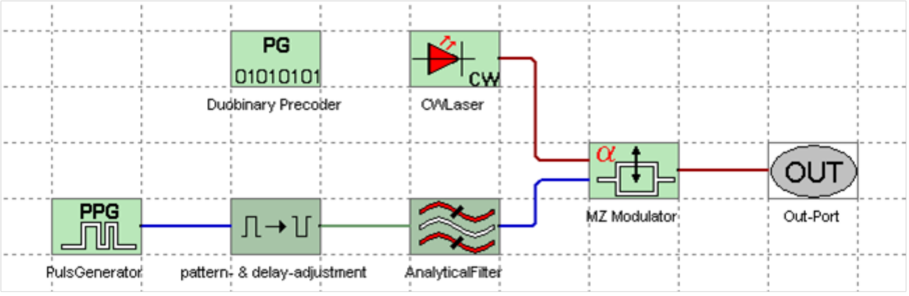


Figure 7.15.3. Simulation Setup

In order to generate a duobinary signal, the bit pattern has to be precoded, following a certain logic. The algorithm is depicted in Figure 7.15.4, where $d(k)$ represents the original bit pattern and $b(k)$ the precoded one.

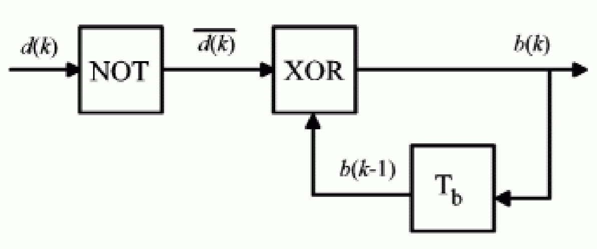


Figure 7.15.4. Duobinary precoder logic

In the pulse generator the encoded bit pattern is read from a file. The pattern- & delay-adjustment component adjusts the parameterized signal to the original bit sequence. At the receiver side a common OOK Receiver may be used.

Characteristics

The output spectrum of the model is depicted in figure 7.15.5.

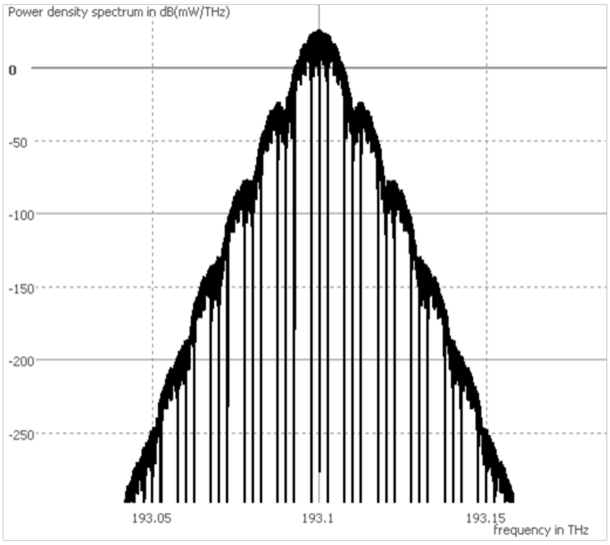


Figure 7.15.5. Spectrum of Duobinary Model

Input / Output

	Number	Type of signal
Input		no Input
Output	1	Optical Field

Component Parameters

Parameter Name	f0
Description	Center frequency of the signal
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	BR
Description	Bitrate
Default Value	10
Parameter Range	0 < BitRate
Unit	GBit/s

Parameter Name	Max_Power
Description	Maximum signal power
Default Value	0.001
Parameter Range	not implemented
Unit	W

Parameter Name	PRBS_length
Description	Length of the PRBS sequence ($2^x - 1$)
Default Value	6
Parameter Range	must be a power of 2

References

[1] W. Kaiser, T. Wuth, M. Wichers and W. Rosenkranz, “Reduced Complexity Optical Duobinary 10-Gb/s Transmitter Setup Resulting in an Increased Transmission Distance”, IEEE Photon. Technol. Lett., vol. 13, pp. 884-886, Aug. 2001

7.15.3 107 Gb/s DBPSK Transmitter (NRZ)

Fundamentals

This user-defined model represents a DBPSK transmitter. The signal is generated by a Mach-Zehnder modulator, which modulates a cw laser. The MZ modulator is driven by an electrical pulse generator and creates a phase modulated signal.

Component settings

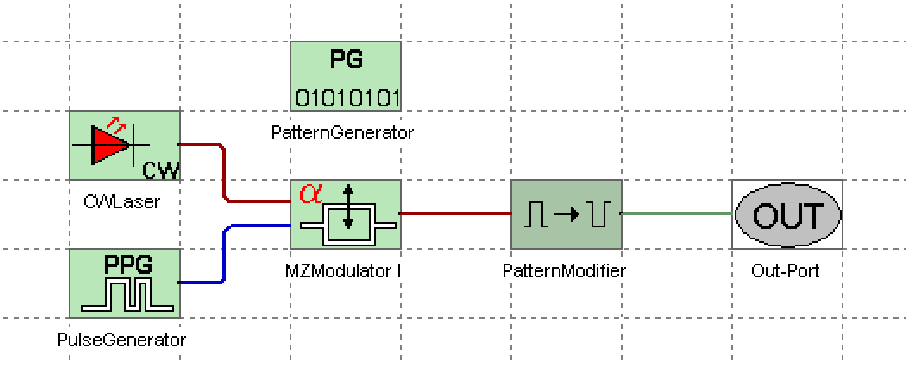


Figure 7.15.6. Simulation Setup

The pattern generator differentially encodes the data stream by an XOR operation with the previous bit.

Characteristics

The output spectrum of the model is depicted in figure 7.15.7.

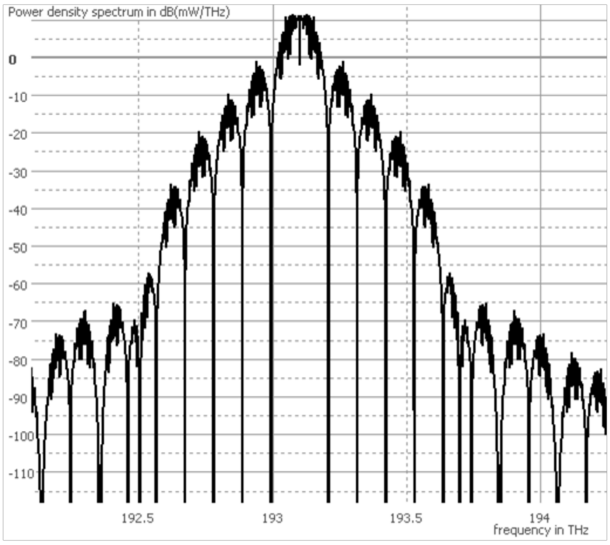


Figure 7.15.7. Spectrum of DBPSK (NRZ) Model

Input / Output

	Number	Type of signal
Input		no Input
Output	1	Optical Field

Component Parameters

Parameter Name	f0
Description	Center frequency of the signal
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	BR
Description	Bitrate
Default Value	10
Parameter Range	0 < BitRate
Unit	GBit/s

Parameter Name	Max_Power
Description	Maximum signal power
Default Value	0.001
Parameter Range	not implemented
Unit	W

Parameter Name	PRBS_length
Description	Length of the PRBS sequence $(2^x - 1)$
Default Value	6
Parameter Range	must be a power of 2

References

[1] A. H. Gnauck, P. J. Winzer, “Optical phase-shift-keyed transmission”, IEEE J. Lightw. Technol, vol. 23, no. 1, pp. 115-130, Jan. 2005.

7.15.4 107 Gb/s DBPSK Transmitter (RZ)

Fundamentals

This user-defined model represents a DBPSK transmitter. The signal is generated by a Mach-Zehnder modulator, which modulates a cw laser. The MZ modulator is driven by an electrical pulse generator and creates a phase modulated signal. In contrast to the NRZ transmitter shown before this time an additional pulse carver is used to generate the RZ signal.

Component settings

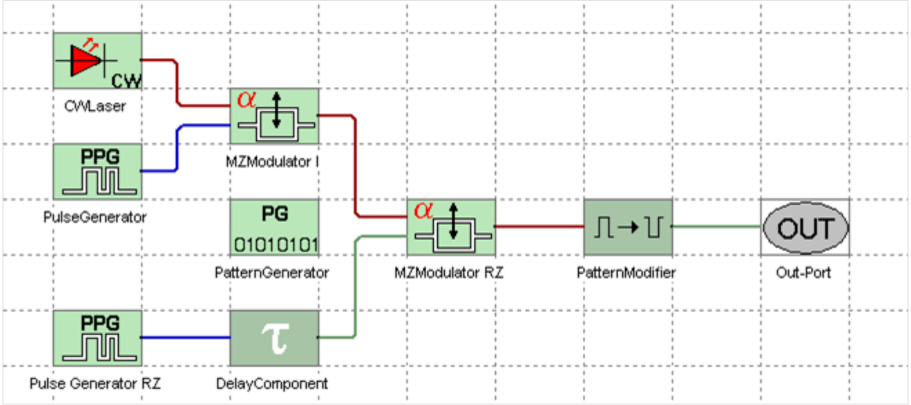


Figure 7.15.8. Simulation Setup

The pattern generator differentially encodes the data stream by an XOR operation with the previous bit.

Characteristics

The output spectrum of the model is depicted in figure 7.15.9.

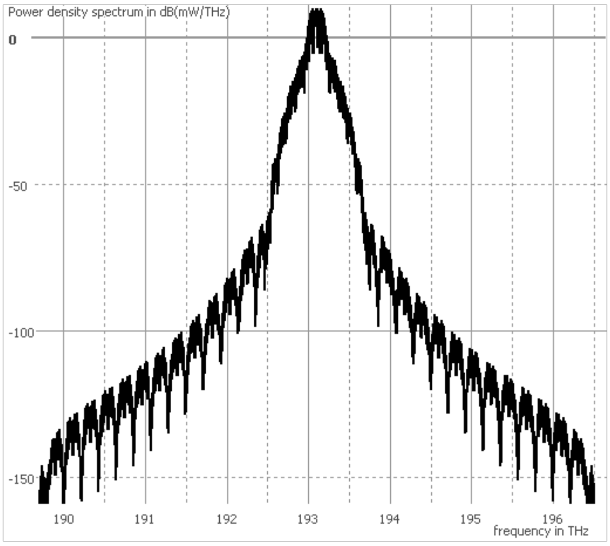


Figure 7.15.9. Spectrum of DBPSK (RZ) Model

Input / Output

	Number	Type of signal
Input		no Input
Output	1	Optical Field

Component Parameters

Parameter Name	f0
Description	Center frequency of the signal
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	BR
Description	Bitrate
Default Value	10
Parameter Range	0 < BitRate
Unit	GBit/s

Parameter Name	Max_Power
Description	Maximum signal power
Default Value	0.001
Parameter Range	not implemented
Unit	W

Parameter Name	PRBS_length
Description	Length of the PRBS sequence ($2^x - 1$)
Default Value	6
Parameter Range	must be a power of 2

References

[1] A. H. Gnauck, P. J. Winzer, “Optical phase-shift-keyed transmission”, IEEE J. Lightw. Technol, vol. 23, no. 1, pp. 115-130, Jan. 2005.

7.15.5 107 Gb/s DBPSK Receiver

Fundamentals

This user-defined model represents a DBPSK receiver. The receiver uses a typical MZ structure with an FSR of the bit rate. After the photo diodes the received signals are subtracted. The pattern modifiers (delete superfluous channel 1 and 2) are used to edit the bit pattern and to set the received power levels correctly.

Component settings

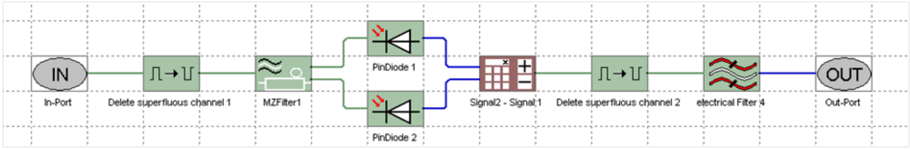


Figure 7.15.10. Receiver Setup

Input / Output

	Number	Type of signal
Input	1	Optical Signal
Output	1	Electrical Field

Component Parameters

Parameter Name	f0
Description	Center frequency of the signal
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	BR
Description	Bitrate
Default Value	10
Parameter Range	0 < BitRate
Unit	GBit/s

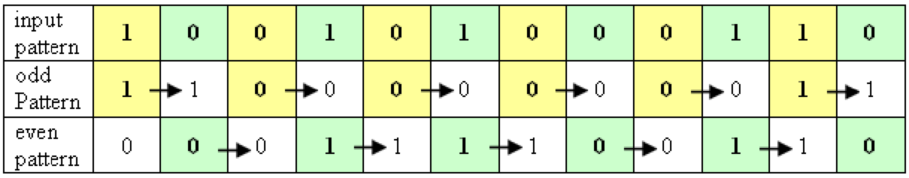
References

[1] A. H. Gnauck, P. J. Winzer, “Optical phase-shift-keyed transmission”, IEEE J. Lightw. Technol, vol. 23, no. 1, pp. 115-130, Jan. 2005.

7.15.6 10 Gb/s DQPSK Modulator

Fundamentals

This model represents a differential QPSK modulator. In order to generate the two differentially encoded signals, which are used by the phase modulator, the input bit sequence has to be manipulated in a certain way. First, the bit sequence is divided into two separate bit streams. One bit stream contains the odd and the other the even bits of the original pattern. These two bit streams have half the bit rate of the input sequence and a delay of half a bit period between each other. To simulate this behavior in PHOTOSS, both patterns are expanded to the size of the input pattern by doubling every bit of the odd and even pattern. A schematic drawing of this operation is depicted in figure 7.15.11.



input pattern	1	0	0	1	0	1	0	0	0	1	1	0
odd Pattern	1	→ 1	0	→ 0	0	→ 0	0	→ 0	0	→ 0	1	→ 1
even pattern	0	0	→ 0	1	→ 1	1	→ 1	0	→ 0	1	→ 1	0

Figure 7.15.11. Bit Pattern

These bit streams are then fed into the precoder, where the differential encoding is done. The streams are encoded according to the following algorithm [1],

$$\begin{aligned} I_k &= \overline{(Q_{k-1} \oplus I_{k-1})(v_k \oplus I_{k-1})} + (Q_{k-1} \oplus I_{k-1})\,\overline{(u_k \oplus I_{k-1})} \\ Q_k &= \overline{(Q_{k-1} \oplus I_{k-1})(u_k \oplus I_{k-1})} + (Q_{k-1} \oplus I_{k-1})\ (v_k \oplus I_{k-1}) \end{aligned} \qquad (1)$$

where I_k and Q_k represent the encoded bits, I_{k-1} and Q_{k-1} the predecessors of I_k and Q_k and v_k and u_k the two input bits.
These encoded bit patterns are then fed into the phase modulator, which produces a signal according the constellation diagram depicted in figure 7.15.12.

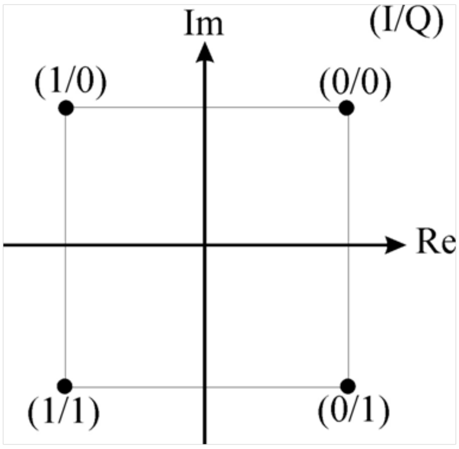


Figure 7.15.12. DQPSK Constellation Diagram

The schematic drawing [1] in figure 7.15.13 illustrates the working principle of this DQPSK modulator.

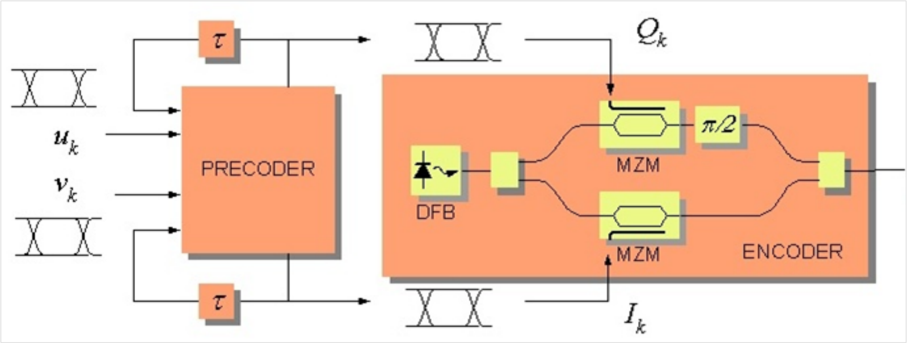


Figure 7.15.13. DQPSK Modulator Working Principle

Component settings

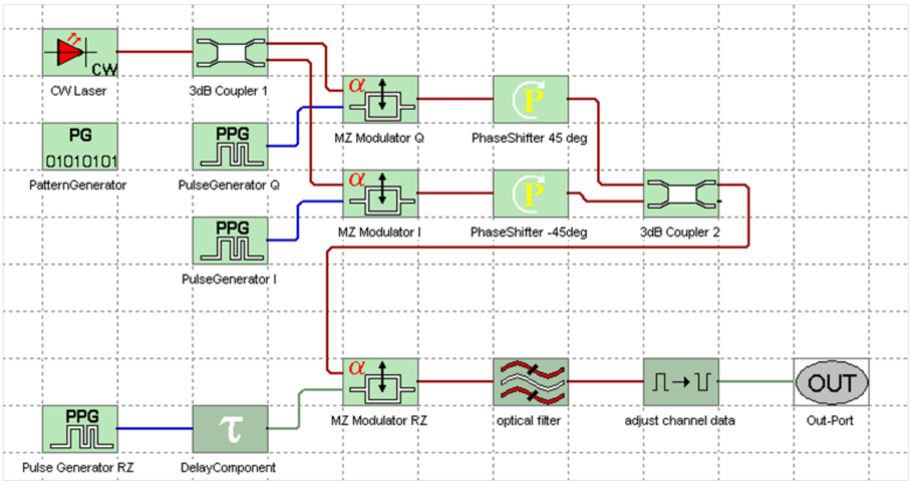


Figure 7.15.14. DQPSK Modulator Setup

The 'DQPSK pattern generator' generates a deBruijn sequence. The uncoded bit sequences are stored to 'Bits_I.txt' and 'Bits_Q.txt'. They are necessary for the decoder, since the bit pattern of the channel has to be adjusted to the decoded bit streams.
The encoded bit patterns are stored in 'Bits_diff_I.txt' and 'Bits_diff_Q.txt'. These two files are then used as input files for the two following pulse generators 'I channel generation' and 'Q channel generation'.

Input / Output

	Number	Type of signal
Input		no input
Output	1	Optical Signal

Component Parameters

Parameter Name	f0
Description	Center frequency of the signal
Default Value	193.1
Parameter Range	within the PHOTOSS simulation bandwidth
Unit	THz

Parameter Name	BR
Description	Bitrate (2x Symbolrate)
Default Value	10
Parameter Range	0 < BitRate
Unit	GBit/s

Parameter Name	avgPower
Description	Average signal power
Default Value	0.001
Parameter Range	not implemented
Unit	W

References

[1] R. Griffin, “Optical Differential Quadrature Phase-Shift Key (oDQPSK) for High Capacity Optical Transmission”, OFC Proceedings, 2002

7.15.7 10 Gb/s DQPSK Demodulator

Fundamentals

The DQPSK demodulator can be used in combination with the DQPSK modulator to set up a DQPSK transmission system. Its basic working principle is depicted below [1]

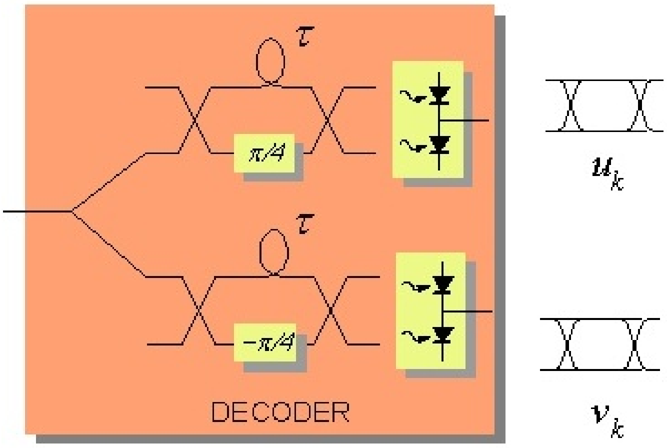


Figure 7.15.15. Working Principle of the DQPSK Demodulator

Component settings

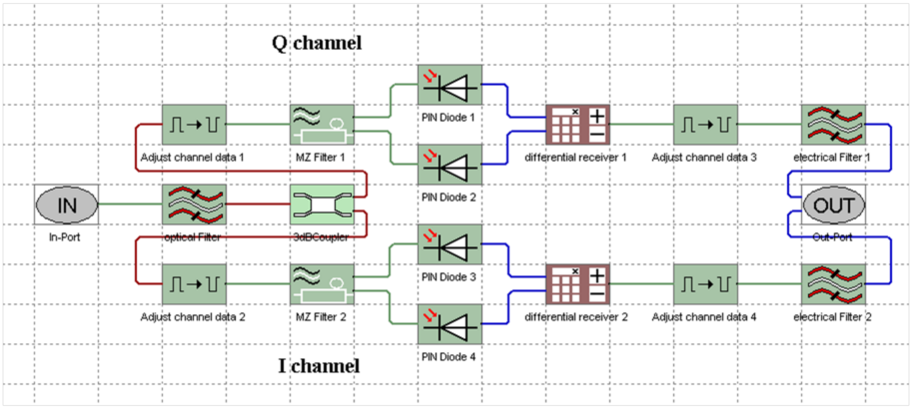


Figure 7.15.16. DQPSK Demodulator Setup

First, the input signal is filtered by a 3rd-order 12.5 GHz Gaussian filter. The signal is then divided into two parts, one for each channel. The succeeding pattern modifier replaces the bit pattern of the encoded bit stream with the unencoded bit pattern. This is necessary for a later analysis of the signal by means of eye analyzers and bit error rate testers.
Afterwards, the signal is split by 3dB couplers. One part of the signal is delayed by 100 ps, (one bit period at 10 GBit/s) and the other one is phase shifted by 45°. Then the two signals are recombined by another 3dB coupler and separated again to feed the two PIN diodes. Together with the calculator 'Signal 2 - Signal1' the PIN diodes form a balanced receiver.
The succeeding electrical filter is a 10th-order 14 GHz Bessel filter that should minimize the noise of the signal. Since the calculator introduces a second channel to the parametric signal, it has to be removed

again by the pattern modifier 'Delete superfluous channel'. This pattern modifier also adjusts the minimal, maximal and average power of the channel to 0 W, 0.75 mW and 0.375 mW respectively. This is done due to the fact that the phase modulation is converted by the DQPSK demodulator into an amplitude modulation with different power levels. Without this adjustment of the parametric signal a succeeding eye analysis will not work correctly.

Input / Output

	Number	Type of signal
Input	1	Optical Signal
Output		no output

Component Parameters

Parameter Name	BR
Description	Bitrate (2x Symbolrate)
Default Value	10
Parameter Range	0 < BitRate
Unit	GBit/s

Parameter Name	avgPower
Description	Average signal power
Default Value	0.001
Parameter Range	not implemented
Unit	W

References

[1] R. Griffin, “Optical Differential Quadrature Phase-Shift Key (oDQPSK) for High Capacity Optical Transmission”, OFC Proceedings, 2002

7.15.8 10 Gb/s SSB Source

Fundamentals

This user-defined model represents an optical single sideband source. The source is modeled using a Pulse Generator and a filter with a distinct parameter set.

Component settings

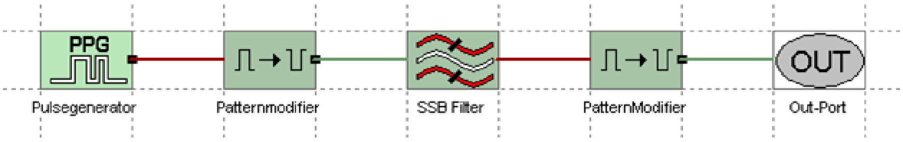


Figure 7.15.17. Simulation Setup

The signal source generates an ordinary 10 GBit/s NRZ OOK signal with a carrier frequency of 193.1 THz. This signal is filtered by the Gaussian-shaped SSB filter with a center frequency of 193.11 THz and a bandwidth of 20 GHz.
To avoid deleting the channel due to a shifted center frequency of the filter with respect to the center frequency of the pulse generator, the parametric signal is adjusted before and after the filter. For this purpose two Pattern Modifiers, which modify the parameterized center frequency temporarily, are deployed.

Characteristics

The output spectrum of the model is depicted in figure 7.15.18.

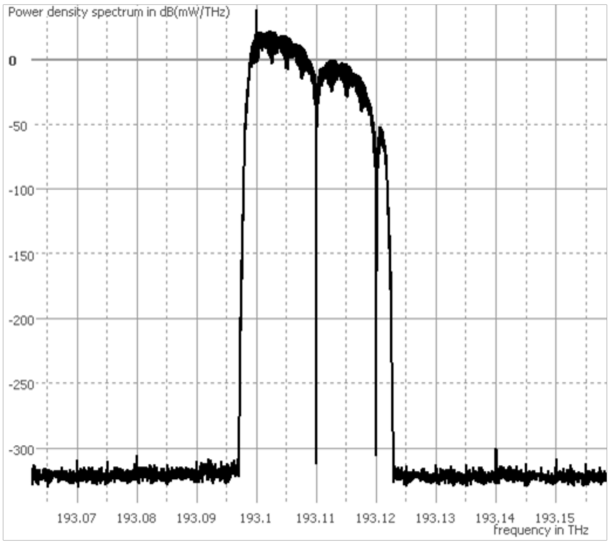


Figure 7.15.18. Spectrum of SSB Model

Input / Output

	Number	Type of signal
Input		no Input
Output	1	Optical Field

7.15.9 10 Gb/s VSB Source

Fundamentals

This user-defined model represents an optical vestigial-sideband source. The source is modeled using a pulse generator and a filter with a distinct parameter set.

Component settings

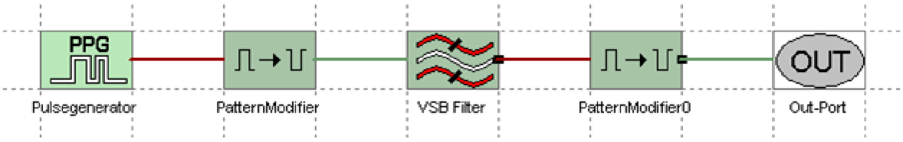


Figure 7.15.19. Simulation Setup

The signal source generates an ordinary 10 GBit/s NRZ OOK signal with a carrier frequency of 193.1 THz. This signal is filtered by a Gaussian-shaped VSB filter with a center frequency of 193.105 THz and a bandwidth of 15 GHz.
To avoid deleting the channel due to a shifted center frequency of the filter with respect to the center frequency of the pulse generator, the parametric signal is adjusted before and after the filter. For this purpose two pattern modifiers, which modify the parameterized center frequency temporarily, are deployed.

Characteristics

The output spectrum of the model is depicted in figure 7.15.20.

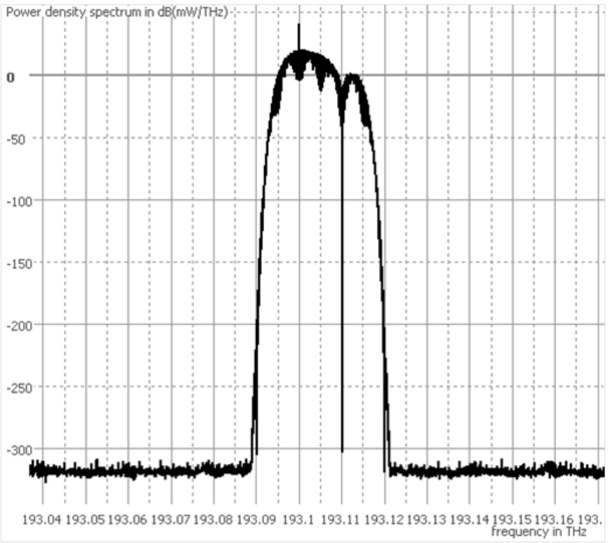


Figure 7.15.20. Spectrum of VSB Model

Input / Output

	Number	Type of signal
Input		no Input
Output	variable	Optical Field

Index