User Guide v-1

Mode Solver
User Guide

SemiVi LLC
Zollikon, Switzerland.

October 11, 2025

Contents

1 Introduction **1**
- 1.1 Features . . . 1
- 1.2 Installation . . . 2
- 1.3 Licensing . . . 2
  - 1.3.1 Purchasing the licenses . . . 3
  - 1.3.2 Installation of SemiVi-activator . . . 3
  - 1.3.3 License activation . . . 3

2 Theory of Mode Solver **5**
- 2.1 Derivation of Mode Equation . . . 5
- 2.2 Vectorial Mode Solver . . . 6
- 2.3 Scalar Mode Solver . . . 7
- 2.4 Calculation of modal electric field and magnetic flux . 8
  - 2.4.1 Normalized modal $\vec{E}$ and $\vec{H}$ . . . 8

3 Configuration File Structure **9**
- 3.1 File Section . . . 10
- 3.2 Mode Solver Section . . . 11
- 3.3 Running Mode Calculation . . . 11
- 3.4 Output Files . . . 12

4 Configuring Mode Solver **15**
- 4.1 Mandatory Inputs . . . 15
- 4.2 Available Inputs . . . 17
- 4.3 Available Solver Settings . . . 18
- 4.4 Miscellaneous Comments . . . 19

4.4.1 Coordinate Transformation 19
4.4.2 Selecting 1D/2D Solver 19
4.4.3 Selecting Numeric Method 19

5 Python Interface **21**
5.1 Import modules 21
5.2 Constructors 22
5.3 Setup the solver 22
5.3.1 setGlobalParameters 22
5.4 Solve for Mode Calculations 24
5.4.1 solveAllModes 24
5.4.2 solveOneMode 24

6 Visualization of Results **25**
6.1 Naming Convention 25
6.2 Visualization 26

A Notation and Acronyms **27**
Acronyms 27

Chapter 1

Introduction

Mode solver is a part of OptoSolver package. It is used to perform mode calculations on 1D or 2D cross-section of a waveguide. The cross-section itself can be provided as a 2D structure. Alternately, the entire waveguide structure can be provided and coordinate of the cross-section at which mode calculation is to be performed can be supplied. Modal electric fields and magnetic fluxes are are calculated by the mode solver and are stored for visualization.

Mode solver is also used for mode calculations which are invoked in Finite Difference Time Domain (FDTD) simulator and Beam Propagation Method (BPM) solver.

1.1 Features

Mode solver supports materials with constant real and imaginary permittivity. If the material config file contains wavelength vs. refractive index table, then wavelength dependent permittivity can be used as well. Calculation of modes with complex effective index can be performed using the 'power method'.

Mode solver outputs modal electric fields and magnetic fluxes in hdf5 file. Additionally, an xdmf file is also written. It can be used to visualize data stored in hdf5 file in paraview.

1.2 Installation

SemiVi currently supports software installation on various Linux distributions. The software installer is available in Debian package (*.deb file) and in RPM format (*.rpm file).

Note, that *if you have downloaded mkl version of the OptoSolver*, the following package needs to be installed manually by you before installing the circuit solver from the installer package.

- Intel math kernel libraries (released in 2020 or later), which include distributions of open-mp, pardiso, etc. specific for Intel processors.

The OptoSolver sources mkl functions from the above installation. These functions can offer speed-up in the calculations on Intel processors. The mkl package can be downloaded from Intel website.

If the OptoSolver without mkl-acceleration is downloaded, then installation of the above package is not necessary.

Once all the above packages are installed, download the installer on the local machine. The installer file named `optosolver_amd64.deb` will appear in the `Downloads` directory. Go to the directory using `cd` command. Use the following command to install the `optosolver` from the installer.

```
>> sudo apt install ./optosolver_amd64.deb
```

Alternately, one may use `dpkg` to install the software and use `apt` to install missing dependencies as follows.

```
>> sudo dpkg -i ./optosolver_amd64.deb
>> sudo apt install -f
```

You need to have `root` access to install the software on your machine.

1.3 Licensing

Two types of licenses can be purchased for SemiVi Mode solver.

Node-locked licenses enable unlimited number of *simultaneous* executions of the Mode solver on the client machine. The node-locked license limits the usage of the Mode solver only to the machine on which the license is activated.

With *server licenses*, the Mode solver can be run on any of the machines in the client organization on which the server license is activated. However, only the specified number of *simultaneous* executions are possibie at a time.

1.3.1 Purchasing the licenses

The clients can place order for any of the above licenses on SemiVi website (https://www.semivi.ch/sales) or by contacting our sales-person.

We will process the request and send the license files by email. The license files need to be activated on the desired machines using the license key which is emailed separately using the following command.

1.3.2 Installation of SemiVi-activator

The license file must be activated on the desired computer before use. For that purpose, download the installer `semivila_amd64.deb` file on the local machine and install it as follows.

```
>> sudo apt install ./semivila_amd64.deb
```

1.3.3 License activation

To activate the license file, please run the following command.

```
>> semivila -a File.lic <Server|NodeLocked>License.lic\\
```

Replace `File.lic` with the your license file, and use appropriate name for the activated file. You will be prompted to input the 16 digit license key. A successful activate of the license file will generate the activated license file. Copy the activated license file to the `/opt/semivi/licenses/` folder and rename it to `ServerLicense.lic` or `NodeLockedLicense.lic` for server and node-locked licenses respectively. If you have more than one license files, please delete the older

expired license files. If you wish to keep more than one active license files, you can also name the license files as `<i>NodeLockedLicense.lic` where `<i>` could be from 0 to 49. For ex. `49NodeLockedLicense.lic` or `49ServerLicense.lic`. The program will read the license files and lock the first available license. All the target users must have read rights on the license file.

User-guides of all the software provided by SemiVi are stored at the location `/opt/semivi/userguides/`.

Tutorials of all the software provided by SemiVi are stored at the location `/opt/semivi/tutorials/optosolver`.

Chapter 2

Theory of Mode Solver

2.1 Derivation of Mode Equation

Maxwell's equations in non-homogeneous media without free charges and free currents are given by,

$$\nabla \cdot \vec{D} = 0, \nabla \cdot \vec{B} = 0, \nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t}, \nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} \tag{2.1}$$

Here, $\vec{D} = \epsilon \vec{E}$ is a displacement vector, E is an electric field, $\vec{H} = \vec{B}/\mu$ is the magnetic flux, and $\vec{B}$ is the magnetic field. Applying curl to the third Maxwell's equation, Right Hand Side (RHS) of the third equation is simplified as,

$$-\frac{\partial \nabla \times \vec{B}}{\partial t} = -\mu \frac{\partial \nabla \times \vec{H}}{\partial t} = -\mu \frac{\partial^2 \vec{D}}{\partial t^2} = -\mu\epsilon \frac{\partial^2 \vec{E}}{\partial t^2}. \tag{2.2}$$

In the above simplification, we have assumed that μ is spatially homogeneous and both ϵ and μ are time-independent. The complete equation can now be written as a function of $\vec{E}$.

$$\nabla \times \nabla \times \vec{E} = -\mu\epsilon \frac{\partial^2 \vec{E}}{\partial t^2} = -\frac{n^2}{c^2} \frac{\partial^2 \vec{E}}{\partial t^2} \tag{2.3}$$

Here, c is the speed of light in free space and n is spatially varying refractive index.

In a waveguide propagating in x-direction, $\vec{E}$ and $\vec{H}$ have constant envelopes in x-direction with spatial variation coming from spatially sinusoidal $\exp(i\beta x)$ term. They can be written as,

$$\vec{E}(x,y,z) = (E_x(y,z)\hat{x} + E_y(y,z)\hat{y} + E_z(y,z)\hat{z})\exp(i\beta x) \quad (2.4a)$$

$$\vec{H}(x,y,z) = (H_x(y,z)\hat{x} + H_y(y,z)\hat{y} + H_z(y,z)\hat{z})\exp(i\beta x) \quad (2.4b)$$

Substituting waveguide $\vec{E}$ in Eq. 2.3, we get

$$\nabla \times \nabla \times \vec{E}(x,y,z) = \frac{\omega^2 n(y,z)^2}{c^2}\vec{E}(x,y,z). \quad (2.5)$$

Property of the waveguide that $\frac{\partial E_x}{\partial x} = \frac{\partial E_y}{\partial x} = \frac{\partial E_z}{\partial x} = 0$ can be used by splitting the operator $\nabla = \nabla_t + \hat{x}\frac{\partial}{\partial x}$ where $\nabla_t = \hat{y}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z}$. Splitting the operator accordingly and using the equality $\nabla \times \nabla \times \vec{E} = \nabla(\nabla\dot{E}) - \nabla^2\vec{E}$ and Poisson equation $\nabla(\dot{\epsilon\vec{E}}) = 0$, the wave equation Eq. 2.3 can be written as,

$$-\nabla_t^2\vec{E}_t + \nabla_t\left[\nabla_t \cdot \vec{E}_t - \frac{1}{n^2}\nabla_t \cdot (n^2\vec{E}_t)\right] - \frac{\omega^2}{c^2}n^2\vec{E}_t = -\beta^2\vec{E}_t \quad (2.6)$$

2.2 Vectorial Mode Solver

When ϵ of the cross-section of a waveguide is strongly inhomogeneous, then the second term on Left Hand Side (LHS) can no longer be ignored. In that case, Eq. 2.6 needs to be solved to calculate mode propagation constant β. It can be put in another form as,

$$\begin{bmatrix} \hat{P}_{yy} & \hat{P}_{yz} \\ \hat{P}_{zy} & \hat{P}_{zz} \end{bmatrix} \begin{bmatrix} E_y \\ E_z \end{bmatrix} = -\beta^2 \begin{bmatrix} E_y \\ E_z \end{bmatrix} \quad (2.7)$$

where each individual operator is given below,

$$\hat{P}_{yy} \equiv -\frac{\partial}{\partial y}(\frac{1}{n^2}\frac{\partial}{\partial y}n^2) - \frac{\partial^2}{\partial z^2} - \frac{\omega^2}{c^2}n^2 \quad (2.8a)$$

$$\hat{P}_{yz} \equiv -\frac{\partial}{\partial y}(\frac{1}{n^2}\frac{\partial}{\partial z}n^2) + \frac{\partial^2}{\partial y \partial z} \quad (2.8b)$$

$$\hat{P}_{zy} \equiv -\frac{\partial}{\partial z}(\frac{1}{n^2}\frac{\partial}{\partial y}n^2) + \frac{\partial^2}{\partial z \partial y} \tag{2.8c}$$

$$\hat{P}_{zz} \equiv -\frac{\partial^2}{\partial y^2} - \frac{\partial}{\partial z}(\frac{1}{n^2}\frac{\partial}{\partial z}n^2) - \frac{\omega^2}{c^2}n^2 \tag{2.8d}$$

The above equations have been discretized on a rectangular grid. Solving the matrix Eq. 2.6, is same as finding eigenvectors and eigenvalues of the matrix on the LHS of Eq. 2.7. The eigenvectors are called modes and eigenvalues are squared effective indexes of the propagating waveguide.

2.3 Scalar Mode Solver

When one component of the transverse field is dominant over the other component and effective index differences are small among different regions, then one may ignore $\frac{\nabla\epsilon}{\epsilon}$. In that case, the second term in Eq. 2.6 can be ignored. If `Polarization` is set to Transverse Magnetic (TM), E_z is the dominant field. If `Polarization` is set to Transverse Electric (TE), E_y is dominant. If E_y is dominant over E_z, the mode equation can be written as,

$$-\frac{\partial^2}{\partial y^2}E_y - \frac{\partial^2}{\partial z^2}E_y - \frac{\omega^2}{c^2}n^2E_y = -\beta^2 E_y \tag{2.9}$$

. The above equation has been discretized on the rectangular grid. Solving the equation is same as finding eigenvectors and eigenvalues of the discretized form of the above equation. If `Polarization` is set to TM, eigenvectors calculated above correspond to the dominant component E_z and $E_y = 0$. If `Polarization` is set to TE, eigenvectors correspond to E_y and $E_z = 0$.

2.4 Calculation of modal electric field and magnetic flux

Only transverse components of electric field (E_y and E_z) are calculated by the mode solver. Longitudinal component E_x is calculated for each mode as a post-processing step. It is calculated as follows,

$$E_z = -\frac{\nabla_t \cdot (n^2 \vec{E}_t)}{i\beta n^2} \tag{2.10}$$

The above equation completes calculation of modal $\vec{E}$. Modal $\vec{H}$ can be calculated as follows,

$$\vec{H} = \frac{\nabla \times \vec{E}}{i\omega\mu_0} \tag{2.11}$$

2.4.1 Normalized modal $\vec{E}$ and $\vec{H}$

After calculating modal $\vec{E}$ and $\vec{H}$, they are normalized such that the integration of the magnitude of Poynting vector is unity. That is,

$$\int \vec{E} \times \vec{H}^* \cdot d\vec{S} = 1.0 \tag{2.12}$$

.

Chapter 3

Configuration File Structure

Optosolver software reads various inputs from mode solver configuration file and performs mode calculation on the device structure. The program can be executed using the following command –

```
>> OptoSolver modesolver modeSiWG_dev.cfg
```

In the above command, the word after `Optosolver` is the name of the program to be executed (in this case – `modesolver`). The program name is followed by the configuration file name (in this case – `modeSiWG_dev.cfg`). A sample configuration file of the mode solver is provided below.

```
File:
{
  Device = "modeSiWG_str.cfg";
  Out = "SiWG";
}

Solver:
{
```

```
  Name = "Def";
  LaserType = "FabryPerot";
  Polarization = "TM"; // applicable for scalar equations only
  Equation = "Vectorial";
  Wavelength = 1.; // in Micrometer
  EffectiveIndex = 3.07;
  DecayConstant = 0.;

  MaximumModes = 1;
  SolverSettings = [
    "FullDevice",
    "UsePowerMethod",
    "WavelengthDepIndex"
    ];

  PowerMethodTol = 1E-8;
  PowerMethodMaxIter = 20;
}
```

The config file is composed of two sections – 1. File 2. Solver. Various keywords in each of the section and their functionality is described below.

3.1 File Section

The keyword `Device` provides the file name from which the device structure is created. Internally, the file is processed differently according to its extension.

- If the file extension is "str.cfg", the file is processed as an input file for the tensor mesh generation.
- If the file extension is "str.h5", The file is read as hdf5 file generated by the structure and tensor mesh generator.

The keyword `Out` sets the prefix to the output file name. In this case, the output files will be called 'SiWG_mode.xdmf' and 'SiWG_mode.h5'.

3.2 Mode Solver Section

This section lists all the information needed for the mode solver other than the device structure. Output files will use the word specified by "Name" in mode solver section as a prefix (in the above file – "Def"). Since the modes of a Fabry-Perot laser are required, LaserType is set to "FabryPerot". Mode polarization is required, when the 2D "Scalar" laser equation is solved or 1D laser equation is solved. The keyword "Wavelength" provide laser wavelength (in μm). Initial effective index and around which the modes are searched is given by the keyword "EffectiveIndex". "DecayConstant" sets imaginary part of the effective index. "MaximumModes" specifies the number of modes to be calculated.

Various flags are provided as a list of comma-separated strings with the keyword "SolverSettings". In the above file, a 'FullDevice' is simulated (instead of a cross-section of the device). If the flag "WavelengthDepIndex" is set, effective index of the material is obtained from the wavelenth vs. refractive index table written in the Material config file for each of the material in the structure. If the keyword "UsePowerMethod" is listed, then a faster 'power method' is used for mode calculation.

3.3 Running Mode Calculation

In this section, the above config file is used to perform mode calculations on a 2D-cross section of a waveguide shown in Fig. 3.1. The structure is created using the command

```
>> OptoSolver str modeSiWG_str.cfg.
```

It is not necessary to generate the structure before simulating it. The config file for generating the structure ('modeSiWG_str.cfg') can be specified as `Device` in `File` section of the mode solver config file modeSiWG_dev.cfg'. The solver internally generates the structure and passes it to the mode solver. For correct mode simulations, the structure config file must also be present in the same folder. Once the config file is set, the mode calculations can be performed using the following command

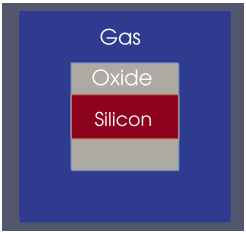


Figure 3.1: Structure of the cross-section of a Fabry-Perot Laser in YZ plane.

```
>> OptoSolver modesolver modeSiWG_dev.cfg
```

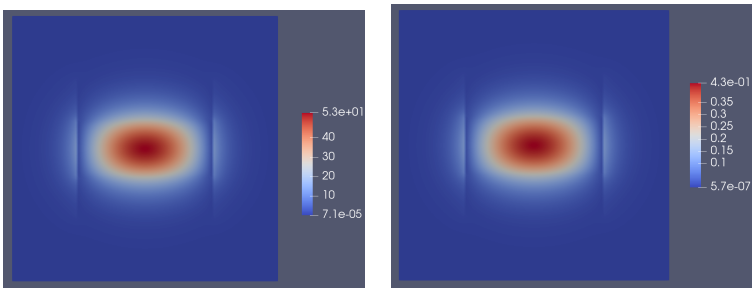
The mode solver calculations will generate and xdmf file (extension *.xdmf) together with an hdf5 file (extension *.h5).

3.4 Output Files

The keyword `Out` in the `File` section of the config file sets the prefix to the output file name. In this case, the output files are called 'SiWG_mode.xdmf' and 'SiWG_mode.h5'. Note, that the xdmf file is simply an XML script which provides additional information on various datasets stored in the hdf5 file for visualization purpose. If Paraview is installed on the machine, the xdmf file can be opened using the following command

```
>> paraview SiWG_mode.xdmf
```

In paraview, various quantities such as, X, Y, and Z components of modal electric field and magnetic flux, magnitude of modal electric field can be selected for visualization. Naming convention for each of the modal quantity is as follows, "ElectricFieldY_Wl_α_Mo_β", where α is the wavelength id in the list of wavelength(s) input by the user and $\beta \in [0, \text{MaximumModes})$ is the mode id. Fig. 3.2 plots electric field in Y-direction and magnetic flux in Z-direction. Note, that the cross-section of the waveguide is in YZ plane. Thus, the horizontal



(a) Electric Field (b) Magnetic Flux

Figure 3.2: Calculated modal electric field along Y direction and magnetic flux along Z direction of the waveguide in Fig. 3.1 are shown in the figures above.

axis is Y-axis and the vertical axis is Z-axis. Also, the electric fields and magnetic fluxes are normalized such that the integration of the magnitude of Poynting vector is unity. That is,

$$\int \vec{E} \times \vec{H}^* \cdot d\vec{S} = 1.0 \tag{3.1}$$

Chapter 4

Configuring Mode Solver

An example configuration file provided in Chapter 3 lists typical configurations. In the chapter, a list of all the configurations and their usage information is provided.

4.1 Mandatory Inputs

Keywords in the `File` section necessary for running the mode calculation are listed below.

- `Device:` Specify either a config file for structure generation or a saved mesh file in hdf5 format.
- `Out:` Specify prefix of the output xdmf and hdf5 files. The output files are named '<out>_mode.xdmf' and '<out>_mode.h5', where '<out>' is the string input by user in `Out`.

Keywords in the `Solver` section necessary for running the mode calculation are listed below.

- `Wavelength:` Wavelength of Fabry-Perot laser cavity must be specified in the units of μm.

- `EffectiveIndex:` Effective refractive index of the waveguide around which the waveguide modes are searched.

- `DecayConstant:` Initial value of the decay constant of the waveguide. When complex mode equation is solved, waveguide modes are searched around complex effective index $n' = n + \kappa\iota$ where n and κ stand for the effective index and the decay constant respectively.

- `Wavelengths:` A list of wavelengths is provided at which mode calculations are performed iteratively. The list is specified in the format shown below.
  ```
  Wavelengths = [0.4, 0.6, 0.8, 1.0];
  ```
  Either `Wavelength` must be set to a floating point number, or the field `Wavelengths` must be set as the list of floating point numbers as shown above.

- `EffectiveIndexes:` A list of effective index values around which the modes are searched in the iterative calculations above. The list is specified in the format shown below.
  ```
  EffectiveIndexes = [3.2, 3.3, 3.4, 3.6];
  ```
  Unless `WavelengthEffectiveIndexGrid` is activated in `SolverSettings`, the number of entries in the above list must be same as the number of entries in `Wavelengths` list. Either `EffectiveIndex` must be set to a floating point number, or `EffectiveIndexes` must be set as the list of floating point numbers as shown above.

- `DecayConstants:` A list of initial values of the decay constant around which the modes are searched in the iterative calculations above. The list is specified in the format shown below.
  ```
  DecayConstants = [0.1, 0.2, 0.3, 0.5];
  ```
  Note, that the number of entries in the above list must be same as the number of entries in `EffectiveIndexes` list.

- `CoordinateCut:` It specifies that the mode calculations are performed along the device cross-section at the nearest grid point along the specified axis. This entry is required only if one of the `XCutDevice`, `YCutDevice`, or `ZCutDevice` is activated in `SolverSettings`.

4.2 Available Inputs

Available keywords in the `Solver` section are listed below. They provide the user a possibility to update default values of various parameters before the mode calculations.

- `LaserType:` Type of laser cavity is specified. Currently only 'FabryPerot' input is accepted.
- `Polarization:` For the mode calculation at 1D cross-section of the waveguide or 2D cross-section with Scalar mode equation, polarization of the waveguide must be specified by the user. Polarization can be 'TE' or 'TM', which stand for 'Transverse Electric' or 'Transverse Magnetic', respectively. Default value is 'TE'.
- `Equation:` For mode calculation at 2D cross-section of the waveguide, either 'Scalar' or 'Vectorial' mode equation can be solved. Equation can be either 'Scalar' or 'Vectorial'. Default value is 'Scalar'.
- `MaximumModes:` Number of modes to be calculated near the user-provided effective index. Default value is 1.
- `PowerMethodTol:` Applicable only when power method is used for mode calculations. When difference between the eigenvalues between successive iterations is less than the tolerance then the calculations are terminated. Default value is 10^{-8}.
- `PowerMethodMaxIter:` Maximum iterations for mode calculations using the power method. Applicable only when power method is used for mode calculations. Default value is 50.
- `SolverSettings:` Various flags are set by providing a list of appropriate keywords as comma-separated strings as follows.

```
SolverSettings = ["FullDevice",
                  "UsePowerMethod",
                  "WavelengthDepIndex"];
```

4.3 Available Solver Settings

Keywords which may be listed in the field `SolverSettings` to set certain flags in the mode solver are listed below. Note, that they are listed in the decreasing order of priority. In the case of a conflict, the upper keyword has higher priority over the lower keyword.

1. `WavelengthDepIndex`: Real and imaginary dielectric constants are determined from the table of wavelength dependent complex refractive indices specified in the material config file. Linear interpolation is used if the wavelength is not in the list. If the wavelength is outside of the table range, then the dielectric constants are set to smallest/largest frequency

2. `UsePowerMethod`: Power method is used for mode calculations, else Arpack routines are used.

3. `modeXcut`: Mode along YZ plane of the device at x-location specified by `CoordinateCut`.

4. `modeYcut`: Mode along XZ plane of the device at y-location specified by `CoordinateCut`.

5. `modeZcut`: Mode along YZ plane of the device at z-location specified by `CoordinateCut`. If the device structure is 2D, then full device is used for the calculations.

6. `Absorption`: Complex refractive index of the materials is used for mode calculations, and the modes with complex index are calculated. At the moment, Arpack routines give error, if this keyword is used.

7. `ZAxisPBC`: Periodic boundary conditions are used at the faces normal to Z-axis.

8. `YAxisPBC`: Periodic boundary conditions are used at the faces normal to Y-axis.

9. `WavelengthEffectiveIndexGrid`: Mode calculations are performed for every effective index value at every wavelength in the wavelength list.

Note, that priority of the keywords *does not* depend on their order in the list supplied with `SolverSettings`.

4.4 Miscellaneous Comments

4.4.1 Coordinate Transformation

The mode solver always transforms the axes, such that the waveguide is oriented along X-axis. That is, the waveguide cross-section is always in YZ-plane. If 1D solver is used, then the field profiles are stored along Y-axis. On the other hand, if 2D solver is used, then the field profiles are stored on a grid in YZ plane.

4.4.2 Selecting 1D/2D Solver

The mode solver selects appropriate solver dimension depending on the user inputs. If a 3D device is supplied and an X, Y, or Z-cut is specified in the settings, then the cross-section is 2D. Therefore a 2D solver is used. If a 2D device is supplied and `modeZcut` is set, then the device itself is treated as a cross-section and a 2D solver is used. The output data is stored and plotted on a 2D grid.

If a 2D device is supplied and `modeXcut` or `modeYcut` is set, then 1D cross-section along Y, or X axis is used for mode calculation using 1D mode.

4.4.3 Selecting Numeric Method

Two numeric methods are provided for mode calculations, namely 1. Arpack routine (default) 2. Power method (Add `UsePowerMethod` to the `SolverSettings`).

Power method yields the results faster than the Arpack routine. It may yield inaccurate modes with indices off their true values, when `MaximumModes` > 1. It is recommended to use power method with `MaximumModes` > 1 to search approximate values of the effective indices. Then, set effective index to one of the values obtained before, and use `MaximumModes` $= 1$, to get the desired mode profile.

At the moment, Arpack routine gives an error when complex index is activated (`Absorption`). Therefore, for mode calculations using complex index, always activate `UsePowerMethod`.

Chapter 5

Python Interface

The Opto-solver package provides a python module to perform FDTD, BPM, and mode simulations using a python script. The package also provides commands to retrieve simulation results. Together with `tensormesher` python interface, it enables users to construct a device, simulate it, and post-process the results using a python script. This would come handy for structure optimization for specific applications.

This chapter describes python interface of the mode solver.

> **Note:** Whenever possible, please use config file to setup the mode solver object. Providing config file ensures that all the data are input in the correct order.

5.1 Import modules

Python modules of the Opto-solver and the tensor-mesher package are imported using the following script.

```
import numpy as np
import cutensormesher as m
import cuoptosolver as s
```

Note, that if you have downloaded hardware-accelerated version of the optosolver, then you must import the modules with prefix `cu` as shown above. Else, import `tensormesher` and `optosolver`. Do not mix them.

5.2 Constructors

Three constructors are provided for the BPM-solver, as follows.

- `s.bpmsolver(Device=dev)` constructs the solver object by taking `m.device()` object `dev` provided by the tensor-mesher as an input.
- `s.bpmsolver(CmdFile="file.cfg")` constructs the solver object by parsing config file of the BPM-solver. Note, this is exactly the same file as described in Chapter 3. It also imports device structure or mesh from the `File`section.
- `s.bpmsolver(CmdFile="file.cfg",Device=dev)` constructs the solver object by parsing config file of the BPM-solver. Note, this is exactly the same file as described in Chapter 3, except that device given as an argument is used in the solver.

5.3 Setup the solver

The command given below is called on the `modesolver` object. It sets up the solver, e.g. specify boundary conditions, etc.

5.3.1 setGlobalParameters

The command `setGlobalParameters()` sets solver various parameters on a global scope. It takes the following arguments.

1. `NumericParams` : A python dictionary mapping parameter name to its numeric value. The following numeric parameters can be supplied.
    - `Wavelength` : Wavelength at which the mode is solved.
    - `EffectiveIndex` : Approximate effective index around which the mode is to be solved. Exact effective index is calculated by mode calculations.
    - `DecayConstant` : Approximate imaginary part of the effective index.

- `CoordinateCut` : coordinate of the cut-plane. The device is cut along the cut-plane and solved for mode.
- `MaximumModes` : Maximum number of modes to be solved.
- `PowerMethodTol` : If power method is used for mode calculations, tolerance of the power method is set.
- `PowerMethodMaxIter` : Maximum iterations for the power method.

2. `StringParams` : A python dictionary mapping parameter name to a string. The following string parameters can be supplied.
   - `Equation` : Possible alternatives - 'Scalar', or 'Vectorial'.
   - `LaserType` : Currently 'FabryPerot' mode can be solved.
   - `Polarization` : Possible alternatives - TM or TE.
   - `Out` : Prefix of the output file in which modal fields are stored.
3. `NumericListParams` : A python dictionary mapping parameter name to a list of numbers. Currently, the `InterfacesX` is recognized. It accepts a list of x-coordinates of the locations where the waveguide effective index is changed, perhaps due to a different cross-sectional geometry.
4. `StringListParams` : A python dictionary mapping parameter name to a list of string. Currently only `Settings` parameter is recognized. The following strings can be supplied to the `Settings` parameter.
   - `WavelengthDepIndex`: Use wavelength-dependent refractive index to calculate permittivity.
   - `Absorption`: Propagate BPM with complex permittivity to take into acount decay of the waveguide amplitude.
   - `UsePowerMethod`: Use power method to calculate modes instead of eigenvalue calculations.
   - `XAxisPBC` or `YAxisPBC` or `ZAxisPBC`: Set periodic boundary condition along the specified axes.

- `XCutDevice` or `YCutDevice` or `ZCutDevice` : Any one of these flags can be listed. `XCutDevice` means the mode is to be calculated in the YZ-cutplane at x-coordinate given by `CoordinateCut` parameter.

5.4 Solve for Mode Calculations

The following commands begin the simulations.

5.4.1 solveAllModes

The command `solveAllModes()` solves for the specified number of modes at the specified set of wavelengths at the specified effective indices. The wavelengths, effective indices are specified in the `setGlobalParameters` command.

5.4.2 solveOneMode

The command `solveOneMode(...)` solves for only one mode at the wavelength given by argument `Wavelength`, and at the effective index and decay constants given by the arguments `EffectiveIndexRe` and `EffectiveIndexIm`, respectively. It is useful to quickly compute a mode.

Chapter 6

Visualization of Results

Real and imaginary parts of X-, Y-, and Z- components of electric field and magnetic flux are calculated at the 1D/2D grid of the waveguide cross-section. They are stored in an hdf5 file. Additionally, a Xdmf script file is written. The output files are named '<out>_mode.xdmf' and '<out>_mode.h5', where '<out>' is the string input by user with the keyword `Out` in `File` section of mode solver config file. Also, effective indexes corresponding to each of the wavelengths and each of the mode are stored in '<out>_mode.csv' file. It can be viewed using a text editor or a csv file viewer program.

6.1 Naming Convention

Number of modes specified by `MaximumModes` are calculated at each wavelength in the list of wavelengths input by the user. The wavelengths are represented by their index in the wavelength list. Each of these modes have a unique electric field and magnetic flux distribution. All the stored quantities are named as '<quantity>_Wl_<wlid>_M0_-<modeid>'. Here, '<wlid>' is the wavelength index, '<modeid>' is mode index, and '<quantity>' is any one of the following modal quantities.

- ElectricFieldX: Real part of electric field along X

- ElectricFieldY: Real part of electric field along Y
- ElectricFieldZ: Real part of electric field along Z
- ImElectricFieldX: Imaginary part of electric field along X
- ImElectricFieldY: Imaginary part of electric field along Y
- ImElectricFieldZ: Imaginary part of electric field along Z
- AbsElectricField: Absolute value of electric field along Z
- MagneticFluxX: Real part of magnetic flux along X
- MagneticFluxY: Real part of magnetic flux along Y
- MagneticFluxZ: Real part of magnetic flux along Z
- ImMagneticFluxX: Imaginary part of magnetic flux along X
- ImMagneticFluxY: Imaginary part of magnetic flux along Y
- ImMagneticFluxZ: Imaginary part of magnetic flux along Z

6.2 Visualization

The output xdmf file is an xml script which links the grid and other attributes to the mode quantities. All the quantities stored in the output hdf5 file can be viewed in the visualization software 'Paraview' using the following command.

```
>> paraview SiWG_mode.xdmf
```

Appendix A

Notation and Acronyms

Acronyms

BPM	Beam Propagation Method
FDTD	Finite Difference Time Domain
LHS	Left Hand Side
RHS	Right Hand Side
TE	Transverse Electric
TM	Transverse Magnetic

Bibliography