User Guide v-1

Beam Propagation Method Simulator User Guide

SemiVi LLC

Zollikon, Switzerland.

October 11, 2025

Contents

Chapter 1

Introduction

BPM solver in Optosolver package is used to calculate Slowly Varying Envelope (SVE) of all the components of electric field and magnetic flux along the waveguide by using BPM method. Both 2D and 3D waveguide structures can be provided as inputs.

BPM solver uses Mode solver in Optosolver package internally for necessary mode calculations to get effective waveguide index.

1.1 Features

BPM solver supports materials with constant real and imaginary permittivity. If the material config file contains wavelength vs. refractive index table, then wavelength dependent real and imaginary permittivity can be used as well.

BPM solver also supports bidirectional BPM calculations which can be used to calculate reflection coefficient. This support is on experimental basis only.

BPM solver stores SVE of all the components of electric field and magnetic flux in an hdf5 file. An xdmf script file is also created for visualizing the results 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 the above package is 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 BPM solver.

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

With *server licenses*, the BPM 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 salesperson.

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 BPM Solver

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

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

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

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

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 $\vec{\mathcal{E}}$.

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

BPM deals with propagation of monochromatic light in the waveguide. $\vec{\mathcal{E}}$ due to the monochromatic light can be written as

$$\vec{\mathcal{E}}(\vec{r}) = \vec{E}(\vec{r}) \exp(i\omega t). \tag{2.4}$$

Substituting this in Eq. 2.3, and using $\nabla \times \nabla \times \vec{E} = \nabla^2 \vec{E} - \nabla(\nabla \cdot \vec{E})$ we get,

$$\nabla^2 \vec{E} - \nabla(\nabla \cdot \vec{E}) + n^2 k_0^2 \vec{E} = 0 \tag{2.5}$$

.

In BPM implementation, mode is set to propagate in x- direction. Splitting ∇ operator and $\vec{E}$ into transverse and longitudinal components, we get

$$\vec{E}(\vec{r}) = E_x(\vec{r})\hat{x} + E_y(\vec{r})\hat{y} + E_z(\vec{r})\hat{z} = E_x(\vec{r})\hat{x} + \vec{E}_t(\vec{r}) \tag{2.6a}$$

$$\nabla = \frac{\partial}{\partial x}\hat{x} + \frac{\partial}{\partial y}\hat{y} + \frac{\partial}{\partial z}\hat{z} = \frac{\partial}{\partial x}\hat{x} + \nabla_t \tag{2.6b}$$

.

Using the above notations, and separating Eq. 2.5 into transverse and longitudinal components, the transverse component can be written as,

$$\nabla^2 \vec{E}_t - \nabla_t(\nabla_t \cdot E_t + \frac{\partial E_x}{\partial x}) + n^2 k_0^2 \vec{E}_t = 0 \tag{2.7}$$

. Using Poisson equation, and assuming $\frac{\partial n^2}{\partial x} \approx 0$,

$$\nabla \cdot (n^2 \vec{E}) = \nabla_t \cdot (n^2 \vec{E}_t) + n^2 \frac{\partial E_x}{\partial x} + E_z \frac{\partial n^2}{\partial x} = 0$$

$$\frac{\partial E_x}{\partial x} = -\frac{\nabla_t \cdot (n^2 \vec{E}_t)}{n^2} = -\frac{\nabla_t n^2}{n^2} \cdot \vec{E}_t - \nabla_t \cdot \vec{E}_t \tag{2.8}$$

Substituting Eq. 2.8 to Eq. 2.7 we get,

$$\frac{\partial^2 \vec{E}_t}{\partial x^2} + \nabla_t^2 \vec{E}_t + \nabla_t \left[\frac{\partial_t n^2}{n^2} \cdot \vec{E}_t \right] + n^2 k_0^2 \vec{E}_t = 0 \tag{2.9}$$

.

2.1 Derivation of BPM Equation

In a waveguide, rapid variation of the transverse fields is the phase variation due to propagation along the guiding axis. Assuming this guiding axis mainly along x-direction, one can define a SVE of the transverse $\vec{E}_t$ as follows,

$$\vec{E}_t = E_y\hat{y} + E_z\hat{z} = \Psi_y \exp(-in_0 k_0 x)\hat{y} + \Psi_z \exp(-in_0 k_0 x)\hat{z}. \tag{2.10}$$

The above equation can be used to calculate $\frac{\partial^2 \vec{E}_t}{\partial x^2}$ in Eq. 2.9.

$$\frac{\partial^2 E_y}{\partial x^2} = \left[\frac{\partial^2 \Psi_y}{\partial x^2} - 2in_0k_0\frac{\partial \Psi_y}{\partial x} - n_0^2k_0^2\Psi_y\right]\exp(-in_0k_0x) \tag{2.11}$$

SVE approximation is given below,

$$\left|\frac{\partial^2 \Psi_y}{\partial x^2}\right| \ll 2in_0k_0\frac{\partial \Psi_y}{\partial x}. \tag{2.12}$$

Similar set of equation can be written for calculating $\frac{\partial^2 E_z}{\partial x^2}$. Substituting SVE of $\vec{E}_t$ given by Eq. 2.10, $\frac{\partial^2 E_y}{\partial x^2}$ and expressions for $\frac{\partial^2 E_z}{\partial x^2}$ given by Eq. 2.11 in Eq. 2.9, we get

$$\begin{aligned} 2in_0k_0\frac{\partial \Psi_y}{\partial x} =& \frac{\partial^2 \Psi_y}{\partial y^2} + \frac{\partial^2 \Psi_y}{\partial z^2} + \frac{\partial}{\partial y}\left[\frac{1}{n^2}\frac{\partial (n^2)}{\partial y}\Psi_y + \frac{1}{n^2}\frac{\partial (n^2)}{\partial z}\Psi_z\right] \\ &+ k_0^2(n^2 - n_0^2)\Psi_y \end{aligned} \tag{2.13a}$$

$$\begin{aligned} 2in_0k_0\frac{\partial \Psi_z}{\partial x} =& \frac{\partial^2 \Psi_z}{\partial y^2} + \frac{\partial^2 \Psi_z}{\partial z^2} + \frac{\partial}{\partial z}\left[\frac{1}{n^2}\frac{\partial (n^2)}{\partial y}\Psi_y + \frac{1}{n^2}\frac{\partial (n^2)}{\partial z}\Psi_z\right] \\ &+ k_0^2(n^2 - n_0^2)\Psi_z \end{aligned} \tag{2.13b}$$

Eq. 2.13 form the equations used for propagating the mode along x direction.

2.2 Vectorial BPM Solver

When ϵ of the waveguide structure is strongly inhomogeneous, coupling terms in Eq. 2.13 between both Ψ_y and Ψ_z cannot be ignored. Eq. 2.13a and Eq. 2.13a need to be solved together for mode propagation. This is performed by Victoria's BPM solver. In this solver, once SVE of electric fields at $x = 0$ ($\Psi_y(0, y, z)$ and $\Psi_z(0, y, z)$) are set, the following equation is solved to obtain their values at all $x > 0$.

$$\frac{\partial}{\partial x}\begin{bmatrix}\Psi_y \\ \Psi_z\end{bmatrix} = \frac{-i}{2n_0k_0}\begin{bmatrix}\hat{P}_{yy} & \hat{P}_{yz} \\ \hat{P}_{zy} & \hat{P}_{zz}\end{bmatrix}\begin{bmatrix}\Psi_y \\ \Psi_z\end{bmatrix} \tag{2.14}$$

where each individual operator is given below,

$$\hat{P}_{yy} \equiv \frac{\partial}{\partial y}(\frac{1}{n^2}\frac{\partial}{\partial y}n^2) + \frac{\partial^2}{\partial z^2} + \frac{\omega^2}{c^2}(n^2 - n_0^2) \tag{2.15a}$$

$$\hat{P}_{yz} \equiv \frac{\partial}{\partial y}(\frac{1}{n^2}\frac{\partial}{\partial z}n^2) - \frac{\partial^2}{\partial y \partial z} \tag{2.15b}$$

$$\hat{P}_{zy} \equiv \frac{\partial}{\partial z}(\frac{1}{n^2}\frac{\partial}{\partial y}n^2) - \frac{\partial^2}{\partial z \partial y} \tag{2.15c}$$

$$\hat{P}_{zz} \equiv \frac{\partial^2}{\partial y^2} + \frac{\partial}{\partial z}(\frac{1}{n^2}\frac{\partial}{\partial z}n^2) + \frac{\omega^2}{c^2}(n^2 - n_0^2) \tag{2.15d}$$

2.3 Scalar BPM 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 $\frac{\nabla\epsilon}{\epsilon}$. If `Polarization` is set to Transverse Magnetic (TM), Ψ_z is the dominant field. If `Polarization` is set to Transverse Electric (TE), Ψ_y is dominant. If Ψ_y is dominant over Ψ_z, the mode equation can be written as,

$$\frac{\partial}{\partial x}\Psi_y = \frac{-i}{2n_0k_0}\hat{P}\Psi_y \tag{2.16}$$

where the operator $\hat{P}$ is given by,

$$\hat{P} = \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} + k_0^2(n^2 - n_0^2) \tag{2.17}$$

2.4 Solving BPM on Finite Difference Grid

In vectorial 3D BPM solver, the operator $\hat{P} \equiv \begin{bmatrix} \hat{P}_{yy} & \hat{P}_{yz} \\ \hat{P}_{zy} & \hat{P}_{zz} \end{bmatrix}$ in Eq. 2.14 has been discretized on a cubic grid at each YZ cross-section along x direction. In scalar 3D BPM solver or 2D BPM solver, the operator $\hat{P}$ in Eq. 2.16 has been discretized on the cubic or rectangular grid at each YZ cross-section. On discretization, $\hat{P}$ becomes a square matrix $\mathbf{P}$ and SVE of fields $\Psi_y(x)$ and $\Psi_z(x)$ become vector $\mathbf{\Psi}(x)$. Eqs. 2.14 and 2.16 are discretized along x direction.

$$\frac{\mathbf{\Psi}_m - \mathbf{\Psi}_{m-1}}{dx} = \frac{-i}{2n_0k_0}\left[\alpha\mathbf{P}_m \cdot \Psi_m + (1-\alpha)\mathbf{P}_{m-1} \cdot \Psi_{m-1}\right] \tag{2.18}$$

where $\mathbf{\Psi}_m$ and $\mathbf{\Psi}_{m-1}$ are discretized SVE at $x = x_i$ and $x = x_{i-1}$, respectively. The term on the RHS results from Crank-Nicholson discretization scheme, where $\alpha \in (0.5, 1.0]$ ensures stability of the scheme. To minimize discretization error, it is advisable to set α as close to 0.5 as possible. Simplifying the above equation gives,

$$\left[\mathbf{I} - \frac{i\alpha \cdot dx}{2n_0 k_0}\mathbf{P}_m\right] \cdot \mathbf{\Psi}_m = \left[\mathbf{I} + \frac{i(1-\alpha) \cdot dx}{2n_0 k_0}\mathbf{P}_{m-1}\right] \cdot \mathbf{\Psi}_{m-1} \quad (2.19)$$

Note, that RHS of the above equation is known since $\mathbf{\Psi}_{m-1}$ has already been calculated or set from the external source. At every $x = x_m$, Eq. 2.19 is solved to calculate $\mathbf{\Psi}_m$ from $\mathbf{P}_m$, $\mathbf{P}_{m-1}$, and $\mathbf{\Psi}_{m-1}$ at $x = x_{m-1}$.

Chapter 3

Configuration File Structure

Optosolver software reads various inputs from a BPM solver configuration file and calculate the SVE of the mode propagating along the waveguide. In BPM solver, the waveguide is always assumed to be traveling along X-axis. The mode is assumed to be in YZ plane. In 2D, the modal fields are assumed to be uniform along Z-axis. The BPM solver is executed by using the following command –

```
>> OptoSolver bpmsolver bpmSiWG_dev.cfg
```

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

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

```
Solver:
{
  Simulation = "BPM"; // Alternates: "Bi-BPM"
  Equation = "Scalar"; // Alternates: "Vectorial"
  Polarization = "TM"; // Alternates: "TE"
  Settings = [ "UsePowerMethod" ];

  Wavelength = 0.9;
  EffectiveIndex = 3.5;
  DecayConstant = 0.1;
  InterfacesX = [-0.5, 2.5];
  BidirectionalDecayFactor = 1E-2;
  BidirectionalTolerence = 1E-3;
  BidirectionalIterations = 10;
}

Source*left:
{
Type = "Mode";
CoordinateXCut = -4.5;
Intensity = 1000;
}

Source*mid:
{
Type = "Mode";
CoordinateXCut = -2.;
Intensity = 1000;
}

Boundary*ybdr:
{
  Axis = [ "Y", "Z"];
  Model = "PML";
  PMLLayers = 5.;
  sigmamax = 1.;
}
```

```
Plot:
{
   Quantities = ["AbsElectricField",
                 "AbsReverseElectricField",
                 "ElectricFieldZ"];
}
```

The above config file is composed of various sections which define the solver settings, types of sources, domain boundaries, the quantities to be plotted. In the config file, the string before '*' gives the section type, whereas the string after '*' specifies the name of the section. Various keywords in each of the section and their functionality is shortly 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_bpm.xdmf' and 'SiWG_bpm.h5'. Also, the mode solver used to calculate modes at various cross-sections in the device also stores the modal quantities in 'SiWG_Xloc_<xid>_mode.xdmf' and 'SiWG_Xloc_<xid>_mode.h5'. Here, <xid> represents index of X coordinate at which the mode in the YZ plane is calculated.

3.2 Solver Section

This section lists the information needed for the BPM solver apart from the device structure. This information is also passed on to the mode solver. For example, mode polarization is required, when the 2D "Scalar" laser equation is solved or 1D laser equation is solved. The keyword "Wavelength" provide wavelength (in μm) of propagating light in the waveguide. Initial effective index around which the modes are searched is given by the keyword "EffectiveIndex". "DecayConstant" sets imaginary part of the effective index. The rest of the quantities required for the mode calculation are set to their default values. Results of mode calculation, such as mode effective index and mode profile, are used for the calculation of SVE of electric field along the waveguide.

Various flags are provided as a list of comma-separated strings with the keyword "Settings". In the above file, the keyword "UsePowerMethod" is listed. Therefore, a faster 'power method' is used for mode calculation.

Exactly one 'Solver' section must be present in the config file.

3.3 Source Section

Multiple sources with different names can be instantiated in a config file.

In the given config file, two sources have been instantiated. They are named, 'left' and 'mid'. It specifies the type of the source (keyword 'Type'), location of the source along x-axis (keyword 'CoordinateXCut'), and its intensity (keyword 'Intensity').

Currently, only 'Mode' type source is available. When mode type source is specified, mode calculations are performed on YZ cross-section of the device at the x-coordinate set by 'CoordinateXCut'. These mode calculations yield electric field and magnetic flux normalized to integrated power of 1 Watt. It is scaled with the intensity specified with the keyword 'Intensity'. The scaled modal electric field is then added to the SVE of forward propagating electric field at the x-coordinate of the source.

3.4 Boundary Section

Multiple 'Boundary' sections with different names can be instantiated in a config file.

In the given file, a boundary section modeling 'PML' boundary type has been instantiated. This model is applicable along the planes at the boundary perpendicular to the axes given in the list with the keyword 'Axis'. In this case, since the keyword 'Y' is specified, the model is applicable at the layers in XZ plane at both maximum and minimum y boundaries. Separate boundary models can be applied at +Y or -Y axis by using the keywords 'Ymax' and 'Ymin', respectively. Boundaries perpendicular to Z axis are modeled in the same manner.

For 'PML' type boundary, the number of absorbing PML layers at the boundaries are specified with the keyword 'PMLLayers'. Also, absorption coefficient at the innermost layer is given with the keyword 'sigmamax'. Absorption coefficient decays polynomially towards the outermost layer.

Separate boundary sections must be used for instantiating different boundary models. Following boundary models are recognized by the BPM solver – 1. Perfectly Matching Layer (PML) 2. Absorbing Boundary Condition (ABC) 3. Reflecting Boundary Condition (RBC) 4. Periodic Boundary Condition (RBC). By default, boundaries along non-propagating axes are assumed to be reflecting. Boundary along the propagating axis 'X' is specially treated as described in the theory.

3.5 Plot Section

Exactly one plot section must be instantiated per config file. The keyword 'Quantities' lists the quantities that must be saved in an hdf5 file as a list of string. SVE of these quantities are calculated in the BPM solver by propagating the mode along X axis.

3.6 Running Mode Calculation

In this section, the above config file is used to perform mode calculations on a lateral 2D cross section of a waveguide shown in Fig. 3.1. The waveguide is aligned along the lateral direction (X-axis). It is invariant

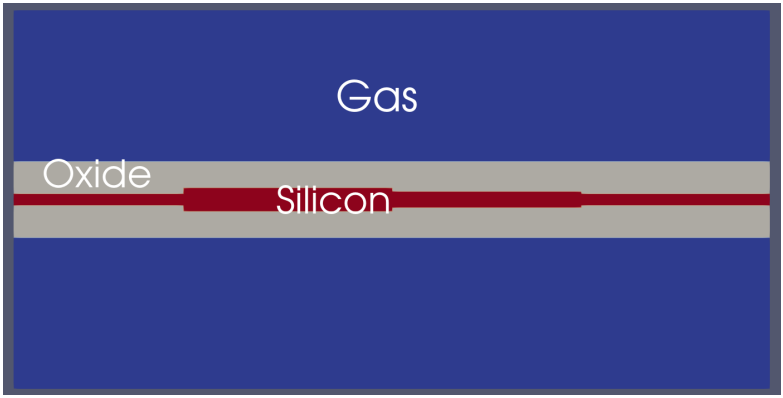


Figure 3.1: Structure of the lateral cross-section of the simulated waveguide along X axis. The waveguide is invariant in Z direction normal to the plane of the figure.

along Z dimension (normal to the plane of the figure). The waveguide structure can be created using the command

```
>> OptoSolver str bpmSiWG_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 BPM solver. The structure config file must also be present in the same folder. Once the config file is set, the BPM calculations can be performed using the following command

```
>> OptoSolver bpmsolver bpmSiWG_dev.cfg
```

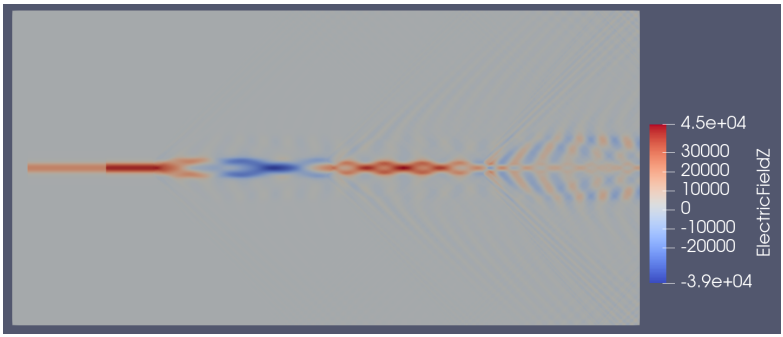
The BPM calculations will generate and xdmf file (extension *.xdmf) together with an hdf5 file (extension *.h5).

3.7 Output Files

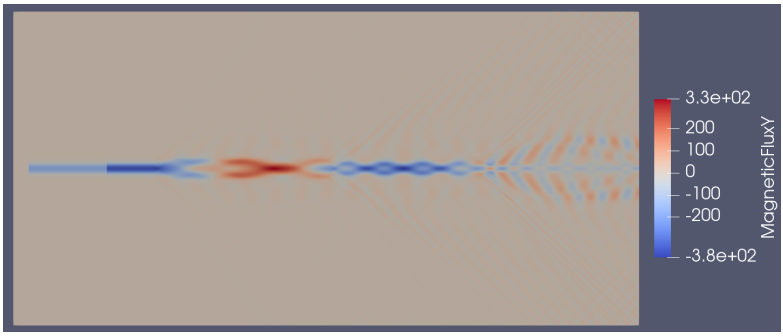
The keyword `Out` in the `File` section of the config file sets the prefix to the output file name. In this case, the output files are called 'SiWG_bpm.xdmf' and 'SiWG_bpm.h5'. Note, that the xdmf file is simply an XML script which provides additional information on various datasets stored in the hdf5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

```
>> paraview SiWG_bpm.xdmf
```

In paraview, various quantities listed in the `Plot` section such as, X, Y, and Z components of SVE of electric field and magnetic flux along the waveguide can be selected for visualization. Fig. 3.2 plots electric field in Z-direction and magnetic flux in Y-direction. The horizontal axis is X-axis and the vertical axis is Y-axis. Note the increased electric field at $x = -4.5\mu m$ and $-2.0\mu m$ due to the modal sources present at those locations. Also, scattering of light from the waveguide into oxide cladding is observed as the mode propagates from a thicker Silicon waveguide into a thinner one.



(a) Electric Field



(b) Magnetic Flux

Figure 3.2: Calculated SVE of electric field along Z direction and magnetic flux along Y direction of the waveguide in Fig. 3.1 are shown in the figures above. Note, that the fields are invariant in Z direction normal to the plane of the figure.

Chapter 4

Configuring BPM Solver

An example configuration file provided in Chapter 3 lists typical configurations of BPM solver. In BPM solver, the waveguide is always assumed to be propagating the beam along X-direction. The beam is generated by the uni-modal source specified by its X-location. Mode Solver is used internally to calculate modal electric fields and magnetic fluxes along the waveguide cross-section at X-location of the source.

In the chapter, a list of all the available configurations in the BPM solver and their usage information is provided. For clarity, the keywords are listed section-wise. The mandatory keywords are marked 'mandatory'. Optional keywords are provided with the default input values.

4.1 Keywords in File Section

Following keywords must be listed in `File` section of BPM solver config file.

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

4.2 Keywords in Solver Section

Following keywords may be listed in `Solver` section of BPM solver config file. The BPM Solver is configured using these keywords.

- `Wavelength:` (Mandatory) Wavelength of all the sources is specified in the units of μm.

- `EffectiveIndex:` (Mandatory) Effective refractive index of the waveguide around which the waveguide modes are searched at the source locations and the X-locations listed in `InterfacesX`. Effective index of the propagating fields at any location in the waveguide is set to the mode effective index of the nearest source or interface which comes before the given location.

- `DecayConstant:` (Mandatory) 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. Complex effective index causes decay of field amplitudes along propagation direction.

- `Polarization:` If a 2D device is simulated or if a 3D device is simulated with the scalar mode equation, polarization of the mode must be specified. Polarization can be 'TE' or 'TM', which stand for 'Transverse Electric' or 'Transverse Magnetic', respectively. Default value is 'TE'.

- `Equation:` If a 3D device is simulated, mode calculations are performed along YZ-plane of the waveguide. In this case, either 'Scalar' or 'Vectorial' mode equation can be solved for mode calculations as well as propagation using BPM. Equation can be either 'Scalar' or 'Vectorial'. Default value is 'Scalar'.

- `PowerMethodTol:` Applicable only when power method is used for mode calculations along waveguide cross-section. 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.

- `Settings:` Various flags are set by providing a list of appropriate keywords as comma-separated strings as follows.

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

- `Simulation:` Specifies the type of BPM simulation to be performed. Following types of BPM simulations can be performed.

  1. `BPM`: Performs forward propagation of the waveguide.
  2. `Bi-BPM`

  .

- `BidirectionalDecayFactor`: In Bidirectional BPM calculations, reverse electric field is multiplied with `BidirectionalDecayFactor` and propagated backward. Default value is 10^{-2}.

- `BidirectionalTolerence` In Bidirectional BPM calculations, forward and reverse propagation are iteratively performed. When the ratio of reverse electric field to the forward electric field at the end of forward propagation is less than `BidirectionalTolerence`, the iterations end. The iterations also end, if the ratio does not change between successive iterations. Default value is 10^{-3}.

- `BidirectionalIterations`: Specifies maximum number of iterations of forward and reverse propagation in Bidirectional BPM. Default value is 10.

- `InterfacesX`: Specifies a list of x-coordinates at which the waveguide discontinuities or interfaces between two different waveguides exist. The x-coordinates are listed in the following format.

```
InterfacesX = [-0.5, 5.0, 10.];
```

Note, that the Mode solver, which is internally used for mode calculations at the source locations, is also configured from the relevant keywords listed in the `Solver` section of BPM solver config file.

4.2.1 Available Settings in BPM solver

Keywords which may be listed in the field `Settings` 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. `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.

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

4.3 Keywords in Source Section

In the config file, the source section is instantiated by the name 'Source*<name>', where <name> is the name of the source. Multiple sources with different names can be instantiated in a config file. Following keywords can be defined in `Source` section.

- `Type`: Currently only the source of the type 'Mode' is supported. The 'Mode' type source performs Mode calculations to determine electric field distribution in the cross section. Default value is 'Mode'.

- `CoordinateXCut`: (Mandatory) Specifies x-coordinate at which the source is present.
- `Intensity`: (Mandatory) Specifies intensity of the mode source in the units of 'W/ţmm^2.

4.4 Keywords in Boundary Section

In the config file, a Boundary section is instantiated by the name 'Boundary*<name>', where <name> represents the boundary name. Multiple boundary definitions with different names can be instantiated in a config file. Following keywords can be defined in `Boundary` section.

- `Axis`: Current boundary definition is applied to the list of boundaries specified by `Axis`. Available sets of boundary names are –
  1. `Ymin`: XZ boundary at the smallest y-coordinate.
  2. `Ymax`: XZ boundary at the largest y-coordinate.
  3. `Y`: XZ boundary at both the smallest and largest y-coordinates.
  4. `Zmin`: XY boundary at the smallest z-coordinate.
  5. `Zmax`: XY boundary at the largest z-coordinate.
  6. `Z` XY boundary at both the smallest and largest z-coordinates.

  The `Axis` are listed in the following format. `Axis = ["Ymin", "Ymax", "Z"];`
- `Model` Boundary type to be applied to the current boundary condition. Following boundary models are recognized by the BPM solver – 1. PML 2. ABC 3. RBC 4. RBC. Separate boundary sections must be used for instantiating different boundary models. By default, boundaries along non-propagating axes are assumed to be reflecting. Boundary along the propagating axis 'X' is specially treated as described in the theory.
- `PMLLayers` Number of PML layers at each boundary. Note, that no user-defined object (excluding the 'Gas' box which defines simulation domain) must intersect these boundary layers. Only applicable for PML boundary model.

- `sigmamax` Value of absorption coefficient in consecutive PML layers increases polynomially with their distance from the innermost layer. `sigmamax` specifies maximum value of the the absorption coefficient at the outermost PML layer.

- `ABCLayers` Number of ABC layers at each boundary. Note, that no user-defined object (excluding the 'Gas' box which defines simulation domain) must intersect these boundary layers. Only applicable for ABC boundary model.

- `kappamax` Imaginary dielectric constant in consecutive ABC layers increases polynomially with their distance from the innermost layer. `kappamax` specifies maximum value of imaginary dielectric constant at the outermost PML layer. Note, that usage of the parameter `sigmamax` in PML and `kappamax` in ABC is different as explained in the theory.

4.5 Plot Section

A Plot section can be specified by using the keyword `Plot: {...}`. The keyword `Quantities` in Plot section lists the quantities in a comma separated list which are to be saved at the end of the simulation. Exactly one plot section must be instantiated per config file. An example `Plot` section is shown below.

```
Plot:
{
   Quantities = ["AbsElectricField",
                 "AbsReverseElectricField",
                 "ElectricFieldZ"];
}
```

A list of all the quantities which can be saved and plotted is given in 6, Section 6.1, together with their description. The listed quantities are stored in the hdf5 file 'SiWG_bpm.h5'. They can be viewed in the visualizer program 'Paraview' using the script file 'SiWG_bpm.xdmf'.

4.6 Miscellaneous Comments

4.6.1 Coordinate Transformation in Mode Solver

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.

4.6.2 Selecting Numeric Method For Mode Solver

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

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 Finite Difference Time Domain (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 BPM solver.

> **Note:** Whenever possible, please use config file to setup the BPM 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 commands given below are called on the `bpmsolver` object. They setup the solver, e.g. add sources, specify boundary conditions, etc.

Note: While modifying the solver object, it is strongly recommended to use the following commands in the same order in which they are listed below. For example, set global parameters, then set boundary conditions, and then add sources.

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 of *all* the mode sources.
- `EffectiveIndex` : Approximate effective index of the waveguide. Exact effective index is calculated by mode calculations.
- `DecayConstant` : Approximate imaginary part of the effective index.
- `BidirectionalDecayFactor` : Decay factor for bi-directional BPM simulations.
- `BidirectionalTolerence` : Tolerance at which the iterations of the bidirectional BPM simulations stop.
- `BidirectionalIterations` : Maximum number of iterations of the bidirectional BPM simulations.

2. `StringParams` : A python dictionary mapping parameter name to a string. The following string parameters can be supplied.
   - `Equation` : Possible alternatives - 'Scalar', or 'Vectorial'.
   - `Simulation` : Possible alternatives - 'Bi-BPM' for bi-directional BPM, 'BPM' for standard BPM.
   - `Polarization` : Possible alternatives - TM or TE.
3. `NumericListParams` : Currently unused.
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 as a list 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.

5.3.2 setDomainBoundary

The command `setDomainBoundary(...)` takes the following arguments -

1. `Model`: type of domain boundary to be set. Possible alternatives are - Convolutional Perfectly Matching Layer (CPML), RBC, and RBC.

2. `Axes`: axes at which the above specified boundary is to be set are provided as a list of strings. Possible alternatives are - "Y", "Z".

3. `NumericParams`: A python dictionary mapping parameter name to its numeric value . The following numeric parameters can be supplied.

   - PMLLayers: number of PML layers at each boundary face.
   - sigmamax: maximum σ parameter of the "CPML" model.

4. `Flags`: An empty list. This parameter is currently unused.

5.3.3 resetDomainBoundaryToReflective

`resetDomainBoundaryToReflective()` command resets all the domain boundaries to "reflective" domain boundaries (RBC).

5.3.4 addSource

The command `addSource(...)` takes the following arguments-

1. `Name` : Name of the source

2. `NumericParams` : A python dictionary mapping parameter name to its numeric value . The following numeric parameters can be supplied.

   - Intensity: source intensity (W/m^2)
   - CoordinateXCut: location of the source-plane. Only X-coordinate is supplied.

3. `StringParams`: A python dictionaly mapping parameter name to its string value. The following string parameters can be supplied.

   - Type: Currently only "Mode" source is available.

4. `NumericListParams`: A python dictionary mapping parameter name to a list of numeric values. The following string parameters can be supplied.

    - minYZPlane: If the mode-source is confined in YZ-plane, then y- and z- coordinates of minimum of the source window are specified.
    - maxYZPlane: If the mode-source is confined in YZ-plane, then y- and z- coordinates of maximum of the source window are specified.

5. `StringListParams`: Currently unused.

5.3.5 removeSource

The command `removeSource(...)` takes name of the source as an input parameter and deletes the source from the solver.

5.4 Solve BPM System

The following commands begin the simulations.

5.4.1 portToGPU

The command `portToGPU(...)` ports the simulations to the first available GPU (with GPU id = 0). If you wish to port to another GPU, please give the GPU id as an argument. This command must be run before solving the system on the GPU.

5.4.2 solveSystem

The command `solveSystem(...)` begins BPM simulations. If `portToGPU` is run before, then the BPM simulator uses GPU for matrix inversion operations. If `portToGPU` is not run before, then the CPU is used for all the computation.

5.5 Retrieve data

Once the simulation is finished, the following commands help in accessing the data stored by the BPM solver.

5.5.1 getSavedQuantitiesNames

The command `getSavedQuantitiesNames(...)` returns the names of all the datasets stored by the BPM solver. The available dataset names are listed in Chapter 6.

5.5.2 getAllSavedQuantitiesData

The command `getAllSavedQuantitiesData(...)` returns the stored dataset values at all the vertices vertex as a 2D "NumPy" array. Leading dimension has number of dataset entries and trailing dimension stores dataset values at each vertex for each of the datasets.

5.5.3 getSavedQuantityData

The command `getSavedQuantityData(...)` returns the stored dataset value of the user-specified dataset at all the vertices vertex as a "NumPy" array. User must specify name of the dataset as an input argument. Leading dimension is one and trailing dimension stores dataset values at each vertex for each of the datasets.

Chapter 6

Visualization of Results

Real and imaginary parts of X-, Y-, and Z- components of electric field and magnetic flux are calculated on the 2D/3D grid of the waveguide. They are stored in an hdf5 file. Additionally, a xdmf script file is written. The output files are named '<out>_bpm.xdmf' and '<out>_bpm.h5', where '<out>' is the string input by user with the keyword `Out` in `File` section of mode solver config file.

Effective indexes corresponding to the active mode at a specific x-location are stored in '<out>_Xloc_<xidx>_mode.csv' file. Here, <xidx> is the index of x-coordinate of cross-section of the waveguide at which the mode is calculated. They can be viewed using a text editor or a csv file viewer program. Similarly, the mode profiles corresponding to the mode at a specific x-location are stored in '<out>_Xloc_<xidx>_mode.h5' file. The corresponding script file is named '<out>_Xloc_<xidx>_mode.xdmf'. It can be viewed using paraview.

6.1 Quantities

Following quantities can be saved at the end of the simulation.

- `ElectricField[X | Y | Z]`: Real part of x-, y-, or z- component of SVE of electric field.

- `ImElectricField[X | Y | Z]`: Imaginary part of x-, y-, or z-component of SVE of electric field.
- `AbsElectricField`: Magnitude of SVE of electric field.
- `MagneticFlux[X | Y | Z]`: Real part of x-, y-, or z- component of SVE of magnetic flux.
- `ImMagneticFlux[X | Y | Z]`: Imaginary part of x-, y-, or z-component of SVE of magnetic flux.
- `AbsMagneticFlux`: Magnitude of SVE of magnetic flux.
- `ReverseElectricField[X | Y | Z]`: Real part of x-, y-, or z-component of SVE of reverse electric field. Only relevant when bi-directional BPM simulation is performed.
- `ImReverseElectricField[X | Y | Z]`: Imaginary part of x-, y-, or z- component of SVE of reverse electric field. Only relevant when bi-directional BPM simulation is performed.
- `AbsReverseElectricField`: Magnitude of SVE of reverse electric field.

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_bpm.xdmf
```

Appendix A

Notation and Acronyms

Acronyms

ABC	Absorbing Boundary Condition
BPM	Beam Propagation Method
CPML	Convolutional Perfectly Matching Layer
FDTD	Finite Difference Time Domain
RBC	Periodic Boundary Condition
PML	Perfectly Matching Layer
RBC	Reflecting Boundary Condition
RHS	Right Hand Side
SVE	Slowly Varying Envelope

TE	Transverse Electric
TM	Transverse Magnetic

Bibliography