

Sentaurus™ Device Electromagnetic Wave Solver User Guide

Version T-2022.03, March 2022



Copyright and Proprietary Information Notice

© 2022 Synopsys, Inc. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

Conventions	8
Customer Support	8
<hr/>	
1. Getting Started	10
Starting EMW.	10
Input and Output File Types.	11
Command File (*_eml.cmd)	11
Tensor-Grid File (*_msh.tdr)	12
Parameter File (*_eml.par)	13
Plot, Extractor, and Save Files (*_eml.tdr)	13
Result File (*_eml.plt)	14
Log Files (*_eml.log, *_eml.out)	14
File Compression	14
Accessing Help on Syntax	14
TCAD Sentaurus Tutorial: Simulation Projects	14
<hr/>	
2. Basic Theory of Finite-Difference Time-Domain Method	16
Basic Theory of FDTD	16
Discretized Maxwell's Equations	16
Accuracy of the Finite Difference Stencil	20
Stability (Courant) Criteria	20
Modeling of Dispersive Media	21
Physical Model	22
Debye Relaxation	22
Lorentzian Resonance	23
Unmagnetized Plasma	23
Drude–Lorentz	24
Modified Lorentz	24
Drude–Modified Lorentz	24
References.	25

3. Boundary Conditions	26
Introduction	26
PEC and PMC Boundary Conditions	26
Periodic Boundary Conditions	27
Absorbing Boundary Conditions	27
Specifying Parameters of CPML Boundary Condition	28
Combining CPML With Other Boundary Conditions	30
References	31
4. Building the Correct Mesh	32
Generating Tensor Meshes for EMW	32
Using the Sentaurus Mesh Tensor-Product Generator	32
Specifying Nodes per Wavelength in Meshing	33
Meshing Examples	35
Example 1: Planar Two-Layer Silicon–Air	35
Example 2: Image Sensor	36
Inspecting and Troubleshooting Tensor-Grid Generation	38
5. Specifying Material Parameters	39
Introduction	39
Complex Refractive Index Model	39
Complex Refractive Index Model Interface	40
Spatially Dependent Complex Refractive Index Model	41
Spatially Dependent Dispersive Media	42
Limitations	44
Dispersive Model Parameters	44
Dispersive Model Command Syntax	44
Specifying User-Defined Dispersive Model Poles in Parameter File	46
Explicit Specification of EMW Material Parameters	49
PEC and PMC Materials	50
Precedence of Material Specification	50

6. Specifying Excitations.	52
Overview	52
Total-Field Scattered-Field	52
Specification of the Illumination Direction	53
Plane Waves	54
Specifying Excitation Regions or Boxes.	54
Specifying Direction and Polarization.	55
Specifying Elliptical Polarization.	57
Spatially Incoherent Illumination	58
Gaussian Beam or Gaussian Field Distribution	59
Temporal Dependency	61
CODE V Interface	63
References.	65
7. Terminating Simulations	66
Options for Terminating FDTD Time-Stepping Loop	66
Detectors	67
8. Extracting and Visualizing Results	70
Overview	70
Plots	71
Monitors	74
Extractors	75
Sensors	77
Reflection, Transmission, and Absorption Sensors	78
Using Multiple Domains in a Sensor	80
Nonaxis-Aligned Sensor.	81
Far Field	82
Specifying the Far-Field Computation	84
Visualizing Results of the Far-Field Computation	87
Postprocessing of Extractors, Sensors, and Far Field.	89
Discrete Fourier Transform	90

Contents

On-the-Fly DFT	91
Defining the DFT Wavelength Table.	91
Normalizing Field Values	93
Downsampling of DFT	93
Limitation of Single-Frequency Periodic Boundary Condition	94
EMW Syntax	95
References.	96
<hr/>	
9. Parallelization	97
Overview of Parallelization.	97
Shared-Memory Parallelization	97
Parallelization Using the Message Passing Interface	98
Domain Decomposition	98
Installing and Running EMW MPI on DP Systems.	99
Choosing Between Shared-Memory Parallelization and Distributed Processing	99
Hardware Acceleration Using Graphics Processing Units	100
Installing EMW GPU	100
Hardware and Software Setup	101
Optimization Settings	101
Running EMW GPU	102
General Considerations and Recommendations	103
Performance.	104
Monitoring.	104
Limitations	105
Licensing for Different Parallel Modes	105
SMP Parallel Mode	105
DP Parallel Mode	106
Mixed DP and SMP Parallel Mode.	106
Hardware-Accelerated Parallel Mode	106
Controlling Parallel Licenses	106
References.	107
<hr/>	
A. Commands	108
Boundary	108
CodeVExcitation	110
ComplexRefractiveIndex	111

Contents

Detector	113
DFT	114
DispersiveMedia	115
Extractor.	116
Farfield.	118
GaussianBeamExcitation.	121
Globals.	124
Monitor.	126
PECMedia	128
PlaneWaveExcitation.	129
Plot.	131
PMCMedia	136
RTA	137
Save.	138
Sensor	140
TruncatedPlaneWaveExcitation	144
<hr/>	
B. Troubleshooting Simulations	147
Unexpected Results.	147
Did the simulation converge?	147
Is the excitation correct?	147
Are the material parameters correct?	148
Simulations Run Nonstop	148
Did the simulation converge?	148
Is the value of the time step reasonable?	148
Did the maximum deviation first decrease but then increase again?	148
Simulator Terminates	149
<hr/>	
Glossary	150

About This Guide

The Synopsys Sentaurus™ Device Electromagnetic Wave Solver (EMW) is a simulation module for the numeric analysis of electromagnetic waves. EMW is a full-wave time-domain simulator based on the finite-difference time-domain (FDTD) method. It is typically used to calculate the optical carrier generation in optoelectronic devices.

EMW can compute full-wave time-domain solutions of Maxwell equations, thereby covering the entire range of electromagnetic problems from high frequency to optical applications.

For additional information, see:

- The TCAD Sentaurus release notes, available on the Synopsys SolvNetPlus support site (see [Accessing SolvNetPlus on page 9](#))
- Documentation available on the SolvNetPlus support site

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Bold text	Identifies a selectable icon, button, menu, or tab. It also indicates the name of a field or an option.
<code>Courier font</code>	Identifies text that is displayed on the screen or that the user must type. It identifies the names of files, directories, paths, parameters, keywords, and variables.
<i>Italicized text</i>	Used for emphasis, the titles of books and journals, and non-English words. It also identifies components of an equation or a formula, a placeholder, or an identifier.

Customer Support

Customer support is available through the Synopsys SolvNetPlus support site and by contacting the Synopsys support center.

Accessing SolvNetPlus

The SolvNetPlus support site includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The site also gives you access to a wide range of Synopsys online services, which include downloading software, viewing documentation, and entering a call to the Support Center.

To access the SolvNetPlus site:

1. Go to <https://solvetplus.synopsys.com>.
2. Enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register.)

Contacting Synopsys Support

If you have problems, questions, or suggestions, you can contact Synopsys support in the following ways:

- Go to the Synopsys [Global Support Centers](#) site on www.synopsys.com. There you can find email addresses and telephone numbers for Synopsys support centers throughout the world.
- Go to either the Synopsys SolvNetPlus site or the Synopsys Global Support Centers site and open a case (Synopsys user name and password required).

Contacting Your Local TCAD Support Team Directly

Send an email message to:

- support-tcad-us@synopsys.com from within North America and South America
- support-tcad-eu@synopsys.com from within Europe
- support-tcad-ap@synopsys.com from within Asia Pacific (China, Taiwan, Singapore, Malaysia, India, Australia)
- support-tcad-kr@synopsys.com from Korea
- support-tcad-jp@synopsys.com from Japan

1

Getting Started

This chapter describes the Sentaurus Device Electromagnetic Wave Solver application, how to set up simulations, and the general syntax framework.

Starting EMW

Sentaurus Device Electromagnetic Wave Solver (EMW) is an electromagnetic solver based on the finite-difference time-domain (FDTD) method. It can numerically solve the temporal evolution of electromagnetic waves in a device structure defined on a tensor grid. The main result is the absorbed photon density distribution, which is typically used in subsequent electrical simulations.

To start EMW from the command line, use:

```
> emw [command_file_name]
```

where the optional `command_file_name` is a valid command file of EMW. For example:

```
> emw ppl_eml.cmd
```

To list the command-line options, use:

```
> emw -h
```

To list all of the installed EMW releases, use:

```
> emw -releases
```

To check the default version number, use:

```
> emw -v
```

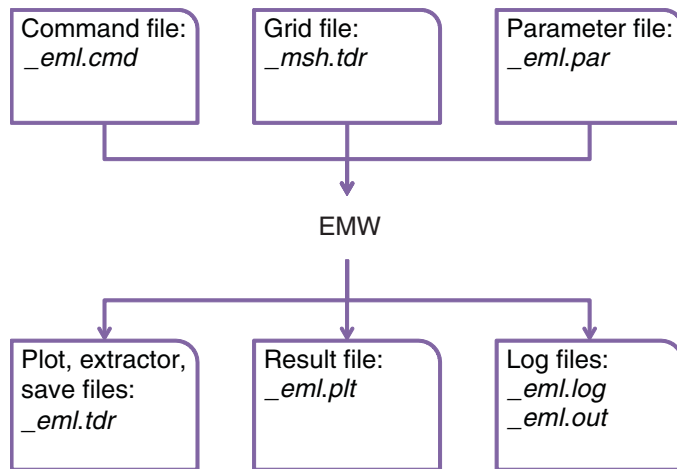
To run a particular release version, use:

```
> emw -rel S-2021.06
```

Input and Output File Types

The main input and output file types used in EMW simulations are shown in [Figure 1](#). The file names can be specified in the EMW command file.

Figure 1 Overview of input and output files for EMW



Command File (*_eml.cmd)

The command file is the main input file for EMW. It contains all the model settings and file specifications, and can be edited. This file is referred to as the *command file* or *input file*.

An EMW command file consists of several command sections of the form `Section {...}`, with each section describing a specific aspect of the simulation. Inside a section, each command keyword has the form `keyword = value`. For example:

```
* Plot the absolute value of the electric field at the end of the
* simulation
Plot {
    Name      = "n@node@_Eabs"
    Quantity  = {AbsElectricField, AbsMagneticField, Region}
    TickStep  = 300
    StartTick = 100
    EndTick   = 800
    FinalPlot = yes
}
```

Chapter 1: Getting Started

Input and Output File Types

Note:

Regarding the command file:

- Keywords are case insensitive, but identifiers such as region names and materials are surrounded by double quotation marks and are case sensitive.
- Lists are defined in braces with their items separated by a comma.
- Each command must be on a single line; only list items can be wrapped before or after the comma.
- Comments are indicated by the first character on a line being # or *.

Typically, an EMW command file consists of the following sections:

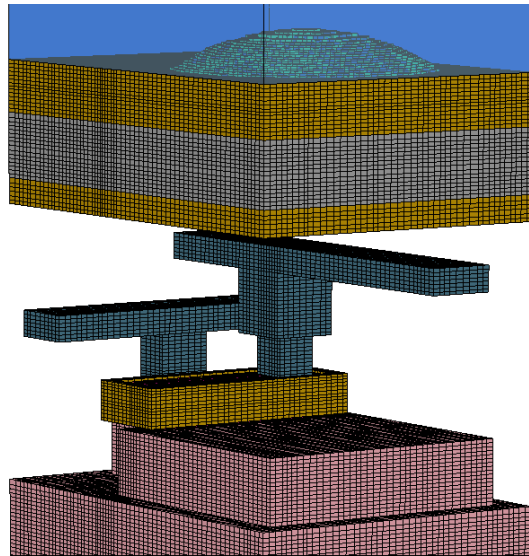
- Globals
- ComplexRefractiveIndex
- DispersiveMedia
- Boundary
- PlaneWaveExcitation
- Plot
- Extractor
- Sensor
- Monitor
- Detector

In general, the order of the sections in the command file is arbitrary. However, it is recommended that you maintain the above order because it simplifies navigation looking for a particular parameter.

Tensor-Grid File (*_msh.tdr)

EMW needs a tensor grid to run FDTD simulations. A typical example of a tensor grid is shown in [Figure 2](#). To generate a suitable tensor-grid file with Sentaurus Mesh, see [Generating Tensor Meshes for EMW on page 32](#) and the *Sentaurus™ Mesh User Guide*.

Figure 2 Example of a tensor grid of a CMOS image sensor; oxide removed for visualization



Parameter File (*_eml.par)

EMW allows you to define the optical parameters as a complex refractive index (CRI) in the material parameter file. It uses the same CRI library as Sentaurus Device. Therefore, the same parameter file as for Sentaurus Device can be used (see the *Sentaurus™ Device User Guide*).

The optical parameters are then read from the corresponding `ComplexRefractiveIndex` section. For example:

```
Material = "Silicon" {  
  ComplexRefractiveIndex {  
    Formula = 1  
    NumericalTable (  
      0.1908  0.84  2.73;  
      ...  
    )  
  }  
}
```

Plot, Extractor, and Save Files (*_eml.tdr)

As multiple `Plot`, `Extractor`, and `Save` sections are allowed, the file names are defined in the corresponding sections. Extracting useful simulation results is described in more detail in [Chapter 8 on page 70](#).

Result File (*_eml.plt)

A result file is used when global values have to be stored, such as the integrated absorbed photon density in a certain volume or the photon flux through a surface. The results can be viewed using Inspect or Sentaurus Visual. For discrete Fourier transform (DFT) simulations, use Sentaurus Visual only.

Log Files (*_eml.log, *_eml.out)

EMW generates the log file during a simulation run. It contains information about input parameters, and the models and values of physical parameters used in the simulation. The log file contains more details than the brief information written to the standard output (.out) during the simulation.

File Compression

You can control further compression, beyond a basic standard compression for plot, sensor, extractor, and save files as well as the result file, in the `Globals` section by specifying `CompressTDR=Yes` | `No`.

Accessing Help on Syntax

Help for the syntax of each keyword can be obtained by using the EMW binary with the `-P` command-line option. For example, to view all of the allowed options for the keyword `Quantity` in the `Plot` section, in a shell, type in a shell prompt:

```
> emw -P:plot:quantity
*****
*****                      SentaurusDeviceEMW                      *****
...
Quantity = { <identifier>, ... }
  Default value: { AbsElectricField }
  Exclusive options are: AbsElectricDisplacement
                        AbsElectricField
...
```


TCAD Sentaurus Tutorial: Simulation Projects

The TCAD Sentaurus Tutorial provides various projects demonstrating the capabilities of Sentaurus Device Electromagnetic Wave Solver.

Chapter 1: Getting Started

TCAD Sentaurus Tutorial: Simulation Projects

To access the TCAD Sentaurus Tutorial:

1. Open Sentaurus Workbench by entering the following on the command line: `swb`
2. From the menu bar of Sentaurus Workbench, choose **Help > Training** or click  on the toolbar.

Alternatively, to access the TCAD Sentaurus Tutorial:

1. Go to the `$STROOT/tcad/current/Sentaurus_Training` directory.

The `STROOT` environment variable indicates where the Synopsys TCAD distribution has been installed.

2. Open the `index.html` file in your browser.

2

Basic Theory of Finite-Difference Time-Domain Method

This chapter describes the basic theory of the finite-difference time-domain (FDTD) method, including the total-field scattered-field methodology and the dispersive material models.

Basic Theory of FDTD

EMW provides a FDTD kernel that specifically targets 3D periodic structures for which light is incident at an oblique angle. Typically, periodic boundary conditions must be imposed in the lateral directions; whereas, absorbing boundary conditions are applied in the normal direction. The implementation of the kernel is based on the sine–cosine formulation [1][2] and can be applied to both coherent and incoherent excitations. Prominent application areas for the FDTD kernel with 3D oblique periodic boundary conditions are CMOS image sensors, solar cells, and photonic bandgap structures.

Note:

The sine–cosine formulation is based on two sets of field components, one with sine and one with cosine time dependency, which are advanced simultaneously in time. Therefore, the memory footprint of this method is, at least, doubled compared to the conventional FDTD algorithm.

Two-dimensional simulations using the 3D FDTD kernel with oblique periodic boundary conditions are performed by extruding the 2D structure by one layer in the third dimension.

Discretized Maxwell's Equations

Maxwell's curl equations in linear, isotropic, nondispersive material are [1]:

$$\begin{aligned}\frac{\partial \mathbf{E}}{\partial t} &= \frac{1}{\epsilon} \nabla \times \mathbf{H} - \frac{1}{\epsilon} (\mathbf{J} + \sigma \mathbf{E}) \\ \frac{\partial \mathbf{H}}{\partial t} &= -\frac{1}{\mu} \nabla \times \mathbf{E} - \frac{1}{\mu} (\mathbf{M} + \sigma^* \mathbf{H})\end{aligned}\tag{1}$$

Chapter 2: Basic Theory of Finite-Difference Time-Domain Method

Basic Theory of FDTD

where:

- **E** is the vectorial electric field in V/m.
- **H** is the vectorial magnetic field in A/m.
- **J** is the electric current density in A/m².
- **M** is the magnetic current density in V/m².
- ϵ is the electrical permittivity in F/m.
- μ is the magnetic permeability in H/m.
- σ is the electric conductivity in S/m.
- σ^* is the magnetic conductivity in Ω /m.

In terms of x-, y-, and z-components, the curl equations can be decomposed into a system of six coupled scalar equations:

$$\begin{aligned}\frac{\partial E_x}{\partial t} &= \frac{1}{\epsilon} \left[\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} - (J_x + \sigma E_x) \right] \\ \frac{\partial E_y}{\partial t} &= \frac{1}{\epsilon} \left[\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} - (J_y + \sigma E_y) \right] \\ \frac{\partial E_z}{\partial t} &= \frac{1}{\epsilon} \left[\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - (J_z + \sigma E_z) \right] \\ \frac{\partial H_x}{\partial t} &= \frac{1}{\mu} \left[\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} - (M_x + \sigma^* H_x) \right] \\ \frac{\partial H_y}{\partial t} &= \frac{1}{\mu} \left[\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z} - (M_y + \sigma^* H_y) \right] \\ \frac{\partial H_z}{\partial t} &= \frac{1}{\mu} \left[\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} - (M_z + \sigma^* H_z) \right]\end{aligned}\tag{2}$$

Maxwell's curl equations can be discretized by the central difference method to achieve second-order accuracy.

Chapter 2: Basic Theory of Finite-Difference Time-Domain Method

Basic Theory of FDTD

For nonuniform grids with $\Delta x \neq \Delta y \neq \Delta z$, the discretized x-, y-, and z-components of the E-fields for the point (i, j, k) are:

$$\begin{aligned}
 E_x \Big|_{i+\frac{1}{2}, j, k}^{n+1} = & C_{a, E_x} \Big|_{i+\frac{1}{2}, j, k} E_x \Big|_{i+\frac{1}{2}, j, k}^n \\
 & + C_{b, E_x} \Big|_{i+\frac{1}{2}, j, k} \left[\left(H_z \Big|_{i+\frac{1}{2}, j+\frac{1}{2}, k}^{n+\frac{1}{2}} - H_z \Big|_{i+\frac{1}{2}, j-\frac{1}{2}, k}^{n+\frac{1}{2}} \right) / \Delta y \right. \\
 & - \left(H_y \Big|_{i+\frac{1}{2}, j, k+\frac{1}{2}}^{n+\frac{1}{2}} - H_y \Big|_{i+\frac{1}{2}, j, k-\frac{1}{2}}^{n+\frac{1}{2}} \right) / \Delta z \\
 & \left. - J_x \Big|_{i+\frac{1}{2}, j, k}^{n+\frac{1}{2}} \right]
 \end{aligned} \tag{3}$$

$$\begin{aligned}
 E_y \Big|_{i, j+\frac{1}{2}, k}^{n+1} = & C_{a, E_y} \Big|_{i, j+\frac{1}{2}, k} E_y \Big|_{i, j+\frac{1}{2}, k}^n \\
 & + C_{b, E_y} \Big|_{i, j+\frac{1}{2}, k} \left[\left(H_x \Big|_{i, j+\frac{1}{2}, k+\frac{1}{2}}^{n+\frac{1}{2}} - H_x \Big|_{i, j+\frac{1}{2}, k-\frac{1}{2}}^{n+\frac{1}{2}} \right) / \Delta z \right. \\
 & - \left(H_z \Big|_{i+\frac{1}{2}, j+\frac{1}{2}, k}^{n+\frac{1}{2}} - H_z \Big|_{i-\frac{1}{2}, j+\frac{1}{2}, k}^{n+\frac{1}{2}} \right) / \Delta x \\
 & \left. - J_y \Big|_{i, j+\frac{1}{2}, k}^{n+\frac{1}{2}} \right]
 \end{aligned} \tag{4}$$

$$\begin{aligned}
 E_z \Big|_{i, j, k+\frac{1}{2}}^{n+1} = & C_{a, E_z} \Big|_{i, j, k+\frac{1}{2}} E_z \Big|_{i, j, k+\frac{1}{2}}^n \\
 & + C_{b, E_z} \Big|_{i, j, k+\frac{1}{2}} \left[\left(H_y \Big|_{i+\frac{1}{2}, j, k+\frac{1}{2}}^{n+\frac{1}{2}} - H_y \Big|_{i-\frac{1}{2}, j, k+\frac{1}{2}}^{n+\frac{1}{2}} \right) / \Delta x \right. \\
 & - \left(H_x \Big|_{i, j+\frac{1}{2}, k+\frac{1}{2}}^{n+\frac{1}{2}} - H_x \Big|_{i, j-\frac{1}{2}, k+\frac{1}{2}}^{n+\frac{1}{2}} \right) / \Delta y \\
 & \left. - J_z \Big|_{i, j, k+\frac{1}{2}}^{n+\frac{1}{2}} \right]
 \end{aligned} \tag{5}$$

Chapter 2: Basic Theory of Finite-Difference Time-Domain Method

Basic Theory of FDTD

where:

$$C_{a,\gamma}|_{i,j,k} = \left(1 - \frac{\sigma_{i,j,k}\Delta t}{2\varepsilon_{i,j,k}}\right) / \left(1 + \frac{\sigma_{i,j,k}\Delta t}{2\varepsilon_{i,j,k}}\right), \quad \gamma = E_x, E_y, E_z \quad (6)$$

$$C_{b,\gamma}|_{i,j,k} = \left(\frac{\Delta t}{\varepsilon_{i,j,k}}\right) / \left(1 + \frac{\sigma_{i,j,k}\Delta t}{2\varepsilon_{i,j,k}}\right), \quad \gamma = E_x, E_y, E_z$$

The discretized x-, y-, and z-components of the H-fields for the point (i, j, k) are:

$$\begin{aligned} H_x \Big|_{i,j+\frac{1}{2},k+\frac{1}{2}}^{n+\frac{1}{2}} &= D_{a,H_x} \Big|_{i,j+\frac{1}{2},k+\frac{1}{2}} H_x \Big|_{i,j+\frac{1}{2},k+\frac{1}{2}}^{n-\frac{1}{2}} \\ &+ D_{b,H_x} \Big|_{i,j+\frac{1}{2},k+\frac{1}{2}} \left[\left(E_y \Big|_{i,j+\frac{1}{2},k+1}^n - E_y \Big|_{i,j+\frac{1}{2},k}^n \right) / \Delta z \right. \\ &- \left(E_z \Big|_{i,j+1,k+\frac{1}{2}}^n - E_z \Big|_{i,j,k+\frac{1}{2}}^n \right) / \Delta y \\ &\left. - M_x \Big|_{i,j+\frac{1}{2},k+\frac{1}{2}}^n \right] \end{aligned} \quad (7)$$

$$\begin{aligned} H_y \Big|_{i+\frac{1}{2},j,k+\frac{1}{2}}^{n+\frac{1}{2}} &= D_{a,H_y} \Big|_{i+\frac{1}{2},j,k+\frac{1}{2}} H_y \Big|_{i+\frac{1}{2},j,k+\frac{1}{2}}^{n-\frac{1}{2}} \\ &+ D_{b,H_y} \Big|_{i+\frac{1}{2},j,k+\frac{1}{2}} \left[\left(E_z \Big|_{i+1,j,k+\frac{1}{2}}^n - E_z \Big|_{i,j,k+\frac{1}{2}}^n \right) / \Delta x \right. \\ &- \left(E_x \Big|_{i+\frac{1}{2},j,k+1}^n - E_x \Big|_{i+\frac{1}{2},j,k}^n \right) / \Delta z \\ &\left. - M_y \Big|_{i+\frac{1}{2},j,k+\frac{1}{2}}^n \right] \end{aligned} \quad (8)$$

$$\begin{aligned} H_z \Big|_{i+\frac{1}{2},j+\frac{1}{2},k}^{n+\frac{1}{2}} &= D_{a,H_z} \Big|_{i+\frac{1}{2},j+\frac{1}{2},k} H_z \Big|_{i+\frac{1}{2},j+\frac{1}{2},k}^{n-\frac{1}{2}} \\ &+ D_{b,H_z} \Big|_{i+\frac{1}{2},j+\frac{1}{2},k} \left[\left(E_x \Big|_{i+\frac{1}{2},j+1,k}^n - E_x \Big|_{i+\frac{1}{2},j,k}^n \right) / \Delta y \right. \\ &- \left(E_y \Big|_{i+1,j+\frac{1}{2},k}^n - E_y \Big|_{i,j+\frac{1}{2},k}^n \right) / \Delta x \\ &\left. - M_z \Big|_{i+\frac{1}{2},j+\frac{1}{2},k}^n \right] \end{aligned} \quad (9)$$

where:

$$\begin{aligned} D_{a,\gamma}|_{i,j,k} &= \left(1 - \frac{\sigma_{i,j,k}^* \Delta t}{2\mu_{i,j,k}}\right) \gamma \left(1 + \frac{\sigma_{i,j,k}^* \Delta t}{2\mu_{i,j,k}}\right), \quad \gamma = H_x, H_y, H_z \\ D_{b,\gamma}|_{i,j,k} &= \left(\frac{\Delta t}{\mu_{i,j,k}}\right) \gamma \left(1 + \frac{\sigma_{i,j,k}^* \Delta t}{2\mu_{i,j,k}}\right), \quad \gamma = H_x, H_y, H_z \end{aligned} \quad (10)$$

In the above equations, Δt is the time step and it must satisfy the [Stability \(Courant\) Criteria](#).

Note:

Magnetic material is not supported, that is, $\mu = \mu_0$ and $\sigma^* = 0$ in [Equation 1](#) and the corresponding equations. Here, μ_0 denotes the vacuum magnetic permeability.

Accuracy of the Finite Difference Stencil

The accuracy of the finite difference stencil critically depends on cell size and mesh uniformity. For a uniform tensor mesh, the Yee algorithm [\[3\]](#) is second-order accurate for both time and spatial discretization. However, the accuracy will degrade if the mesh is nonuniform, because the numeric dispersion of the finite difference method increases when the cells have large aspect ratios or abrupt grading or both.

Note:

Whenever an FDTD simulation has slow convergence or unstable results, first check whether convergence and stability can be improved by using a denser uniform mesh for the same problem.

Stability (Courant) Criteria

The explicit FDTD-updating algorithm initially proposed by Yee [\[3\]](#) is conditionally stable, which means that there is an upper limit to the allowed time step for stable operation:

$$\Delta t \leq \Delta t_{\text{stable}} \quad (11)$$

Time steps that are greater than Δt_{stable} cause exponentially growing solutions. The theoretical proof of the stability criterion is usually given for the linear case on uniform grids, derived by examining how a plane wave is impacted by the FDTD discretized stencil.

However, numeric experiments have shown that the following selection of Δt is also satisfactory for rectilinear (nonuniform) grids and in combination with the commonly used boundary conditions:

3D case:

$$\Delta t_{\text{stable}} = \frac{1}{v \sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}}} \quad (12)$$

2D case:

$$\Delta t_{\text{stable}} = \frac{1}{v \sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}}} \quad (13)$$

where v denotes the phase velocity of the wave, and Δx , Δy , and Δz are the space increments of the grid.

Note:

In the case of a rectilinear grid, the smallest space increments of the grid must be used in [Equation 12](#) and [Equation 13](#).

Modeling of Dispersive Media

In the derivation of the FDTD-updating scheme, which forms the basis of EMW, it is generally assumed that the material parameters ϵ , μ , σ , and σ_m are independent of frequency. As long as interest is restricted to model situations with quasimonochromatic excitation, the formulations in the previous sections are sufficient. However, the electromagnetic parameters of most materials generally exhibit a dependency on frequency that might be pronounced, especially in the range of visible wavelengths and frequencies.

If the excitation signal has a spectrum that is nonzero over a range of frequencies in which the material properties change significantly, the dispersive behavior must be taken into account in the simulation. In addition, for a highly absorbing material that has a complex refractive index $n + i\kappa$ such that $\kappa > n$ (especially true for some metals), the real part of the dielectric constant $\text{Re}[\epsilon_r] = n^2 - \kappa^2$ becomes negative and will result in instabilities when the nondispersive FDTD is used. This problem can be solved by treating the material as dispersive.

Various approaches for taking frequency-dependent material parameters into account in FDTD have emerged [\[1\]](#). Adequate treatment of material dispersion is important for simulating the propagation of short pulses as well as for efficient modeling of propagation phenomena over a wide range of frequencies in the optical and microwave regimes.

Chapter 2: Basic Theory of Finite-Difference Time-Domain Method

Basic Theory of FDTD

The common methods are based on three main principles:

- The z-transformation of Maxwell's curl equations.
- The solution of auxiliary differential equations (ADEs) between the electric displacement density and the polarization.
- The recursive computation of the convolution integrals between the electric field and the time-domain susceptibility.

Numerous investigations have shown that the ADE method offers very good accuracy and stability with moderate computational effort. In addition, it is very flexible with respect to the modeling of common types of dispersive material.

Note:

Only the ADE method is used in EMW.

Since the majority of materials of interest for various applications is not magnetically dispersive, EMW is restricted to the modeling of frequency-dependent electric materials.

Physical Model

If the electric permittivity depends on the frequency, the constitutive relation between the electric field and the displacement density becomes:

$$\bar{D}(\omega) = \varepsilon(\omega)\bar{E}(\omega) = \varepsilon_0\varepsilon_\infty\bar{E}(\omega) + \bar{P}(\omega) = \varepsilon_0[\varepsilon_\infty + \chi(\omega)]\bar{E}(\omega) \quad (14)$$

where:

- $\varepsilon(\omega)$ is the permittivity (generally a complex quantity).
- $\bar{P} = \varepsilon_0\chi(\omega)\bar{E}(\omega)$ is the electric polarization.
- $\chi(\omega) = \varepsilon(\omega)/\varepsilon_0 - \varepsilon_\infty$ is the electric susceptibility.

The frequency dispersive characteristics can be fully described by the electric susceptibility. The actual form of the susceptibility depends on the microscopic properties of the material.

Next is an overview of the common types of dispersive material model encountered in electromagnetics.

Debye Relaxation

The Debye dispersive model describes the relaxation behavior of water molecules. Therefore, this dispersive type plays an important role in modeling electromagnetic polarization in biological tissues.

The susceptibility of a multiterm Debye dispersive medium is given by:

$$\chi(\omega) = \sum_{p=1}^{N_p^{\text{Debye}}} \frac{\Delta\epsilon_p}{1 - i\omega\tau_p} \quad (15)$$

where:

- N_p^{Debye} denotes the number of Debye poles.
- $\Delta\epsilon_p = (\epsilon_s - \epsilon_\infty)A_p$ is the change in the relative permittivity due to the p^{th} pole.
- A_p is the amplitude.
- τ_p is the pole relaxation time.

Lorentzian Resonance

The Lorentzian dispersive model adequately describes the frequency-dependent response caused by damped oscillation of bounded electrons due to the applied external fields, and it is typically used for modeling semiconductors and insulators.

The susceptibility of a multiterm Lorentzian dispersive medium is given by:

$$\chi(\omega) = \sum_{p=1}^{N_p^{\text{Lorentz}}} \frac{\Delta\epsilon_p \omega_p^2}{\omega_p^2 - 2i\omega\delta_p - \omega^2} \quad (16)$$

where:

- N_p^{Lorentz} denotes the number of Lorentzian pole pairs.
- $\Delta\epsilon_p = (\epsilon_s - \epsilon_\infty)A_p$ is the change in the relative permittivity due to the p^{th} pole pair.
- A_p is the amplitude.
- ω_p is the undamped frequency of the p^{th} pole pair, and δ_p is its damping factor.

Unmagnetized Plasma

The Drude dispersive model describes the behavior that is exhibited by free electrons in metals. The susceptibility of a multiterm Drude medium is given by:

$$\chi(\omega) = - \sum_{p=1}^{N_p^{\text{Drude}}} \frac{\omega_p^2}{\omega^2 + i\omega\gamma_p} \quad (17)$$

where:

- N_p^{Drude} denotes the number of Drude poles.
- ω_p is the Drude pole frequency.
- γ_p is the inverse of the pole relaxation time or Drude damping factor.

Drude–Lorentz

For modeling certain semiconductors and metals with higher accuracy, you can combine the Lorentz and Drude models to characterize dispersive media over a wide frequency range [4]. The susceptibility of a multiterm Drude–Lorentz medium is given by:

$$\chi(\omega) = \sum_{p=1}^{N_p^{\text{Lorentz}}} \frac{\Delta\epsilon_p \omega_p^2}{\omega_p^2 - 2i\omega\delta_p - \omega^2} - \sum_{p=1}^{N_p^{\text{Drude}}} \frac{\omega_p^2}{\omega^2 + i\omega\gamma_p} \quad (18)$$

where the dispersive parameters are defined as in [Equation 16](#) and [Equation 17](#).

Modified Lorentz

Another effective dispersive model for semiconductors and metals is the modified Lorentz model. This model offers a better description of the dielectric function of silicon, aluminum, silver, and gold in the visible wavelength range, although the parameters in this model do not necessarily have a physical meaning [5][6]. The susceptibility of a multiterm modified-Lorentz medium is given by:

$$\chi(\omega) = \sum_{p=1}^{N_p^{\text{ModLorentz}}} \frac{\Delta\epsilon_p (\omega_p^2 - i\omega\delta_p')}{\omega_p^2 - 2i\omega\delta_p - \omega^2} \quad (19)$$

Compared with the Lorentz model in [Equation 16](#), δ_p' in the numerator is an additional damping factor that describes the direct polarization of the dispersive medium by the time derivative of the applied electric field [Equation 19](#).

Drude–Modified Lorentz

The Drude model can be combined with the modified Lorentz model to form the Drude–modified Lorentz model. The susceptibility of a multiterm Drude–modified Lorentz medium is given by:

$$\chi(\omega) = \sum_{p=1}^{N_p^{\text{ModLorentz}}} \frac{\Delta\epsilon_p (\omega_p^2 - i\omega\delta_p')}{\omega_p^2 - 2i\omega\delta_p - \omega^2} - \sum_{p=1}^{N_p^{\text{Drude}}} \frac{\omega_p^2}{\omega^2 + i\omega\gamma_p} \quad (20)$$

where the dispersive parameters are defined as in [Equation 17](#).

The formulations of the ADEs and associated convolutional perfectly matching layers (CPMLs) of the above dispersive materials are described in detail in [\[1\]](#).

Caution:

Dispersive materials can be attached only to CPMLs, not to analytic absorbing boundaries. The physics and derivation of the above dispersive models are available in the literature [\[4\]\[7\]](#).

For the syntax for activating the dispersive models, see [Dispersive Model Parameters on page 44](#).

References

- [1] A. Taflove and S. C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Boston: Artech House, 3rd ed., 2005.
- [2] P. Harms, R. Mittra, and W. Ko, "Implementation of the Periodic Boundary Condition in the Finite-Difference Time-Domain Algorithm for FSS Structures," *IEEE Transactions on Antennas and Propagation*, vol. 42, no. 9, pp. 1317–1324, 1994.
- [3] K. S. Yee, "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media," *IEEE Transactions on Antennas and Propagation*, vol. AP-14, no. 3, pp. 302–307, 1966.
- [4] *Handbook of Optical Constants of Solids III*, E. D. Palik (ed.), San Diego: Academic Press, 1998.
- [5] A. Deinega and S. John, "Effective optical response of silicon to sunlight in the finite-difference time-domain method," *Optics Letters*, vol. 37, no. 1, pp. 112–114, 2012.
- [6] A. Vial *et al.*, "A new model of dispersion for metals leading to a more accurate modeling of plasmonic structures using the FDTD method," *Applied Physics A*, vol. 103, no. 3, pp. 849–853, 2011.
- [7] M. Dressel and G. Grüner, *Electrodynamics of Solids: Optical Properties of Electrons in Matter*, Cambridge: Cambridge University Press, 2002.

3

Boundary Conditions

This chapter describes the boundary conditions that can be set up in Sentaurus Device Electromagnetic Wave Solver (EMW).

Introduction

Setting the right boundary conditions and the corresponding parameters, if required by the specific type of boundary condition, can play a crucial role in the quality of the simulation results.

The typical simulation setup for CMOS image sensor (CIS) and solar-cell applications is a normal or an oblique incident plane wave with periodic boundary conditions (BCs) at the sides and absorbing BCs at the top and bottom of the structure. Periodic BCs at the sides are inherent to the sine–cosine formulation [\[1\]](#) and require no additional configuration parameters to be set by the user. Absorbing BCs are discussed here.

PEC and PMC Boundary Conditions

Perfect electric conductor (PEC) and perfect magnetic conductor (PMC) boundary conditions are defined as BCs with zero tangential components of E- and H-fields, respectively.

An example of specifying PEC boundary conditions at the terminating sides in the x-direction is:

```
Boundary {  
    Type = PEC  
    Sides = {X}  
}
```

Periodic Boundary Conditions

Two types of periodic boundary condition are available:

- Periodic BC for normal incidence (`periodic`)
- Periodic BC for oblique incidence (`periodicoblique`)

The `periodic` BC applies to plane wave excitations with normal incidence. The E- and H-fields are copied directly from one periodic facet to the opposing one during field update. Since only real fields are updated in the FDTD loop, the simulation is faster than the sine–cosine method [1], where both the real and imaginary fields must be updated.

The `periodicoblique` BC applies to plane wave excitations with both normal and oblique incidence. The E- and H-fields are updated by the sine–cosine method at periodic boundaries with a phase shift that depends on the incident angle.

Note:

For stability reasons, using the `periodicoblique` BC is recommended even for plane wave excitation with normal incidence.

Absorbing Boundary Conditions

Three types of absorbing boundary condition are available:

- Mur (`Mur`)
- Higdon (`Higdon`)
- Convolutional perfectly matched layer (CPML) (`CPML`)

Mur and Higdon BCs are computationally less costly than CPML BCs and do not require the specification of further-tuning parameters. Usually, the Higdon BC absorbs traveling waves incident on the boundary better than the Mur BC.

The most advanced absorbing BC is the CPML [2]. It generally gives the best results and can absorb waves impinging the boundary with a wide range of incidence angles. However, its quality depends on the choice of configuration parameters and it comes at a higher computational cost.

Note:

For sine–cosine simulation with large incident angles, it is preferable to use CPML to eliminate unwanted angle-dependent scatterings of the simulation results.

Absorbing boundaries and CPMLs can be placed on all sides of the simulation domain. This is useful for studying the scattering from a standalone structure instead of an array of

Chapter 3: Boundary Conditions

Absorbing Boundary Conditions

structures. However, you cannot adjoin Mur and Higdon absorbing boundary conditions to CPMLs. Either all external boundaries are CPMLs or all external boundaries are Mur or Higdon absorbing BCs.

An example of specifying 25-layer CPMLs for all sides is:

```
Boundary {  
    Type = CPML  
    Sides = {X, Y, Z}  
    ...  
    Thickness = 25  
}
```

Specifying Parameters of CPML Boundary Condition

The CPML BC is configured with three different parameter profiles, σ , κ , and α , which determine the properties of each CPML. The number of layers can be specified independently for each side by setting the keyword `Thickness` to the corresponding integer value. The actual thickness of each CPML is predetermined by the tensor cell-size adjacent to the respective boundary. The two options of specifying the parameter profiles are either explicitly or by defining a polynomial grading.

For example, the σ profile of a five-layer CPML boundary at the bottom can be explicitly specified in the command file by using:

```
Boundary {  
    Type = CPML  
    Sides = {Zmin}  
    Thickness = 5  
    SigmaProfile = {0.003, 0.04, 0.09, 0.12, 0.18, 0.27, 0.49, 0.78,  
        0.92, 1.57}  
    ...  
}
```

The other profiles can be specified accordingly.

Note:

The number of values listed for each profile must be twice the number of CPMLs. The first value is assigned to the boundary interface, the second value is located half a cell into the CPML boundary, and so on. This is due to the staggered nature of the FDTD grid.

In the second option of polynomial grading, only two parameters for each profile must be specified: a maximum value and an exponent that determines how fast the value increases to its maximum within the CPML boundary.

Chapter 3: Boundary Conditions

Absorbing Boundary Conditions

This is shown in the following command file excerpt:

```
Boundary {  
    Type = CPML  
    Sides = {Zmax}  
    Thickness = 15  
    SigmaMax = 4.3e5      # unit: 1/(Ohm*m), for 20 nm cell size  
    OrderSigma = 3  
    KappaMax = 1.  
    OrderKappa = 1  
    AlphaMax = 0  
    OrderAlpha = 1  
    ...  
}
```

If the same parameters are to be used for both the lower boundary and the upper boundary in one coordinate direction, the keyword `Sides` can simply be set to `z` in the above example.

By default, values are taken as specified in [2]:

Keyword	Default value
AlphaMax	0.2
KappaMax	15
OrderAlpha	1
OrderKappa	3
OrderSigma	3
SigmaMax	σ_{opt}
Thickness	15

where σ_{opt} depends on the cell size and the refractive index of the neighboring region. It is calculated for each boundary individually:

$$\sigma_{\text{opt}} = \frac{0.8(m+1)}{n\Delta Z_0} \quad (21)$$

where:

- m is the value of `OrderSigma`.
- n is the refractive index.

Chapter 3: Boundary Conditions

Absorbing Boundary Conditions

- Δ is the cell size of the neighboring cell.
- Z_0 is the vacuum impedance.

According to [2], this choice of σ_{opt} has proven to be relatively robust for many applications.

If the CPML quality needs to be improved, you can:

- Increase `Thickness` by a factor of two.
- Change `SigmaMax` by up to 50%.
- Change `AlphaMax` by up to 100%.

For more details on performance tuning of the CPML, refer to [2].

Combining CPML With Other Boundary Conditions

The CPML boundary condition can be combined with other types of boundary condition to suit various applications. Table 1 and Table 2 list the possible combinations for 2D and 3D simulations, respectively.

Note:

For Table 1, x_1, x_2 can be either x, y or y, x . For Table 2, x_1, x_2, x_3 can be either x, y, z or y, z, x or z, x, y .

Table 1 *Boundary condition combinations with CPML in two dimensions*

$x_{1,\text{min}}$	$x_{1,\text{max}}$	$x_{2,\text{min}}$	$x_{2,\text{max}}$
PeriodicOblique	PeriodicOblique	CPML	CPML
Periodic	Periodic	CPML	CPML
PEC	PEC	CPML	CPML
PMC	PMC	CPML	CPML
PEC	CPML	CPML	CPML
CPML	PEC	CPML	CPML
PMC	CPML	CPML	CPML
CPML	PMC	CPML	CPML
CPML	CPML	CPML	CPML

Chapter 3: Boundary Conditions

References

Table 2 *Boundary condition combinations with CPML in three dimensions*

X_{1,min}	X_{1,max}	X_{2,min}	X_{2,max}	X_{3,min}	X_{3,max}
PeriodicOblique	PeriodicOblique	PeriodicOblique	PeriodicOblique	CPML	CPML
Periodic	Periodic	Periodic	Periodic	CPML	CPML
PeriodicOblique	PeriodicOblique	CPML	CPML	CPML	CPML
Periodic	Periodic	CPML	CPML	CPML	CPML
CPML	CPML	CPML	CPML	CPML	CPML

References

- [1] P. Harms, R. Mittra, and W. Ko, "Implementation of the Periodic Boundary Condition in the Finite-Difference Time-Domain Algorithm for FSS Structures," *IEEE Transactions on Antennas and Propagation*, vol. 42, no. 9, pp. 1317–1324, 1994.
- [2] A. Taflove and S. C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Boston: Artech House, 3rd ed., 2005.

4

Building the Correct Mesh

This chapter describes how to build suitable meshes, to control mesh size, and to set up criteria for mesh quality and stability.

Generating Tensor Meshes for EMW

To perform FDTD simulations with EMW, a tensor mesh must be generated using Sentauros Mesh. The size and quality of the tensor mesh have significant impact on the FDTD simulation time and the accuracy of the results.

Note:

The success of the FDTD simulation depends critically on obtaining a good-quality tensor mesh. An unstable simulation generally indicates the violation of the stability (Courant) criteria, and the first thing to inspect is the size of the smallest mesh cell and the time step.

This chapter reviews some important concepts of tensor-mesh generation for EMW using Sentauros Mesh (see the *Sentauros™ Mesh User Guide* for details).

Using the Sentauros Mesh Tensor-Product Generator

The tensor-product meshes for EMW are generated by Sentauros Mesh using the command:

```
snmesh -t -emw <base_name>_msh
```

Sentauros Mesh checks the corresponding input files `<base_name>_bnd.tdr` and `<base_name>_msh.cmd`, and then applies the tensor-mesh discretization to a device domain according to the command file specifications. The resulting tensor mesh is saved as `<base_name>_msh.tdr`.

Chapter 4: Building the Correct Mesh

Using the Sentaurus Mesh Tensor-Product Generator

All of the necessary controls to generate the tensor mesh are specified in the `Tensor` section of the command file `<base_name>_msh.cmd`, which can include different `Mesh` and `EMW` subsections:

```
Tensor {  
    Mesh {...}  
    EMW {...}  
}
```

where:

- The `Mesh` subsection defines the general tensor-grid controls.
- The `EMW` subsection defines dedicated controls for EMW applications.

In the following, the most frequently used controls to obtain a suitable mesh for EMW applications are listed:

```
Mesh {  
    maxCellSize = float  
    maxCellSize material "materialName" float  
    maxCellSize direction "x|y|z" float  
    maxCellSize material direction "materialName" "x|y|z" float  
    minCellSize = float  
    minCellSize direction "x|y|z" float  
}
```

where:

- `maxCellSize` specifies the maximum cell size allowed in the tensor grid, directionally or in all spatial directions. For efficient and accurate EMW simulations, it is suggested that `maxCellSize` is smaller than 1/10th of the wave periodicity in the respective materials (that is, more than 10 nodes per wavelength).
- `minCellSize` specifies the minimum cell size in micrometers (default is 0.0001). Since the FDTD time step in EMW depends on the minimum cell size in the tensor grid (the Courant–Friedrichs–Lewy (CFL) stability condition). For efficient and fast EMW simulations with large stable time steps, it is suggested that `minCellSize` is set to a value of 0.001 or larger.

Specifying Nodes per Wavelength in Meshing

Using the `EMW` subsection is an alternative (or additive) solution to control the tensor-mesh generation. The controls in the `EMW` subsection allow for the structure to be meshed by specifying *nodes per wavelength* criteria directly:

```
EMW {  
    wavelength = float  
    parameter filename = "string"
```

Chapter 4: Building the Correct Mesh

Using the Sentaurus Mesh Tensor-Product Generator

```
nodesperwavelength | npw {
    material "materialName" float
    material direction "materialName" "x|y|z" float
    region "regionName" float
    region direction "regionName" "x|y|z" float
}
nodesperwavelengthX | npwx = integer
nodesperwavelengthY | npwy = integer
nodesperwavelengthZ | npwz = integer
nodesperwavelength | npw = integer

ComplexRefractiveIndex WavelengthDep Real Imag
ComplexRefractiveIndex region "regionName" WavelengthDep
[Real | Imag]
ComplexRefractiveIndex material "materialName" WavelengthDep
[Real | Imag]

grading off
}
```

where:

- `wavelength` specifies the wavelength of the incident wave in micrometers (default is 0.555).
- `parameter filename` indicates a user-defined parameter file that contains the `ComplexRefractiveIndex` optical database table of the materials that are present in the device structure. See the *Sentaurus™ Device User Guide* for details.
- `nodesperwavelength | npw` defines the value of the nodes per wavelength regionwise or materialwise, directionally or in all spatial directions. Alternatively, the value in each spatial direction can be set for all materials using either of the `npw` statements.
- `ComplexRefractiveIndex` defines the quantity of the complex refractive index evaluated at `wavelength` that should be used to refine according to `nodesperwavelength | npw`.
- The `grading off` statement is used to specify that no mesh grading should be applied at interfaces. Mesh grading causes overly thin layers at interfaces, and the cell size increases with distance from the interface.

Note:

The `grading on` statement (the default) results in a graded mesh that can negatively impact accuracy of the EMW simulation results, which is caused by numeric dispersion.

All controls available for `material` are also available for `region`. A complete list of controls is available in the *Sentaurus™ Mesh User Guide*.

Meshing Examples

Two examples of generating suitable tensor grids for EMW simulations are presented here. First, tensor-grid generation for a simple planar two-layer silicon–air example is performed. Such simple examples are usually used to benchmark FDTD results for `OpticalGeneration` and `PowerFluxDensity` against their analytic values. Second, a Sentauros Mesh command file is shown for tensor-grid generation that is typical for image sensor applications.

Example 1: Planar Two-Layer Silicon–Air

Tensor-mesh generation with Sentauros Mesh requires two input files: a boundary file and a command file. The boundary file for a simple planar two-layer silicon–air structure can be generated by Sentauros Structure Editor using the following script (`simple_dvs.cmd`):

```
(sde:clear)
(sdegeo:create-cuboid (position 0 0 -2.0) (position 0.1 0.1 0.0)
  "Silicon" "R.Substrate")
(sdegeo:create-cuboid (position 0 0 0.0) (position 0.1 0.1 1.0)
  "Ambient" "R.Ambient")
(sdeio:save-tdr-bnd (get-body-list) "simple_bnd.tdr")
```

Running Sentauros Structure Editor in batch mode using the script (`simple_dvs.cmd`):

```
sde -e -l simple_dvs.cmd
```

produces the boundary file `simple_bnd.tdr`. A possible Sentauros Mesh command file (`simple_msh.cmd`) for tensor-mesh generation can look like:

```
Tensor {
  Mesh {
    maxCellSize = 0.02
    maxCellSize direction "z" 0.01
    minCellSize = 0.005
  }
}
```

Here, the controls define the maximum cell size allowed in the tensor grid to be 0.02 μm . It is typical to define a fine mesh in the main propagation direction. The main propagation direction is assumed to be z in this example, so the maximum cell size in the z-direction is limited to 0.01 μm .

To ensure that the FDTD time step in EMW will be sufficiently large, the minimum cell size allowed is set to 0.005 μm .

Calling Sentauros Mesh with:

```
snmesh -t -emw simple_msh
```

produces the tensor-mesh file `simple_msh.tdr`, which is suitable for FDTD simulation. The mesh file can be visualized with Sentaurus Visual using:

```
svisual simple_msh.tdr &
```

Example 2: Image Sensor

In this section, the meshing controls to obtain a suitable tensor mesh for an EMW image sensor simulation are described only, without the Sentaurus Structure Editor command files. For details of generating boundary files with Sentaurus Structure Editor, see the *Sentaurus™ Structure Editor User Guide* or the Sentaurus Structure Editor module of the TCAD Sentaurus Tutorial.

Image sensors contain materials with different wavelength-dependent refractive indices and, therefore, different wave periodicities in the materials.

Instead of defining `minCellSize` for all materials separately to obtain a tensor grid with a reasonable number of nodes per wavelength, the `EMW` subsection will be used to define the nodes per wavelength criteria directly (`cis_msh.cmd`):

```
Tensor {
  Mesh {
    minCellSize direction "x" 0.005
    minCellSize direction "y" 0.005
    minCellSize direction "z" 0.003
  }
  EMW {
    Parameter Filename = "emw.par"
    Wavelength = 0.6
    ComplexRefractiveIndex WavelengthDep real

    npwx = 10
    npwy = 10
    npwz = 15
    grading off
  }
}
```

In this example, `minCellSize` is specified for each different direction. To fully resolve thin layers in the z-direction, a denser mesh is allowed.

Note:

There is always a trade-off between high geometric resolution (small `minCellSize`) and the FDTD simulation time step in EMW that benefits from a larger `minCellSize`.

Chapter 4: Building the Correct Mesh

Meshing Examples

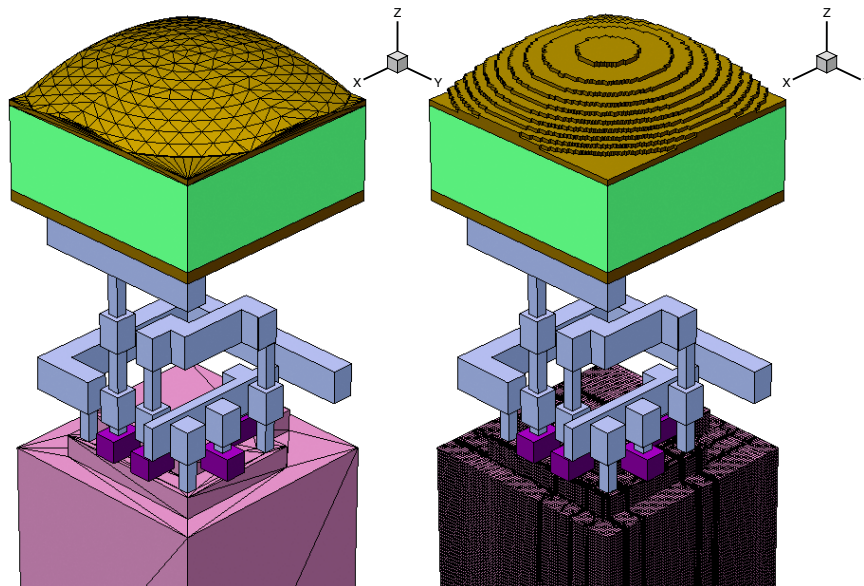
In the file `emw.par`, the complex refractive indices are defined. For example:

```
Material = "Silicon" {  
  ComplexRefractiveIndex {  
    Formula = 1          ** use NumericalTable  
    NumericalTable (    ** wavelength [um], n [1], k [1]  
      ...  
      0.5904  3.969  0.03;  
      0.5961  3.956  0.027;  
      0.6019  3.943  0.025;  
      0.6078  3.931  0.025;  
      0.6138  3.918  0.024;  
      ...  
    )  
  }  
}
```

The `wavelength` is set to $0.6\ \mu\text{m}$. The wave periodicity in the material is calculated using the real part of the complex refractive index at `wavelength`. In addition, `npwx=10` and `npwy=10` define that the tensor grid should consist of at least 10 nodes per wavelength for all materials in the x- and y-direction. In the main propagation direction (z-direction), a finer discretization is chosen with `npwz=15`.

To minimize the impact of numeric dispersion, no mesh grading is used (`grading off`). Applying the command file `cis_msh.cmd` to a typical image sensor boundary will result in a suitable tensor mesh for EMW simulations (see [Figure 3](#)).

Figure 3 (Left) Visualization of typical image sensor boundary (oxide and vacuum regions not shown). (Right) Visualization of corresponding tensor grid generated by Sentaurus Mesh using the command file `cis_msh.cmd`. Grid lines are shown only in silicon. The lens shape is given as a staircase caused by the tensor-grid approximation.



Inspecting and Troubleshooting Tensor-Grid Generation

If EMW simulations do not work as expected, the tensor grid is most often the problem. If the grid is too coarse, simulation results are not accurate. If the grid is too fine, the number of cells in the tensor grid will be huge (for example, 200 million cells), which will result in excessive EMW simulation times. If the smallest cell in the tensor grid is too small, the resulting FDTD time step will be small, which will also lead to excessive simulation times.

Therefore, it is recommended to perform some basic EMW inspections:

- Check whether the FDTD time step is in a reasonable range: It should be between 5×10^{-19} s and 5×10^{-17} s for visible light excitations. Note that the FDTD time step is proportional to the minimum cell size.
- Check whether the geometry is approximated well enough by the staircase tensor grid.
- Vary `maxCellSize` or `npw` to analyze whether the EMW simulation results are accurate and independent of meshing parameters.

5

Specifying Material Parameters

This chapter explains how to specify material parameters and dispersive material models for Sentaurus Device Electromagnetic Wave Solver (EMW) simulations.

Introduction

The material parameters used in EMW are based on the complex refractive index (CRI) module. Parameters used in the fundamental equations of the FDTD method (see [Chapter 2 on page 16](#)), such as permittivity and conductivity or dispersive model parameters, are derived from it internally. To maintain backward compatibility, it is still possible to specify these parameters explicitly in the parameter file.

EMW supports the same parameter file specification of the CRI that is used by different tools such as Sentaurus Device and Sentaurus Mesh and, therefore, ensures consistent input data. This includes the C++ application programming interface (API), which allows for the specification of user-defined models.

In general, the CRI is defined by selecting a model in the command file and by specifying the corresponding model parameters for each region or material in the parameter file. A model can be assigned to a material, a region, or the entire device. The name of the parameter file must be given in the `Globals` section using the keyword `ParameterFile`; otherwise, default parameters are used if they exist for a requested material.

Complex Refractive Index Model

To activate the CRI model, a `ComplexRefractiveIndex` section must be defined in the command file that specifies the dependencies of the CRI for a given region or material. If neither a region nor a material is assigned to the `ComplexRefractiveIndex` section, it is assumed that the dependencies apply to all regions of the device structure. EMW supports all dependencies of the CRI given in *Sentaurus™ Device User Guide*, Complex Refractive Index Model. However, some dependencies are based on spatially varying fields such as charge carrier density or temperature, which are not natively available in the input tensor-grid file. In this case, an extra TDR file containing the datasets required by the

Chapter 5: Specifying Material Parameters

Complex Refractive Index Model Interface

specified dependencies must be provided, which is described in [Spatially Dependent Complex Refractive Index Model on page 41](#).

For a spatially constant CRI, that is, its value is constant within a region or material, supported dependencies are wavelength and temperature.

The following example shows how to use a CRI in a region called `substrate` whose real and imaginary parts depend on the excitation wavelength and its real part depends on the global device temperature:

```
Globals {
    Temperature = 380    # global device temperature [K]
}
ComplexRefractiveIndex {
    Region = "substrate"
    WavelengthDep = {Real, Imag}
    TemperatureDep = {Real}
}
```

Note:

Region sections always overwrite material sections if, for a given region, both a material and a region section exist in the command file.

Temperature dependency is supported for both a spatially constant CRI and a spatially dependent CRI. For the former, a global device temperature must be specified in the `Globals` section; for the latter, an external temperature profile must be loaded.

The parameter `Formula` in the corresponding `ComplexRefractiveIndex` section in the parameter file determines the type of wavelength dependency, for example, tabulated values or an analytic formula.

For more details, see *Sentaurus™ Device User Guide*, Wavelength Dependency and *Sentaurus™ Device User Guide*, Using Complex Refractive Index.

Complex Refractive Index Model Interface

The complex refractive index model interface (CRIMI) allows the addition of new CRI models. These models must be implemented as C++ functions, and EMW loads the functions at runtime using the dynamic loader. No access to the EMW source code is necessary.

The generated shared object file containing the model implementation can be used together with Sentaurus Device and Sentaurus Mesh.

See *Sentaurus™ Device User Guide*, Complex Refractive Index Model Interface, and *Sentaurus™ Mesh User Guide*, Computing Cell Size Automatically.

Chapter 5: Specifying Material Parameters

Spatially Dependent Complex Refractive Index Model

Three main steps are required for integrating user-defined models:

1. First, a C++ class implementing the CRI model must be written.
2. Second, a shared object must be created that can be loaded at runtime.
3. Third, the model must be activated in the command file.

The first two steps are explained in detail in *Sentaurus™ Device User Guide*, C++ Application Programming Interface, and *Sentaurus™ Device User Guide*, Shared Object Code.

To load CRI models into EMW, the `CRIMIPath` search path must be defined in the `Globals` section of the command file. The value of `CRIMIPath` is allowed to contain multiple directory paths, separated by whitespace. For example:

```
Globals {  
    CRIMIPath = ". /home/joe/lib /home/mary/sdevice/lib"  
}
```

For each CRI model that appears in the `ComplexRefractiveIndex` section, the given directories are searched for a corresponding shared object file, for example, `modelname.so.linux64`.

A CRI model can be activated in the `ComplexRefractiveIndex` section of the command file by specifying the name of the model as shown in the following example:

```
ComplexRefractiveIndex {  
    Material = "Silicon"  
    CRIModel = "modelname"  
}
```

Note:

A CRI model name can only contain alphanumeric characters and underscores (`_`). The first character must be either a letter or an underscore. All CRI models can be specified regionwise or materialwise.

Spatially Dependent Complex Refractive Index Model

EMW allows for the definition of a spatially dependent CRI in contrast to a regionwise constant profile. For certain devices, the dependency of the CRI on temperature, carrier concentration, electric field, mole fraction, or any other spatially varying field might have a significant impact on their device characteristics. The CRIMI described in the previous section, paired with the ability to load arbitrary spatially varying field quantities into EMW, provides maximum flexibility for modeling any given CRI profile.

A spatially varying CRI can be either imported directly or defined in terms of the quantities on which it depends, which is controlled by the specified `Type` (`FromFile` or

Chapter 5: Specifying Material Parameters

Spatially Dependent Complex Refractive Index Model

Inhomogeneous) in the `ComplexRefractiveIndex` section of the command file. The file from which the CRI or its dependencies are read is specified in the `Globals` section using the `FieldDataFile` keyword.

To model a CRI that depends on wavelength, temperature, and carrier concentration, use the following syntax:

```
Globals {  
    FieldDataFile = "external_data_fields.tdr"  
}  
ComplexRefractiveIndex {  
    Type = Inhomogeneous  
    WavelengthDep = {Real, Imag}  
    CarrierDep = {Real, Imag}  
    TemperatureDep = {Real}  
}
```

If `Type=FromFile` in the `ComplexRefractiveIndex` section, the TDR file specified by `FieldDataFile` must contain the datasets `cplxRefIndex` and `cplxExtCoeff`, which are associated with the real and imaginary parts of the CRI, respectively.

The datasets in the TDR file can be defined on any arbitrary tensor or mixed-element grid with different region or material names than the EMW input tensor mesh. However, the grids must overlap within the global coordinate system. The CRI will be computed on the grid of the `FieldDataFile` and then linearly interpolated onto the EMW tensor grid.

Note:

If the CRIMI is used (see [Complex Refractive Index Model Interface on page 40](#)), then you must specify `Type=Inhomogeneous` in the `ComplexRefractiveIndex` section. By default, all datasets available in the `FieldDataFile` are loaded and can be accessed to compute the CRI. However, they must be registered in the constructor of the respective CRI model before they can be used.

Spatially Dependent Dispersive Media

For device structures with inhomogeneous CRI profiles whose imaginary part is larger than its real part in certain domains, dispersive modeling must be applied. From a physical point of view, it is possible that the CRI differs at each tensor vertex and so would the dispersive media associated with it. Depending on the number of affected mesh vertices, the increased computational effort that comes with dispersive modeling can have a significant impact on the overall memory consumption and performance. To balance performance with accuracy, different options are available for the dispersive modeling of spatially dependent CRI domains.

Chapter 5: Specifying Material Parameters

Spatially Dependent Complex Refractive Index Model

The most accurate but potentially also the most demanding option in terms of computation is to create a dispersive medium for each unique CRI value whose imaginary part is larger than its real part ($n < k$):

```
ComplexRefractiveIndex {
  Type = fromFile
  Material = "Silicon"
  MaxDispersiveMedia = -1
}

DispersiveMedia {
  Material = {"Silicon"}
  Model = SingleDipoleDrude
  ModelParameters = ComputeFromComplexRefractiveIndex
}
```

In this example, setting `MaxDispersiveMedia = -1` indicates that the number of dispersive media is unlimited. The dispersive medium for each vertex is set to be computed from the imported CRI. A value of 0 (default) suppresses any dispersive modeling, and the imported CRI is assigned to each vertex regardless of $k > n$. Any vertex with $k > n$ will be modeled as a perfect electric conductor (PEC) to maintain stability.

To approximate the solution obtained by the preceding example with the benefit of improved performance, you can set `MaxDispersiveMedia = 1`, which only allows the use of a single user-supplied reference dispersive medium:

```
ComplexRefractiveIndex {
  Type = fromFile
  Material = "Silicon"
  MaxDispersiveMedia = 1
  DispersiveMedia = "liquidSilicon"
}

DispersiveMedia {
  Material = {"Silicon"}
  Model = SingleDipoleDrude
  ModelParameters = ComputeFromComplexRefractiveIndex
}

ComplexRefractiveIndex {
  Material = "liquidSilicon"
  WavelengthDep = {real, imag}
}
```

Here, the use of `DispersiveMedia` in the `ComplexRefractiveIndex` section specifies from which material the CRI is taken for the computation of the respective reference dispersive medium. Typically, the reference material name does not exist as a material in the tensor mesh. It is solely introduced in the parameter file to provide the respective CRI from which the reference dispersive medium is computed. The `DispersiveMedia{...}` section controls how EMW computes the CRI values for the vertices treated as dispersive media. The usage is identical as for regionwise constant dispersive media.

Limitations

The use of a spatially dependent CRI is subject to the following limitations:

- DFT simulations are not supported. Mole fraction dependency is only supported for wavelength dependency using numerical tables and for the CRIMI.
- If `Type=FromFile` in the `ComplexRefractiveIndex` section, then the `cplxRefIndex` dataset in the `FieldDataFile` must be defined on the entire domain given by the keywords `Region` and `Material`. Partial overlap will lead to an error stating that zero relative permittivity has been encountered.

Depending on the number of datasets in `FieldDataFile` that will be loaded and interpolated onto the input tensor mesh, the memory usage will increase temporarily during the material initialization phase. However, the memory footprint during the time-stepping loop will not be affected significantly.

Dispersive Model Parameters

To use one of the dispersive models described in [Modeling of Dispersive Media on page 21](#), it is necessary to choose the regions or materials in which the selected model should be applied in the EMW command file. The corresponding dispersive model parameters either are derived from the CRI or must be supplied directly by the user.

Dispersive Model Command Syntax

Activation of the dispersive material requires a separate `DispersiveMedia` section as shown here.

Assuming that materials named `DispMatA` and `DispMatB` in the tensor-product grid are of the dispersive type `Debye`, the following section must be included in the EMW command file:

```
DispersiveMedia {  
  Material = {"DispMatA" "DispMatB" ...}  
  # Region = {"reg1" "reg2" ...}  
  # ExcludeMaterial = {"mat1" "mat2" ...}  
  # ExcludeRegion = {"reg3" "reg4" ...}  
  Model = Debye # or Lorentz or Drude or DrudeLorentz  
  ModelParameters = UserDefined  
  DeltaK = <float>  
  InterfaceAveraging = <identifier>  
}
```

Chapter 5: Specifying Material Parameters

Dispersive Model Parameters

The valid dispersive models are `Debye`, `Drude`, `DrudeLorentz`, `DrudeModLorentz`, `Lorentz`, `ModLorentz`, and `SingleDipoleDrude`. The setting of `DeltaK` (Δk) activates the dispersive model when $k + \Delta k \geq n$. The dispersive model is always activated by setting `DeltaK` to a large number. In addition to setting the dispersive materials by `Material`, you can set the materials by `Region`, `ExcludeMaterial`, and `ExcludeRegion`.

If `InterfaceAveraging=Yes`, then a weighted average of the dielectric function in the frequency domain is used at the interface (electric field edges among adjacent cells). If `InterfaceAveraging=No`, then EMW uses the material parameter of the material with the highest precedence at the interface (electric field edges among adjacent cells). The criteria for material precedence are:

- Lossy dielectric < Debye < Lorentz < modified Lorentz < Drude < Drude–Lorentz < Drude–modified Lorentz < PMC < PEC.
- Within the same dispersive model, the following parameters (see [Specifying User-Defined Dispersive Model Poles in Parameter File on page 46](#)) are compared in sequence:

`epsilon_static`, `eps_infinity`, `sigma`, `nPole_E`, `E_pole_amplitude`,
`E_relaxation_time`, `E_pole_frequency`, `E_damping_factors`

The material with the highest value has highest precedence.

One limitation of the `InterfaceAveraging` keyword is that it must be `Yes` or `No` for all `DispersiveMedia` sections. Another limitation is that when `InterfaceAveraging=Yes`, all `DispersiveMedia` sections must not use `Model=Debye`. Averaging of the frequency-dependent dielectric function is performed only at interfaces with at least one dispersive region. At interfaces formed by nondispersive regions, only the dielectric constants, that is, ϵ_r , μ_r , ρ , and σ , are averaged.

Additional restrictions can be applied by limiting the dispersive type only to materials defined in `Material` but not in `ExcludeMaterial`, or in regions defined in `Region` but not in `ExcludeRegion`.

Note:

Only one dispersive model can be defined in each `DispersiveMedia` section.

For monochromatic excitations, a promising approach is to use a single-dipole Drude model (see [Unmagnetized Plasma on page 23](#)).

Setting ϵ_∞ to one, the Drude pole frequency (ω_p) and the Drude damping factor (γ_p) can be derived from the CRI. Therefore, the dispersive model `SingleDipoleDrude` must be used in conjunction with the CRI library (see [Complex Refractive Index Model on page 39](#)).

Chapter 5: Specifying Material Parameters

Dispersive Model Parameters

The following `DispersiveMedia` section shows how to use the `SingleDipoleDrude` model:

```
DispersiveMedia {
  Material = {"Gold"}
  Model = SingleDipoleDrude
  ModelParameters = ComputeFromComplexRefractiveIndex
  DeltaK = <float>
}
```

If `ModelParameters` is set to `ComputeFromComplexRefractiveIndex`, then the poles of the specific dispersive model are extracted automatically from the CRI values. If `UserDefined` is set, the poles of the dispersive model are loaded from the user-defined poles table of the material parameter file (see [Specifying User-Defined Dispersive Model Poles in Parameter File on page 46](#)).

Note:

`ComputeFromComplexRefractiveIndex` is supported only for the dispersive model `SingleDipoleDrude`.

For a summary of keywords supported in the `DispersiveMedia` section, see [Appendix A on page 108](#). You can consider using the dispersive model in the following situations:

- When $k \geq n$ for a material.
- When the dispersiveness (frequency dependency of n and k) of a material must be captured, for example, in a broadband DFT simulation.

Specifying User-Defined Dispersive Model Poles in Parameter File

The following syntax sets a material with the types Debye, Lorentz, modified Lorentz, and Drude, respectively, in a material parameter file:

```
Material = "Debye_materialname" {
  EMW {
    mu_r = mu_r
    sigma_m = sigma_m
    DebyeModel (
      Epsilon_static = epsilon_s
      Epsilon_inf = epsilon_infty
      Number_E_poles = N_p
      E_polesTable (
        # amplitude, relaxation times
        A_1 tau_1
        A_2 tau_2
        ...
      )
    )
  }
}
```

Chapter 5: Specifying Material Parameters

Dispersive Model Parameters

```
Material = "Lorentz_materialname" {
  EMW {
    mu_r = mu_r
    sigma_m = sigma_m
    LorentzModel (
      Epsilon_static = epsilon_s
      Epsilon_inf = epsilon_infty
      Number_E_poles = N_p
      E_polesTable (
        # amplitude, frequency (Hz), damping factor (rad/s)
        A_1 f_1 gamma_1
        A_2 f_2 gamma_2
        ...
      )
    )
  }
}

Material = "Modified_Lorentz_materialname" {
  EMW {
    mu_r = mu_r
    sigma_m = sigma_m
    ModLorentzModel (
      Epsilon_static = epsilon_s
      Epsilon_inf = epsilon_infty
      Number_E_poles = N_p
      E_polesTable (
        # amplitude, frequency (Hz), damping factor (rad/s),
        # damping factor prime (rad/s)
        A_1 f_1 gamma_1 gamma_prime_1
        A_2 f_2 gamma_2 gamma_prime_2
        ...
      )
    )
  }
}

Material = "Drude_materialname" {
  EMW {
    mu_r = mu_r
    sigma_m = sigma_m
    DrudeModel (
      Epsilon_inf = epsilon_infty
      Number_E_poles = N_p
      E_polesTable (
        # frequency (Hz) and damping factor (rad/s)
        f_1 gamma_1
        f_2 gamma_2
        ...
      )
    )
  }
}
```

Chapter 5: Specifying Material Parameters

Dispersive Model Parameters

For the Drude–Lorentz model, both the `LorentzModel` and `DrudeModel` sections must be specified simultaneously:

```
Material = "DrudeLorentz_materialname" {
  EMW {
    mu_r = mu_r
    sigma_m = sigma_m
    # Both Lorentz and Drude models are specified
    # to give a combination Drude-Lorentz model
    LorentzModel (
      Epsilon_static = epsilon_s
      Epsilon_inf = epsilon_infty
      Number_E_poles = N_p
      E_polesTable (
        # amplitude, frequency (Hz), damping factor (rad/s)
        A_1 f_1 gamma_1
        A_2 f_2 gamma_2
        ...
      )
    )
    DrudeModel (
      Epsilon_inf = epsilon_infty
      Number_E_poles = N_p
      E_polesTable (
        # frequency (Hz) and damping factor (rad/s)
        f_1 gamma_1
        f_2 gamma_2
        ...
      )
    )
  }
}
```

For the Drude–modified Lorentz model, both the `ModLorentzModel` and `DrudeModel` sections must be specified simultaneously:

```
Material = "DrudeModifiedLorentz_materialname" {
  EMW {
    mu_r = mu_r
    sigma_m = sigma_m
    # Both ModLorentz and Drude models are specified to give
    # a combination Drude-modified Lorentz model
    ModLorentzModel (
      Epsilon_static = epsilon_s
      Epsilon_inf = epsilon_infty
      Number_E_poles = N_p
      E_polesTable (
        # amplitude, frequency (Hz), damping factor (rad/s),
        # damping factor prime (rad/s)
        A_1 f_1 gamma_1 gamma_prime_1
        A_2 f_2 gamma_2 gamma_prime_2
      )
    )
  }
}
```


Chapter 5: Specifying Material Parameters

Explicit Specification of EMW Material Parameters

```
DrudeModel (
    Epsilon_inf = epsilon_infty
    Number_E_poles = N_p
    E_polesTable (
        # frequency (Hz) and damping factor (rad/s)
        f_1 gamma_1
        f_2 gamma_2
        ...
    )
)
}
```

Explicit Specification of EMW Material Parameters

If no `ComplexRefractiveIndex` section in the command file exists for a specific region or material, then EMW assumes that its material parameters are specified explicitly. To this end, an `EMW` section must be defined in the material parameter file where permittivity, permeability, and conductivity values are listed. For most simulations, the specification of values for the permittivity and conductivity is sufficient.

The following examples show `EMW` sections that should be added to the parameter file if the CRI model is not chosen.

For a materialwise definition, use:

```
Material = "Si3N4" {
    EMW {
        epsilon_r = 2.56
        mu_r      = 1.0
        sigma     = 0.0
        sigma_m   = 0.0
    }
}
```

For a regionwise definition, use:

```
Region = "Substrate" {
    EMW {
        epsilon_r = 16.6818
        mu_r      = 1.0
        sigma     = 10068.5
        sigma_m   = 0.0
    }
}
```

Note:

Region sections always overwrite material sections if, for a given region, both a material and a region section exist in the parameter file.

Since magnetic material is not supported, it is required that $\mu_r=1.0$ and $\sigma_m=0.0$ for all materials except PEC and PMC materials.

PEC and PMC Materials

Perfect electric conductor (PEC) and perfect magnetic conductor (PMC) materials are defined as materials with zero tangential components of E- and H-fields, respectively.

The reason for replacing lossy materials by PEC and PMC is to see the ideal situation whereby the electric and magnetic losses of a system are zero.

For a two-layer structure with vacuum (region `vac`) above a gold substrate (region `substrate`), the following are equivalent sections to force the substrate to be of PEC material:

```
PECMedia {  
    Region = {"substrate"}  
}  
  
PECMedia {  
    ExcludeRegion = {"vac"}  
}  
  
PECMedia {  
    Material = {"Gold"}  
}  
  
PECMedia {  
    ExcludeMaterial = {"Vacuum"}  
}
```

In a similar manner, the region or material can be set to PMC by using the `PMCMedia` section.

Precedence of Material Specification

In complex setups, different material models are assigned to specific regions or materials using the keywords `Region`, `Material`, `ExcludeRegion`, and `ExcludeMaterial`, in the respective material model section.

If you specify different material models for the same region, the following precedence applies (highest priority to lowest priority):

1. `PECMedia`
2. `PMCMedia`
3. `DispersiveMedia`

Chapter 5: Specifying Material Parameters

Precedence of Material Specification

4. `ComplexRefractiveIndex`
5. Explicit specification in `EMW` section of parameter file

6

Specifying Excitations

This chapter describes how to specify excitations.

Overview

An excitation is defined by its spatial and temporal dependencies. The spatial dependency is characterized by a constant phase wavefront whose propagation direction and polarization can be specified by the user. For the temporal dependency, either a harmonic or a Gaussian signal is applied. Both signals can be ramped up from zero to maximum amplitude over a given number of periods to reduce numeric instabilities.

EMW provides several excitation sources to model scattering problems. The basics are plane wave excitation and a spatial Gaussian beam that can be propagated at an angle from the defined excitation plane. The defined excitation plane arises from the total-field scattered-field (TFSF) formalism, described in [Total-Field Scattered-Field on page 52](#).

Additional excitation sources have been implemented to facilitate the modeling of experimental setups, such as that commonly used in the design of CMOS image sensors. These additional sources include an incoherent illumination source as well as an interface to input the complex vectorial beams from the Synopsys CODE V® tool.

Total-Field Scattered-Field

In EMW, all excitations have been implemented using the TFSF formulation. This excitation method is based on splitting the simulation space into a total-field zone and a scattered-field zone. The excitation source has an effect at the interface between the two zones such that the total field \mathbf{E}_T and the scattered field \mathbf{E}_S are related by:

$$\mathbf{E}_T = \mathbf{E}_S + \mathbf{E}_I \quad (22)$$

where \mathbf{E}_I denotes the excitation field (also called the incident field). Detailed descriptions of the TFSF technique can be found in the literature [\[1\]](#).

The TFSF excitation is used mainly for scattering problems, where a target inside the simulation space is impinged by an electromagnetic wave incident from the outside. In this

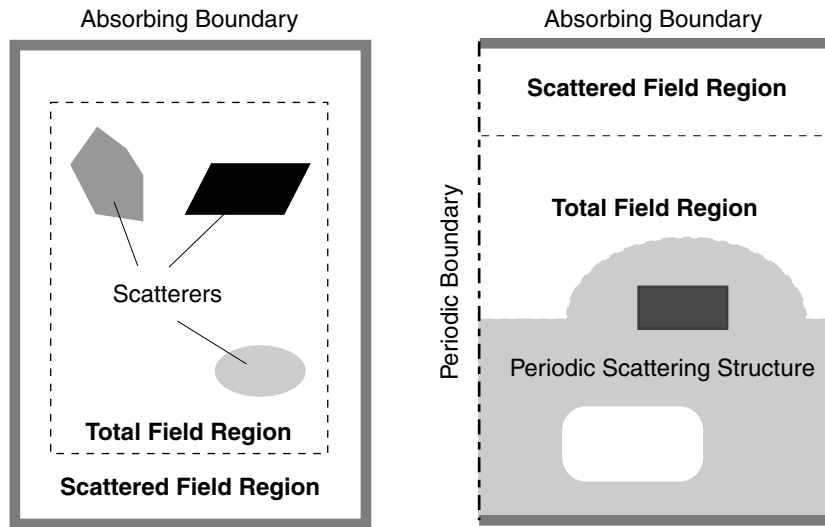
Chapter 6: Specifying Excitations

Specification of the Illumination Direction

case, the target is enclosed inside a total-field zone and the outer part of the simulation domain is the scattered-field zone.

In general, the geometric information associated with a TFSF excitation consists of a box that must be defined such that the scattering object is completely inside the box, that is, the total-field zone must be a closed volume. If periodic boundaries are present, a plane (in three dimensions) or a line (in two dimensions) is sufficient to separate the two regions. In this case, the total-field zone is extended to infinity in the lateral directions and can still be considered a closed volume.

Figure 4 TFSF boundary (dashed line) for (left) nonperiodic and (right) periodic structures



Specification of the Illumination Direction

For excitations based on the TFSF formalism such as plane wave or Gaussian beam, illumination from the top or the bottom of a device structure can be specified directly in the respective excitation section:

```
GaussianBeamExcitation | PlaneWaveExcitation |  
  TruncatedPlaneWaveExcitation {  
    Direction = fromTop | fromBottom  
    ...  
  }
```

Using the keyword `Direction` sets the parameters determining the propagation direction to normal incidence and uses the default values for all other parameters. In addition, the excitation plane (see [Specifying Excitation Regions or Boxes on page 54](#)) is set two tensor cells away from the device boundary according to the illumination direction. Any changes to the default values as a result of specifying `Direction` are reported in the log file and can be

Chapter 6: Specifying Excitations

Plane Waves

overwritten by explicitly specifying the respective keyword in the command file. For example, a typical use case is changing to oblique incidence instead of normal incidence by explicitly specifying `Theta` and `Phi`.

Note:

EMW recognizes the coordinate system (unified coordinate system versus DF–ISE coordinate system) of the input tensor mesh and interprets the top and bottom of the device structure accordingly.

The specification of other excitation parameters, including the ones that are set when using `Direction`, is described in the following sections.

Another advantage of specifying the illumination direction is that it allows EMW to create the various sensors necessary for computing reflection, transmission, and absorption without you having to specify those sensors explicitly. See [Reflection, Transmission, and Absorption Sensors on page 78](#) for details.

Plane Waves

Parameters specifying the plane wave excitation such as propagation direction and polarization are defined in the `PlaneWaveExcitation` or `TruncatedPlaneWaveExcitation` section of the command file.

Specifying Excitation Regions or Boxes

The TFSF boundary is most conveniently defined by specifying the excitation plane normal in the `PlaneWaveExcitation` section using the keyword `PlaneX`, `PlaneY`, or `PlaneZ`. Alternatively, you can specify the opposing corners of a box using the keywords `BoxCorner1` and `BoxCorner2`. This is useful for defining truncated plane waves. If periodic boundary conditions are applied to the sides, the specified box must define a plane in three dimensions or a line in two dimensions.

Note:

In all cases, it is recommended not to have material interfaces or simulation domain boundaries within a three-cell layer of a TFSF boundary for numeric reasons.

To specify a plane wave in the $z = 0.05 \text{ } \mu\text{m}$ plane, you can use:

```
PlaneWaveExcitation {  
    PlaneZ = 0.05          # um  
    ...  
}
```

Chapter 6: Specifying Excitations

Plane Waves

To specify a truncated plane wave in the $z = 0.05 \text{ } \mu\text{m}$ plane with a $0.6 \times 0.6 \text{ } \mu\text{m}^2$ window, you can use:

```
TruncatedPlaneWaveExcitation {  
    BoxCorner1 = {0.2, 0.2, 0.05}      # um  
    BoxCorner2 = {0.8, 0.8, 0.05}      # um  
    ...  
}
```

Due to the truncation, the plane-wave front diverges as it propagates away from the excitation plane. A common use of the truncated plane wave is to excite wave propagation in a waveguide. In this case, the truncated plane-wave excitation plane is placed near and parallel to the open facet of the waveguide. The area of the truncation window is set similar to the cross-sectional area of the open facet of the waveguide.

Specifying Direction and Polarization

Using ζ for the coordinate along the propagation direction, the plane wave is defined as:

$$E(\mathbf{r}, t) = s(\zeta - vt)\hat{\mathbf{p}} \quad (23)$$

$$\mathbf{H}(\mathbf{r}, t) = \sqrt{\frac{\epsilon}{\mu}} \hat{\mathbf{k}} \times \mathbf{E}(\mathbf{r}, t) \quad (24)$$

where:

- s is the signal.
- v is the phase velocity of the medium.
- $\hat{\mathbf{p}}$ is the unit vector specifying the polarization direction.
- $\hat{\mathbf{k}}$ is the unit wave vector of the plane wave.

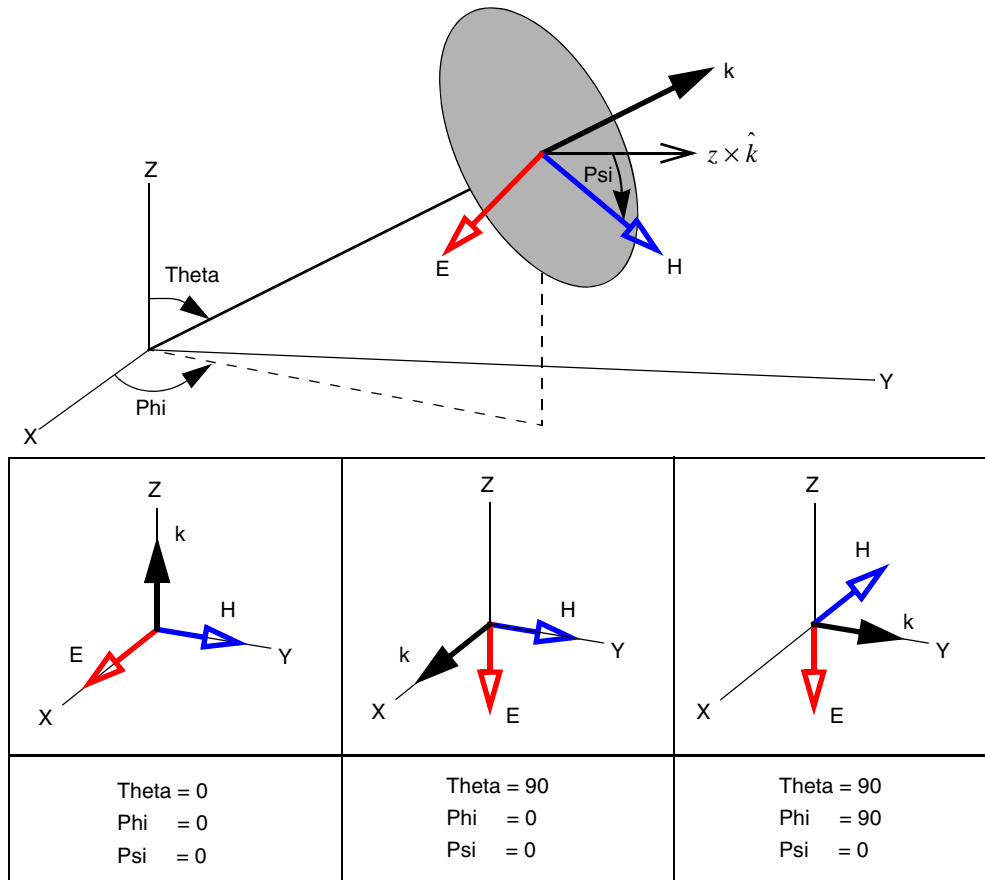
In EMW, the propagation direction is defined by the two parameters `Theta` and `Phi`, which denote the angles in degrees from the z -axis and x -axis, respectively.

A third parameter `Psi` is used to set the polarization angle, which measures the angle between \mathbf{H} and the $z \times \hat{\mathbf{k}}$ axis (see [Figure 5](#)). In this definition, $\text{Psi} = 90^\circ$ and $\text{Psi} = 270^\circ$ correspond to TE polarization; whereas, $\text{Psi} = 0^\circ$ and $\text{Psi} = 180^\circ$ correspond to TM polarization.

Chapter 6: Specifying Excitations

Plane Waves

Figure 5 Definition of coordinate system for 3D plane wave excitation and examples of parameters of 3D plane wave



The following example shows the command file specification of a typical time-harmonic plane-wave excitation in three dimensions:

```
PlaneWaveExcitation {
  Amplitude = 1      # [V/m]
  Wavelength = 500   # [nm]
  Theta = 30         # [deg]
  Phi = 45           # [deg]
  Psi = 0            # [deg]
  Signal = Harmonic
  NRise = 5
  PlaneZ = 3         # [um]
}
```

Depending on the dimension and the used coordinate system of the device, achieving a simple illumination from the top or the bottom requires specific settings for the location of the excitation plane and for the excitation angles. Using the keyword `Direction`, the correct

values for these settings are determined automatically. See [Specification of the Illumination Direction on page 53](#).

Two-dimensional simulations are performed by extruding the 2D structure by one layer in the third dimension (z-direction). For ease of use, an alternate definition scheme exists for this case. If `Polarization=TE` or `Polarization=TM` is set in the respective `PlaneWaveExcitation` section, then the plane wave is completely determined when the parameter `Theta` is specified (see [Figure 6](#)).

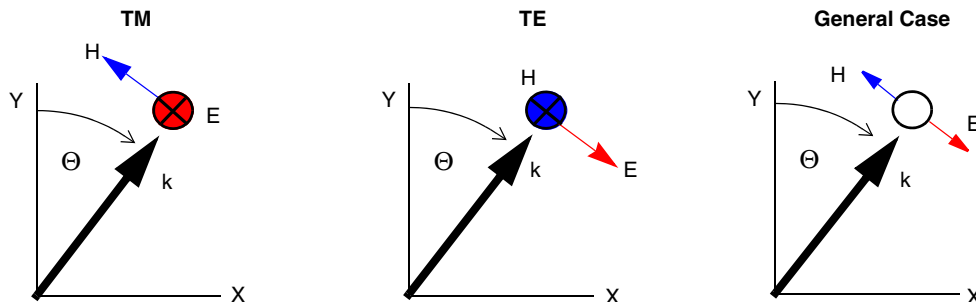
For the general polarization case, that is, with neither `TE` nor `TM` selected, an additional angle `Psi` is defined in the same way as in the full 3D case, with `Psi = 0°` as default. In addition, you can specify a real value in the interval $[0,1]$ for `Polarization`, where `Polarization=0` refers to TM excitations, and `Polarization=1` refers to TE excitations.

A simple relationship between `Psi` and `Polarization` can be established: $\text{Polarization} = \sin^2(\text{Psi})$.

Note:

This plane wave representation is valid only for homogeneous and loss-free domains, which means that all field components belonging to the excitation region should have identical material properties.

Figure 6 Definition of angle of incidence of 2D plane wave, where TE and TM follow Taflove and Hagness definition (see following note): (left to right) $\text{Psi}=0^\circ$, 90° , and 45°



Note:

TE and TM refer to the definition used by Taflove and Hagness [\[1\]](#), that is, TM (transverse magnetic) defines the magnetic field perpendicular to the z-direction.

Specifying Elliptical Polarization

By default, the plane wave excitation is linearly polarized, that is, the E-field and H-field vectors always maintain fixed directions as the incident wave propagates. You can also simulate plane waves that are polarized elliptically by defining the polarization parameters

Chapter 6: Specifying Excitations

Spatially Incoherent Illumination

EllipticalPolarizationMagnitude and EllipticalPolarizationPhase in the PlaneWaveExcitation section.

For simplicity, consider a linearly polarized plane wave traveling in the z-direction, $\vec{E}(t) = E_0 x \cos(\omega t - kz)$. With the polarization parameters defined, the plane wave reads as:

$$\vec{E}(t) = E_0 \frac{\hat{x} \cos(\omega t - kz) + \rho \hat{y} \cos\left(\omega t - kz + \frac{\pi}{180} \delta\right)}{\sqrt{1 + \rho^2}} \quad (25)$$

where:

- ω is the angular frequency.
- k is the propagation constant.
- ρ is the ratio of the polarization magnitude of the major and minor axes.
- δ is the phase offset between the major and minor axes.

Therefore, the direction of the E-field can change as the wave propagates in space or as the time varies. When $\rho = 1^\circ$ and $\delta = 90^\circ$, the plane wave is a circularly polarized wave.

An example of specifying a circularly polarized plane wave in the command file is:

```
PlaneWaveExcitation {  
    ...  
    Theta = 180  
    Phi = 0  
    Psi = 0  
    EllipticalPolarizationMagnitude = 1.0  
    EllipticalPolarizationPhase = 90  
}
```

Spatially Incoherent Illumination

To model an incoherent illumination due to a source of finite extent, you must sum the fields contributed by plane waves incident from different directions [3].

Assuming the light passes through a circular pupil and a circular objective lens before reaching the structure on the focal plane, the incident angles are limited by the half-extended angle of the objective:

$$\theta_{\text{obj}} = \text{asin}(\sigma \text{NA}), \text{ for } \sigma \text{NA} < 1 \quad (26)$$

where NA is the numerical aperture of the objective lens. For circular source and pupils, the partial coherence factor σ is defined as the ratio of the source image dimension to the pupil size [4].

Chapter 6: Specifying Excitations

Gaussian Beam or Gaussian Field Distribution

Since the incident angles are confined in a cone, you can synthesize the spatially incoherent illumination by a finite number of plane waves with different incident angles. The greater the number of plane waves, the more accurately the incoherent illumination can be represented. However, each incident angle corresponds to one FDTD simulation. Therefore, you might want to limit the number of plane waves to obtain satisfactory results within a reasonable simulation time.

An example of specifying a spatially incoherent illumination in the command file is:

```
PlaneWaveExcitation {  
    ...  
    PartialCoherenceFactor = 1.0 # ratio of source image dimension  
                                # to pupil size  
    NumericalAperture = 0.5     # half-extended angle of objective  
                                # lens = 30 deg  
    NumberOfPlaneWaves = 4      # number of plane waves for incoherent  
                                # excitation  
}
```

Gaussian Beam or Gaussian Field Distribution

For applications involving converging or diverging light beams, such as light passing through a lens, it is helpful to use spatially Gaussian-distributed incident fields on the TFSF excitation plane. EMW adopts the formulation in [2] for the implementation of the Gaussian beam (see [Figure 7 on page 61](#)). Electromagnetic fields at the TFSF excitation plane are calculated by the scalar paraxial solution of a 3D fundamental-mode Gaussian beam:

$$E(r, z') = E_{\max} \frac{w_0}{w(z')} \exp\left[\frac{-r^2}{w(z')^2}\right] \exp\left[-ikz' - ik\frac{-r^2}{2R(z')} + i\xi(z')\right] \quad (27)$$

Following the procedure in [2] and solving the scalar wave equation in two dimensions yield the beam profile in two dimensions:

$$E(r, z') = E_{\max} \sqrt{\frac{w_0}{w(z')}} \exp\left[\frac{-r^2}{w(z')^2}\right] \exp\left[-ikz' - ik\frac{-r^2}{2R(z')} + i\frac{\xi(z')}{2}\right] \quad (28)$$

In [Equation 27](#) and [Equation 28](#):

- r is the radial distance from the center axis of the beam.
- z' is the axial distance from the beam waist.
- i is the imaginary unit (for which $i^2 = -1$).
- $k = 2\pi/\lambda$ is the wave number.
- w_0 is the radius of the beam waist.
- $z'_R = \pi w_0^2/\lambda$ is the Rayleigh length.

Chapter 6: Specifying Excitations

Gaussian Beam or Gaussian Field Distribution

- $w(z') = w_0 \sqrt{1 + (z'/z'_R)^2}$ is the radius at which the field amplitude and the intensity drop to $1/e$ and $1/e^2$ of their axial values, respectively.
- $R(z') = z' [1 + (z'_R/z')^2]$ is the radius of curvature of the wave fronts of the beam.
- $\xi(z') = \text{atan}(z'/z'_R)$ is the Gouy phase shift, an extra contribution to the phase that is seen in Gaussian beams.

The peak electric field E_{\max} can be set directly using the keyword `Amplitude` or can be calculated from the `Intensity` (I_{\max}). Alternatively, the total power P_{\max} of the beam can be set by `BeamPower`. In three dimensions, $E_{\max} = 2 \sqrt{Z_0 P_{\max} / \pi n} / w_0$. In two dimensions, $E_{\max} = (2 \sqrt{2} / \pi Z_0 P_{\max} / n L_z w_0)^{1/2}$. Here, Z_0 and n denote, respectively, the vacuum wave impedance and the refractive index of the medium at the excitation plane. L_z is the dimensionality constant along the third dimension in the 2D case, for which the default value is 1 μm . The box corners (P_1, P_2 set by `BoxCorner1, BoxCorner2`) define the truncation window of the Gaussian beam. For oblique incidence, when the axial direction of the beam is not perpendicular to the excitation plane, the Gaussian beam is projected onto the excitation plane.

Since the implementation of Gaussian beams is based on a scalar paraxial solution, the beam quality or, from another perspective, the deviation of the beam spatial profile from the Gaussian is determined by how well the paraxial approximation ($w_0 > \lambda$) is satisfied. In general, a smaller beam waist (tighter focus), a focal point farther away from the excitation plane, and a more tilted beam lead to a less accurate Gaussian beam profile. When the excitation condition deviates from that enforced by the paraxial approximation, enlarging the excitation plane or increasing the mesh density only leads to a marginal improvement of the results.

Due to the large phase difference in the wave front at the excitation plane, users must specify a slow ramp-up of the field to a harmonic signal.

Note:

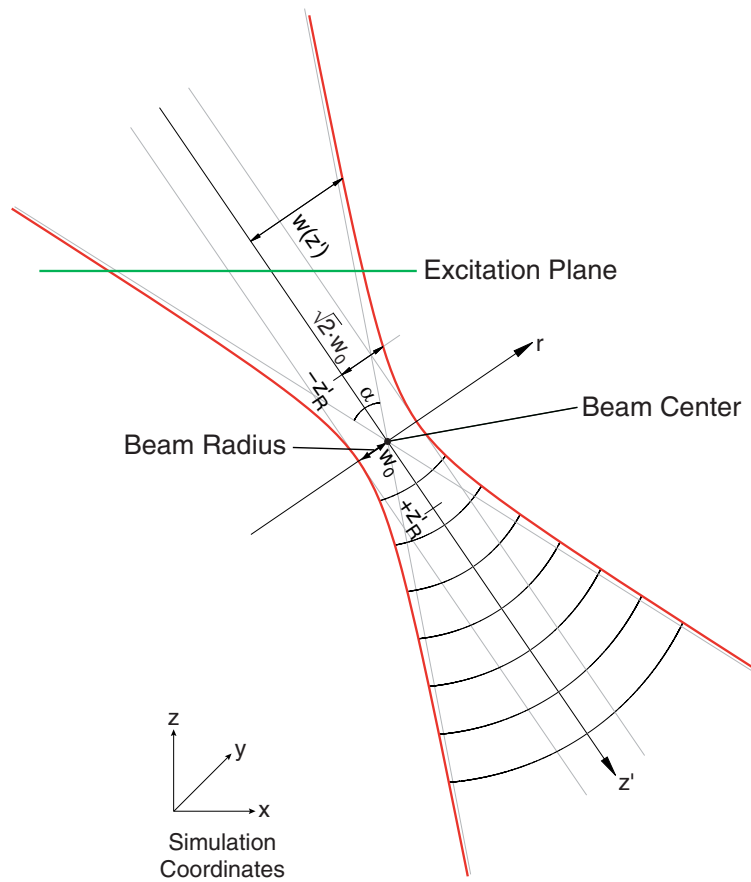
Gaussian beam excitation is available only in the z-direction. In addition, temporal dependency (`Signal=Gauss`) is not available for Gaussian beam excitation.

For example, to specify a truncated, oblique, Gaussian beam on the $z = 0.05 \mu\text{m}$ plane (defined by box corners) within a $2 \times 2 \mu\text{m}^2$ window and with a peak intensity of 0.1 W/cm^2 , the following syntax applies:

```
GaussianBeamExcitation {
  PlaneZ = 0.05
  BeamCenter = {0.3,0.0,-0.4}
  BeamRadius = 0.10          # um
  Theta = 135
  Phi = 0
  ...
  Intensity = 0.1            # in W/cm^2
}
```

Temporal Dependency

Figure 7 *Schematic of Gaussian beam excitation*



Temporal Dependency

Signals are used to define the time dependency of the incident field in the respective excitation section. By default, a ramped harmonic signal of the following form is used:

$$s(t) = \begin{cases} A \left\{ 1 - \exp \left[\frac{-(t-t_0)^2}{\tau^2} \right] \right\} \sin \left(\frac{2\pi c_0}{\lambda} t + \varphi \right) & \text{for } t > t_0 \\ 0 & \text{otherwise} \end{cases} \quad (29)$$

where:

- $\tau = \frac{N_\tau \lambda}{2c_0}$ determines the rise of the signal from zero to its maximum amplitude.
- λ denotes the wavelength.
- φ denotes the phase.

Chapter 6: Specifying Excitations

Temporal Dependency

- The parameter t_0 specifies a delay for the ramping of the signal.
- N_τ is the number of signal periods before the full amplitude is reached (see [Table 3](#) for the corresponding keyword).
- c_0 is the vacuum speed of light.

Instead of amplitude A , you can specify intensity I and, instead of wavelength λ , you can specify frequency f , according to the following conversion formulas:

$$\begin{aligned} A &= \sqrt{2ZI} \\ \lambda &= \frac{c_0}{f} \end{aligned} \tag{30}$$

where Z is the impedance of the excitation medium.

Specifying `Signal=Gauss` selects a Gaussian signal modulated by a sinusoidal wave as time excitation of the form:

$$s(t) = A \exp\left[\frac{-(t-t_0)^2}{2\sigma^2}\right] \sin\left(\frac{2\pi c_0}{\lambda}t + \varphi\right) \tag{31}$$

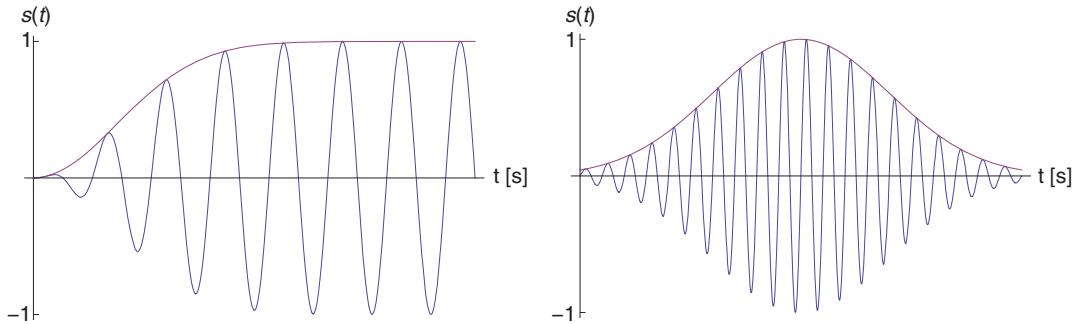
The width of the Gaussian envelope is controlled by $\sigma = \frac{N_\sigma \lambda}{c_0}$. For all other keywords, see [Table 3](#).

Table 3 Main keywords to control temporal signals

Symbol	Keyword	Unit
A	Amplitude	V/m
N_τ	Nrise	Number of periods
N_σ	Sigma	Number of periods
λ	Wavelength	Nm
φ	Phase	degree
t_0	Delay	Number of periods
f	Frequency	Hz
I	Intensity	W/cm ²

Figure 8 illustrates the harmonic and Gaussian signal shapes.

Figure 8 Examples of (left) harmonic signal and (right) Gaussian signal for $A = 1$, $\lambda = 0.6 \mu\text{m}$, $N_s = N_t = 4$



CODE V Interface

EMW uses the `CodeVExcitation` section to specify the excitation type that inputs complex vectorial field data from the CODE V tool. The equivalence theorem is used to set up such an excitation on the excitation plane using the TFSF formalism.

The CODE V tool is used to design and optimize a complex lens assembly. The interface between the CODE V tool and EMW provides a seamless flow of optical simulation and design of CMOS image sensor devices.

The syntax to specify CODE V beam excitation is:

```
CodeVExcitation {  
    InputDataFile = {"zf.dat", "zfPlus20.dat"}  
    AnchorPoint1 = {-1, -1, 0.28} # [um]  
    AnchorPoint2 = {-1, -1, 0.30} # [um]  
    Wavelength = 300 # [nm]  
    Signal = Harmonic  
    NRise = 20  
    Delay = 0  
    AmplitudeScaling = 1  
    PhaseShift = 0  
}
```

where:

- `InputDataFile` specifies two slices of complex electric-field data that are output from the CODE V tool. The beam is assumed to propagate from the first slice to the second slice.
- `AnchorPoint1` and `AnchorPoint2` specify the physical coordinates to position the two slices of data in EMW.

Chapter 6: Specifying Excitations

CODE V Interface

- As the absolute amplitude and the phase of the complex electric field are provided in the data files, `AmplitudeScaling` and `PhaseShift` are used to specify a relative amplitude scaling and phase shift.

For a summary of keywords supported in the `CodeVExcitation` section, see [Appendix A on page 108](#).

Typically, the content of the `zf.dat` file is as follows:

```
Wavelength:      300      nm
Grid spacing:    0.000020      0.000020      mm
Array size:      101      101
...
<field data>
...
```

Table 4 Keywords in the CODE V data file (`zf.dat`)

Keyword	Description	Unit
Wavelength	Wavelength of the excitation	nm
Grid spacing	Grid spacing in the x- and y-directions	mm
Array size	Number of electric field values in the x- and y-directions	1

The field data part has the following format:

```
ExRe(x1,y1)  ExIm(x1,y1)  EyRe(x1,y1)  EyIm(x1,y1)  EzRe(x1,y1)
EzIm(x1,y1)  ExRe(x2,y1)  ExIm(x2,y1)  EyRe(x2,y1)  EyIm(x2,y1)
EzRe(x2,y1)  EzIm(x2,y1)  ...          ExRe(xn,y1)  ExIm(xn,y1)
EyRe(xn,y1)  EyIm(xn,y1)  EzRe(xn,y1)  EzIm(xn,y1)

...

ExRe(x1,yn)  ExIm(x1,yn)  EyRe(x1,yn)  EyIm(x1,yn)  EzRe(x1,yn)
EzIm(x1,yn)  ExRe(x2,yn)  ExIm(x2,yn)  EyRe(x2,yn)  EyIm(x2,yn)
EzRe(x2,yn)  EzIm(x2,yn)  ...          ExRe(xn,yn)  ExIm(xn,yn)
EyRe(xn,yn)  EyIm(xn,yn)  EzRe(xn,yn)  EzIm(xn,yn)
```

The order of the x- and y-indices is:

```
x1 < x2 < ... < xn
y1 > y2 > ... > yn
```

where $(x1, yn)$ correspond to the indices of the anchor point of the field data.

Chapter 6: Specifying Excitations

References

The limitations of CODE V beam excitation are:

- Beams in two dimensions are not supported.
- The beam propagation direction must be in the z-direction. The two slices of data must be taken in the xy plane.
- The two slices of data must be taken in vacuum or air ($\epsilon_r = 1$).
- To obtain the correct beam propagation, the two slices of data must vanish before reaching the xy boundaries. If this is not satisfied, spurious diffraction will emerge.
- The x (y) coordinates of `AnchorPoint1` and `AnchorPoint2` must be the same. Their z-coordinate difference must be the same as the EMW z-cell size between slices corresponding to `AnchorPoint1` and `AnchorPoint2`.
- Only harmonic signal is allowed in the `CodeVExcitation` section.
- The `CodeVExcitation` section can be used only with the boundary conditions in [Table 5](#).

Table 5 Allowed boundary conditions for CodeVExcitation section

x_{min}	x_{max}	y_{min}	y_{max}	z_{min}	z_{max}
CPML	CPML	CPML	CPML	CPML	CPML
Periodic	Periodic	Periodic	Periodic	CPML	CPML

References

- [1] A. Taflove and S. C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Boston: Artech House, 3rd ed., 2005.
- [2] H. Kogelnik and T. Li, "Laser Beams and Resonators," *Applied Optics*, vol. 5, no. 10, pp. 1550–1567, 1966.
- [3] J. W. Goodman, *Introduction to Fourier Optics*, Englewood, Colorado: Roberts & Company, 3rd ed., 2005.
- [4] A. K.-K. Wong, *Optical Imaging in Projection Microlithography*, Tutorial Texts in Optical Engineering, vol. TT66, Bellingham, Washington: SPIE Press, 2005.

7

Terminating Simulations

This chapter discusses issues pertaining to terminating simulations.

Options for Terminating FDTD Time-Stepping Loop

There are different options for terminating the FDTD time-stepping loop:

- Setting the total simulation time directly.

This option is controlled by the keyword `TotalSimulationTime` or `TotalTimeSteps` in the `Globals` section, which allows you to determine the simulation time either directly or in terms of the total number of time steps, respectively. However, the final state of the simulation is unclear.

- Using detectors to define the termination criteria.

Applying a detector allows you to terminate the computation as soon as a certain state (such as steady state) or situation is reached. This is especially important when quantities are computed that require the instantaneous electromagnetic field distribution to be time harmonic within part or all of the simulation domain (see [Extractors on page 75](#) and [Sensors on page 77](#)).

- Sending the POSIX interrupt signal INT.

Sending the POSIX interrupt signal INT to the running process terminates the simulation. However, before quitting the process, any pending plots, monitors, extractors, or sensors scheduled for the regular completion of the simulation will be performed unless a second signal INT is received by the process. In that case, the simulation terminates immediately. A signal can be sent to a process by invoking the `kill` command. For example:

```
kill -s INT <pid>
```

where `<pid>` is the process ID, which can be extracted from the `.log` file. In an interactive shell, use the shortcut key combination Ctrl+C to send the signal INT to the active process.

Chapter 7: Terminating Simulations

Detectors

These options for terminating a simulation are nonexclusive. For example, the first option can be used to limit the total simulation time for a simulation that shows poor or no convergence with respect to the termination criterion defined by the detector. Without such a limit, a simulation could progress indefinitely.

Detectors

A detector works on a specified domain, which can be defined by a region of the tensor mesh or by specifying a box using the keywords `BoxCorner1` and `BoxCorner2` (see [Chapter 8 on page 70](#)). If nothing is specified, the detector is applied to the entire simulation domain.

For the most common case of `periodicoblique` boundary conditions, the detector tracks the total energy in the system at each detection period. In all other cases, a more stringent criterion is used, which checks the value of each electric field component at each detection period. However, this latter criterion tends to show noisy behavior especially for structures with high internal scattering.

By default, the period is derived from the wavelength of the excitation signal. You can explicitly select the period using the keyword `Frequency`. In both cases, the simulation time step is adjusted (reduced) automatically to an integer fraction of the excitation signal period.

By default, the termination criterion considers all grid nodes within the detector domain. When using the tracking criterion for electric field components, you can further reduce the number of evaluations for the corresponding coordinate direction by providing a skip factor using the keywords `StepX`, `StepY`, and `StepZ`. For example, choosing a skip factor of 20 instead of 10 (default) for each coordinate direction reduces the computation complexity of the detector by approximately a factor of eight.

The termination criterion depends on the time-domain excitation signal in the `Excitation` section and the type of the detector. Different cases are shown in [Table 6](#).

Table 6 *Detector termination criterion for different time-domain signal and detector types*

Signal	Type	Convergence criterion
Harmonic	Periodic	For simulations using <code>periodicoblique</code> boundary conditions, the total energy in the system is tracked, and its relative change is compared to the value specified by the keyword <code>Tolerance</code> . For all other simulations, each electrical field component is compared to the last detection period. The difference divided by the maximum field value in the simulation domain must be less than the value specified by <code>Tolerance</code> .
Harmonic	Decayed	Not allowed.

Chapter 7: Terminating Simulations

Detectors

Table 6 *Detector termination criterion for different time-domain signal and detector types*

Signal	Type	Convergence criterion
Gauss	Periodic	For each electrical field component, its maximum field value at the current detection period divided by its maximum field value, at all previous detection periods, is less than the value specified by the keyword <code>Tolerance</code> .
Gauss	Decayed	For each electrical field component, its maximum field value at the current detection period divided by the amplitude of the excitation Gaussian time pulse is less than the value specified by the keyword <code>Tolerance</code> .

Note:

Detectors can slow down a simulation considerably. This is especially critical if a large detector domain with small step parameters is used, for example, the entire simulation domain for a large tensor mesh in terms of the number of grid cells.

A detector also can terminate simulations that diverge or run into numeric problems due to, for example, a mesh that is too coarse leading to nonfinite field (NaN) values. If the keyword `BreakOnError` is set to `Yes`, the simulation stops immediately; however, the final `Plot` or `Extractor` sections are still processed. The exit code of such simulations is set to a nonzero value so that a corresponding node in Sentaurus Workbench is flagged as failed. Similarly, the keyword `FailureOnUnreachedTolerance` controls the exit code of a simulation that has not fallen below the value of `Tolerance` specified in the command file.

The keywords `StartTime` and `StartTick` allow you to restrict the time interval during which the detector is actually applied. Since the evaluation of the termination criterion for all field components might slow down the simulation substantially, it can be beneficial to delay the application of the detector. Therefore, `StartTick` is offset to a reasonable value automatically, according to the excitation signal and the `Tolerance`. To overwrite the automatic value, specify `StartTick` explicitly.

Note:

If the MPI parallelization is activated, the detector must be applied to the entire simulation domain to prevent slowing down the MPI speedup.

In the course of the simulation, the `Detector` section writes the maximum deviation of the entire `Detector` domain and its location therein to standard output and the log file. How the maximum deviation is computed depends on the excitation signal and the detector type, which is described in [Table 6](#). Sometimes, it can be useful to monitor the convergence behavior visually to have a better impression of the current state of the simulation and to decide whether an increased number of time steps would significantly reduce the maximum deviation and, therefore, lead to more accurate results.

Chapter 7: Terminating Simulations

Detectors

For this purpose, the maximum deviation can be plotted over time or over the number of time steps (denoted as `TickStep`). The data is saved continuously in the `ResultFile` specified in the `Globals` section of the command file and can be plotted in Sentaurus Visual. Clicking the **Reload** button in Sentaurus Visual during the simulation adds any new data points computed in the intervening time to the existing curve.

For simulations where each electric field component is compared to the last detection period, the location of the maximum deviation is plotted as well. It might not be unique because it is possible that several vertices exhibit the same deviation. In parallelized simulations (shared-memory parallelization (SMP) and distributed processing (DP)), the reported location of the maximum deviation can vary for different runs of the same simulation.

Note:

In DP simulations, the maximum location reported can lie outside the physical device structure in one of the CPML regions.

8

Extracting and Visualizing Results

This chapter presents how to extract useful quantities from EMW simulation results.

Overview

EMW offers several ways of extracting and visualizing its simulation results. Besides plotting spatially resolved data such as electrical field values and material parameters that were used in the FDTD formulation, integrated data derived from it can be generated. Material parameters such as the complex refractive index can be output using plots. Field values such as the electric field can be accessed using plots or monitors. Derived physical quantities such as the absorbed photon density are computed by extractors or sensors. While extractors calculate derived physical quantities in a given geometric domain, the output of sensors is usually the result of integrating a specific quantity over a user-defined surface or volume. Whether the integration domain is a surface or a volume depends on the quantity of interest.

A major difference between plots, monitors, and sensors on the one hand and extractors on the other hand is that the former can generate output at any time step during the simulation, while the latter is restricted to the end of the simulation. If a detector has been specified and the given tolerance has not been reached before the application of extractors or sensors, a warning message is issued. Certain quantities computed by extractors and sensors rely on a quasistationary solution of the FDTD equations to produce physically meaningful results.

A common feature of plots, extractors, and sensors is the selection of an applicable 2D or 3D domain by specifying one of the following options:

- One or more regions of the corresponding grid
- One or more materials of the corresponding grid (applies only to sensors)
- A box with the coordinates of two opposing corners using the keywords `BoxCorner1` and `BoxCorner2`
- A plane (a line in two dimensions) normal to one of the coordinate axes using the keywords `PlaneX`, `PlaneY`, and `PlaneZ`

Note:

Only sensors support the specification of multiple domains (see [Using Multiple Domains in a Sensor on page 80](#) for details). However, regions and materials as well as a box and a plane are mutually exclusive.

Monitors are used to probe field values at specific points in the grid defined by their coordinates. For plots and extractors, data can be output to either a tensor grid or a mixed-element grid. On the other hand, the results of monitors and sensors are written to a `.plt` file, similar to the current plot file in Sentaurus Device (see *Sentaurus™ Device User Guide*, Current File). The quantities of interest are specified using the keyword `Quantity`.

Note:

The data format of the grid file specified in the `Globals` section determines the data format to be used for plots and extractors.

You can control further compression, beyond a basic standard compression for plot, sensor, extractor, and save files as well as the result file, by specifying `CompressTDR=Yes | No` in the `Globals` section.

Extractors and sensors can be used in either of two ways:

- They are applied during (sensors only) or at the end of the simulation after the FDTD time-stepping loop.
- They are applied to previously saved results, which are loaded for postprocessing purposes.

The advantage of the latter use is that it is not always clear, at the beginning of a simulation, which subdomains are of interest. Using the postprocessing feature, you can easily extend or modify the initial sets of extractors and sensors to further investigate the simulation results without having to rerun the entire FDTD simulation, which is generally time consuming.

Plots

Field values and material parameters can be output to either a tensor grid or a mixed-element grid. In both cases, you can output the data to either the entire grid or only a subdomain. Subdomains can be specified directly by providing the coordinates of the two opposing corners of a box. The corners with the minimum and maximum values for all coordinate axes are denoted by `BoxCorner1` and `BoxCorner2`, respectively. Alternatively, if the requested subdomain is a line in two dimensions or a plane in three dimensions, you can use a short form and define a line or plane perpendicular to one of the coordinate axes denoted by the keywords `PlaneX`, `PlaneY`, and `PlaneZ`. A plane is defined by a coordinate value for the axis to which it is perpendicular. In addition to specifying explicit coordinate

Chapter 8: Extracting and Visualizing Results

Plots

values, you can specify `min` or `max`, which is interpreted as the minimum or maximum device extent along the corresponding coordinate axis.

Using the keyword `Region` in the `Plot` section is another option for specifying a subdomain. When outputting in tensor-grid data format, only a single region of the input tensor-grid file can be specified.

If output is requested on a mixed-element mesh, you can select several regions of the mixed-element grid file for plotting.

The exact meaning of the box corner and region depends on the grid type of the plot. For tensor plots, box corner and region refer to the input tensor grid. If the specified coordinates of a box corner do not coincide with a specific vertex of the tensor grid, the nearest vertex will be chosen instead. The dimension of the subdomain can be lower than that of the input tensor grid. Contrary to mixed-element plots, the resulting plot file only contains the specified subdomain. A typical example is a two-dimensional cut through a three-dimensional device structure.

For large (in terms of cell number) tensor meshes, visualization can be challenging, and plotting only the surface data might be a better option. In 3D device structures, the keyword `Domain` allows you to switch between volume and surface plots. For the latter, you can select explicitly certain faces of the domain using the keyword `Sides`.

For tensor plots, specifying the keyword `Region` in `Quantity` enables the visualization of the region information with the simulation results. For example:

```
Quantity = { AbsElectricField, AbsMagneticField, Region }
```

Note:

The keyword `Region` can appear as a command in the `Plot` or `Extractor` section or as a keyword in the list of `Quantity`. The former restricts the plotting area to a region, and the latter instructs EMW to include region information in the plot file.

Since a 2D tensor plot is supported only in xy format, when making 2D cuts from a 3D simulation, the coordinate systems are transformed using the rules in [Table 7](#).

Table 7 Coordinate system transformation rule for 2D cuts from a 3D simulation

Cut	Transformation rule
x-cut	Y (3D) -> X (2D)
	Z (3D) -> Y (2D)
	X (3D) -> Z (2D)

Table 7 *Coordinate system transformation rule for 2D cuts from a 3D simulation*

Cut	Transformation rule
y-cut	Z (3D) -> X (2D)
	X (3D) -> Y (2D)
	Y (3D) -> Z (2D)
z-cut	X (3D) -> X (2D)
	Y (3D) -> Y (2D)
	Z (3D) -> Z (2D)

Note:

Only the spatial coordinates follow this rule: The sub-indices of a vector quantity are unchanged when making a 2D cut. For example, the field components `AbsElectricFieldX`, `AbsElectricFieldY`, and `AbsElectricFieldZ` in the 3D plot are unchanged in the 2D x-cut, y-cut, and z-cut.

When `Region` is specified in `Quantity`, `Domain=Surface` is disallowed for visualizing 2D surface plots. To obtain 2D plots with region information in 2D simulations, `Region` in `Quantity` and `Domain=Volume` are specified in the `Plot` or `Extractor` section. To obtain 2D plots with region information in 3D simulations, specify `Region` in `Quantity`, `Domain=Volume`, and a boxed 2D region in the `Plot` or `Extractor` section.

Plotting the results on an arbitrary mixed-element mesh is triggered by specifying the corresponding grid file in the `Plot` section, using the keyword `GridFile`. For mixed-element plots, box corner and region refer to the grid file of the respective `Plot` section; whereas, specifying a plane is not supported. If a subdomain has been specified, the plot quantities are interpolated between the input tensor grid and the mixed-element grid for overlapping vertices. The values of the remaining vertices are set to zero.

Another group of keywords, which apply to both tensor and mixed-element plots, controls when data is plotted. A time interval during which plotting is active can be specified by either the absolute time (in seconds) or the number of time steps. The corresponding keywords are `StartTime`, `EndTime` or `StartTick`, `EndTick`. How often a plot file is generated is determined by the value of the keyword `TickStep`. If neither limit of the interval is specified, plotting is disabled and only a final plot at the end of the simulation is created. Whether a final plot is generated is controlled by the keyword `FinalPlot` whose default value is `yes`.

The file name for the plots is composed of the `Name` of the `Plot` section, the tool specific file suffix `.eml`, and a running number if more than one plot file is written.

For a list of supported plot quantities, see [Plot on page 131](#).

Chapter 8: Extracting and Visualizing Results

Monitors

The following example shows two typical tensor `Plot` sections and a mixed-element `Plot` section:

```
Plot {
  Name = "box"
  Quantity = {AbsElectricField, AbsMagneticField,
              ElectricConductivityXX}
  BoxCorner1 = {min, min, 0}
  BoxCorner2 = {max, max, 4.5}
  StartTick = 500
  EndTick = 10000
  TickStep = 100
  FinalPlot = no
}

Plot {
  Name = "X-Normal-Cut"
  Quantity = {AbsElectricField, AbsMagneticField,
              ElectricConductivityXX}
  PlaneX = 1
  StartTick = 500
  EndTick = 10000
  TickStep = 100
  FinalPlot = no
}

Plot {
  Name = "mixed-element-plot"
  Quantity = {AbsElectricField, AbsMagneticField, Region}
  GridFile = "mixed-element-grid.tdr"
  Region = {"substrate", "center-box"}
}
```

Monitors

Monitors are used to probe field values at specific tensor-grid points in the device structure. Similar to plots and sensors, it is possible to control when fields are probed during the simulation using the keywords `StartTick/Time`, `EndTick/Time`, and `TickStep/Frequency`. Monitors can also be used to observe the convergence behavior at certain locations in the device. The computational cost of monitors is much lower than that of plots because, typically, only a limited number of points are probed. The probed field values along with the coordinates of their corresponding location are written to the `ResultFile` specified in the `Globals` section.

The locations to monitor are specified as a list of coordinate vectors as shown in the following example:

```
Monitor {
  Name = "monitor"
  Location = { {0.5, 2, 0.1}, {0.4, 3, 0.5} } # [um]
```

Chapter 8: Extracting and Visualizing Results

Extractors

```
Quantity = { AbsElectricField, CplxEMField }  
TickStep = 100  
}
```

The quantity `CplxEMField` is an alias for the real and imaginary parts of all electromagnetic field components. However, it is also possible to list only specific field components such as `RealElectricFieldX`. For a complete list of supported quantities, see [Monitor on page 126](#).

Extractors

Extractors compute the quasi-monochromatic power flux density, the absorbed photon density, and the charge carrier generation rates assuming a quantum yield of 1 in absorbing semiconductor or metal media. A time-harmonic excitation signal is a prerequisite for the application of extractors. Therefore, extractors are only valid with a time-harmonic excitation signal and a steady-state solution.

Note:

The simulation must be performed until a time-harmonic state is reached with sufficient accuracy (a detector can be used for this purpose; see [Chapter 7 on page 66](#)) before the specified extractor quantities are computed.

The absorbed photon density and carrier generation rate are computed from the intensity distribution and the electromagnetic material parameters as outlined here.

Output data is computed on the native FDTD tensor grid and can be written either to a tensor-grid file or a mixed-element grid file using linear finite-element interpolation. In both cases, it is possible to extract the required quantities only on a subdomain of the entire grid by using the keywords `Region`, `or BoxCorner1`, `BoxCorner2`, `or PlaneX`, `PlaneY`, `PlaneZ` as described in [Plots on page 71](#). Unless a mixed-element grid file is given as `GridFile` in the `Extractor` section, output is generated in the tensor-grid format with the same discretization as the input tensor-grid file specified in the `Globals` section.

Note:

Specifying a tensor-grid file using `GridFile` is not supported.

When extracting quantities to a mixed-element grid, the quantities must be interpolated from the native FDTD tensor grid. The interpolation can be controlled with the keyword `Interpolation`. The options `Standard` and `Simple` stand for different variants of bilinear interpolation; whereas, `Conservative` employs a special algorithm [1] that conserves the spatial integral of the interpolated quantity.

The following assumptions are made in the computation of the absorbed photon density and the carrier generation rate: When light propagates in an absorbing medium with complex

Chapter 8: Extracting and Visualizing Results

Extractors

refractive index $n = n' + in'' = \sqrt{\epsilon}$, the corresponding \mathbf{k} -vector also becomes complex, being given by:

$$\mathbf{k}^2 = n^2 \mathbf{k}_0^2 = (n' + in'')^2 \mathbf{k}_0^2 = (\epsilon' + i\epsilon'') \mathbf{k}_0^2 \quad (32)$$

where \mathbf{k}_0 is the vacuum wave vector.

To model this behavior in EMW, a nonzero conductivity σ must be assigned to that medium. Since you also have:

$$\mathbf{k}^2 = \omega^2 \epsilon \mu \left(1 + i \frac{\sigma}{\omega \epsilon}\right) = \mathbf{k}_0^2 \mu_r \left(\epsilon_r + i \frac{\sigma}{\omega \epsilon_0}\right) \quad (33)$$

the complex refractive index is correctly modeled if you set:

$$\sigma = \frac{2\pi c_0 \epsilon_0 \epsilon''}{\lambda_0 \mu_r} \quad (34)$$

where λ_0 is the vacuum wavelength and c_0 is the vacuum speed of light. In the microscopic world, this can correspond to a *real* conductivity caused by free charge carriers (in metals) or to a conductivity caused by material polarization.

The absorbed power density is derived from the time-averaged Poynting vector:

$$S_{av} = \frac{1}{2} \Re(\mathbf{E} \times \bar{\mathbf{H}}) \quad (35)$$

where the complex quantities are found from the real field values at two different times under the time-harmonic state assumption. Note that the intensity distribution corresponds to the magnitude of the Poynting vector $|S_{av}|$. The absorbed power density at each point is computed as:

$$W = -\nabla \cdot \mathbf{S}_{av} = \frac{1}{2} \sigma |\mathbf{E}|^2 \quad (36)$$

With the photon energy given by $E_{ph} = h\nu$, the optical carrier generation rate G_{opt} can be computed as:

$$G_{opt} = \eta \frac{W}{E_{ph}} \quad (37)$$

where η is the quantum yield, which gives the average number of charge carriers generated by a single photon; η is assumed to be 1. The absorbed photon density is simply given by the absorbed power density divided by the photon energy. The absorbed photon density and optical generation rate are written to the output file in units of $s^{-1}cm^{-3}$, the absorbed power density is written in units of Wcm^{-3} , and the power flux density is written in units of Wm^{-2} .

You can obtain all available output quantities in a single extractor by using the keyword `Quantity` as follows:

```
Quantity = { AbsorbedPhotonDensity, OpticalGeneration,
             AbsorbedPowerDensity, PowerFluxDensity }
```

Chapter 8: Extracting and Visualizing Results

Sensors

In addition, you can visualize the region information by using the keyword `Region` in the `Quantity` statement of the `Extractor` section as follows:

```
Quantity = { AbsorbedPhotonDensity, OpticalGeneration,  
             AbsorbedPowerDensity, PowerFluxDensity, Region }
```

Note:

The values for the absorbed photon density and optical generation are identical except in insulator regions where the latter is always zero, independent of the imaginary part of the CRI in that medium.

Sensors

In general, sensors compute some integral quantity based on the values of the electromagnetic field available on the simulation grid. Their results are written to a file whose name is specified using the keyword `ResultFile` in the `Globals` section. Time-averaged results are extracted by probing the time-harmonic state when using a harmonic excitation signal, or by performing Fourier transform when using a Gaussian excitation signal.

One of the basic inputs for a sensor is the specification of the integration domain. The type of geometry that is required depends on the sensor quantity of interest. Power flux density and photon flux density are integrated over a surface; while the integration domain of the absorbed photon density, absorbed power density, and optical generation is a volume defined by a box, a region, or a material of the tensor grid.

A box is defined by two corners whose coordinates are given in units of micrometers. The corners with minimum and maximum values for all three coordinate axes are called `BoxCorner1` and `BoxCorner2`, respectively. A surface is specified by additionally selecting one or more sides of a box. If no side is given, output is generated for all six sides. An easier way to define a surface that extends to the device boundaries is to specify a plane using the keywords `PlaneX`, `PlaneY`, and `PlaneZ`. A plane is defined by a coordinate value for the axis to which it is perpendicular. In addition to specifying explicit coordinate values, you can specify `min` or `max`, which is interpreted as the minimum or maximum device extent along the corresponding coordinate axis.

Time-averaged sensors support two computation modes, which can be controlled with the keyword `Mode`. In both modes, `Integrate` and `Average`, the surface or volume integral of the given quantity is computed. For `Average`, the integrated value is divided by the surface or volume of the integration domain.

The following examples show how to define an optical generation sensor and a power flux density sensor for two different regions, a box, and a plane:

```
Sensor {  
  Name = "sens1"  
  Quantity = OpticalGeneration  
  Region = {"substrate","ambient"} # defined as box in Sentaurus Mesh
```

Chapter 8: Extracting and Visualizing Results

Sensors

```
BoxCorner1 = {0.5,0.5,1.5}
BoxCorner2 = {0.9,0.9,2.9}
Mode = {Integrate,Average}
}

Sensor {
  Name = "sens2"
  Quantity = PowerFluxDensity
  Region = {"substrate","ambient"} # defined as box in Sentaurus Mesh
  Sides = {Zmin, Zmax}
  BoxCorner1 = {0,0,min}
  BoxCorner2 = {1,1,max}
  Mode = {Integrate,Average}
}

Sensor {
  Name = "sens3"
  Quantity = PowerFluxDensity
  PlaneZ = 0.5
  Mode = {Integrate,Average}
}
```

Note:

Sensors whose integration domain is a surface support only box-like regions or materials; whereas, sensors computing a volume integral can have regions or materials with arbitrary shapes as input.

Instead of specifying an explicit coordinate value, you can specify `min` or `max`, which is interpreted as the minimum or maximum device extent along the corresponding coordinate axis.

Computation time for integration domains defined by regions or materials can result in longer runtimes. Therefore, specification of a box or plane is preferable whenever applicable.

Reflection, Transmission, and Absorption Sensors

Quantifying the reflection, transmission, and absorption of light incident on a device structure, for example, a CMOS image sensor, is a common simulation task. If you specify the illumination direction with the keyword `Direction` in a supported `Excitation` section (see [Specification of the Illumination Direction on page 53](#)), EMW can generate the sensors required to compute those important figures of merit for you. This means you do not need to specify several sensors explicitly in the command file, which includes having to determine the exact location of the corresponding sensor planes. The latter can be inconvenient, for example, when your simulation tasks involve varying the device geometry. Such changes often require adjusting the location of the sensors and, therefore, mean that the command file depends on the input mesh. Specifying an empty `RTA` section generates four sensors

Chapter 8: Extracting and Visualizing Results

Sensors

with the properties shown in [Table 8](#). For all sensor keywords not listed in the table, regular default values are used. You can overwrite the default values for a subset of keywords that control the evaluation of the sensors during FDTD time-stepping such as `StartTick/Time`, `EndTick/Time`, and `TickStep` by specifying them in the `RTA` section.

Table 8 *Keywords for reflection, transmission, absorption, and total-field sensors if an empty RTA section is specified in the command file*

Sensor name	Quantity	Mode	PlaneX, PlaneY, PlaneZ	BoxCorner1, BoxCorner2
R	PhotonFluxDensity	Integrate	min/max	—
T	PhotoFluxDensity	Integrate	max/min	—
A	AbsorbedPhotonDensity	Integrate	—	{min,min,min} , {max,max,max}
TF	PhotoFluxDensity	Integrate	<excitation plane>+ <offset>	—

The sensors generated by the `RTA` section produce the same output in the log file and are written to the result file as if they were specified explicitly in the command file. In addition, the following quantities derived from the regular sensor output are reported as design-of-experiments parameters in the log file for automatic variable table import in Sentaurus Workbench and are written to the result file:

- R: Reflection
- T: Transmission
- A: Absorption
- RTA: Sum of reflection, transmission, and absorption, which must equal 1 for a fully converged simulation on a sufficiently fine mesh
- Rnorm: Normalized reflection, that is, R/RTA
- Tnorm: Normalized transmission, that is, T/RTA
- Anorm: Normalized absorption, that is, A/RTA

To prevent certain quantities from being reported in the log file, you can specify explicitly the quantities you are interested in using the keyword `Log` in the `RTA` section. If you want to restrict the type of sensors generated by the `RTA` section by default, for example, to reduce the computational cost incurred by integrating the absorbed photon density over the whole simulation domain, select only the required sensors using the keyword `Sensors`.

Note that specifying at least the reflection sensor is mandatory because it is needed to compute the incident field, based on the implicitly generated total-field sensor.

Note:

Normalized quantities can be computed only if the reflection, transmission, and absorption sensors are enabled either by default or specified explicitly using the keyword `Sensors`.

For example, if you want to compute the reflection, transmission, and absorption not only at the end of the simulation, but also during FDTD time-stepping, and to report only the normalized values in the log file, use the following `RTA` section:

```
RTA {  
    StartTick = 0  
    Log = {Rnorm, Tnorm, Anorm}  
}
```

Note:

The sensor named `TF` generated by the `RTA` section (see [Table 8 on page 79](#)) extracts the total field two tensor cells below or above the excitation plane, depending on whether light is incident from the top or bottom, respectively. Together with the reflection sensor, it is needed to compute the incident field for the evaluation of the reflection, transmission, and absorption.

For DFT simulations, the `WavelengthList`, `WavelengthPrecision`, and `WavelengthRange` keywords are also supported in the `RTA` section and, if specified, they overwrite the default values for sensors (see [RTA on page 137](#)).

Using Multiple Domains in a Sensor

When more than one integration domain is specified, the keyword `Domains` controls how they are plotted as follows:

- `Domains = Separate` (default) plots the integration value for each domain separately.
- `Domains = United` plots one integration value corresponding to the integration volume of the overall domain.
- `Domains = Intersecting` performs the integration only for the intersection of all the specified domains.

Note:

Regions and materials as well as a box and a plane are mutually exclusive integration domains.

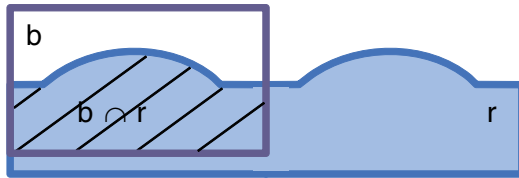
Chapter 8: Extracting and Visualizing Results

Sensors

For example, to extract optical generation in the intersecting domains of a region "r" and a box specification b (see [Figure 9](#)), use:

```
Sensor {  
  Name = "GreenSensor"  
  Quantity = OpticalGeneration  
  Mode = {Integrate, Average}  
  Region = {"r"}  
  BoxCorner1 = {0,0,min}  
  BoxCorner2 = {1,1,max}  
  Domains = Intersecting  
}
```

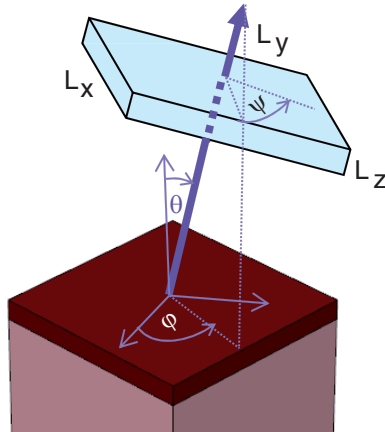
Figure 9 Quantity extraction in limited region allows users to study absorption in intersecting regions



Nonaxis-Aligned Sensor

Power flux sensors also can be defined as nonaxis-aligned boxes (see [Figure 10](#)).

Figure 10 Nonaxis-aligned sensor showing dimensions and rotation angles



The box dimensions L_x , L_y , and L_z are defined by the keyword `BoxDimensions`. The center of the box is set by `BoxCenter`, and the orientation can be controlled by `RotationAngles = { θ , ϕ , ψ }`, where the angles correspond to the definition of the plane wave excitation angles (θ , ϕ , ψ) defined in [Figure 5 on page 56](#), when replacing E with x, H with y, and k with z.

For example, the syntax to place a sensor box centered at (1, 1, 5) is:

```
Sensor {
  Name = "NonAxisAlignedSensor"
  Quantity = PowerFluxDensity
  Mode = {Integrate}
  BoxDimensions = {1.0, 1.0, 0.2}
  BoxCenter = {1.0, 1.0, 5.0}
  RotationAngles = {45, 45, 0}
}
```

Note:

The nonaxis-aligned sensor is supported only for measuring power flux and in three dimensions.

Note that a second rotation matrix following [2] can be chosen by setting `CoordinateTransformation=Standard`:

$$\bar{\mathbf{R}}(\theta, \phi, \psi) = \bar{\mathbf{R}}(\theta, \phi) \bar{\mathbf{R}}(\psi) = \begin{bmatrix} \sin\phi & \cos\theta\cos\phi & \sin\theta\cos\phi \\ -\cos\phi & \cos\theta\sin\phi & \sin\theta\sin\phi \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (38)$$

Far Field

EMW supports the computation of the far field by a near-to-far-field transformation, which is based on the derivation described in [3][4]. The near field is the result of a single FDTD simulation and it is extracted in the scattered-field region. The calculated far-field intensity is represented as a function of the polar diffraction angle, θ , and the azimuthal diffraction angle, ϕ , as illustrated in Figure 11. To obtain the far-field power, the intensity can be integrated over a user-supplied solid angle. For device structures that are periodic in the x- and the y-direction, the diffraction modes for a given array size are computed *a priori* assuming that the array is periodic to infinity.

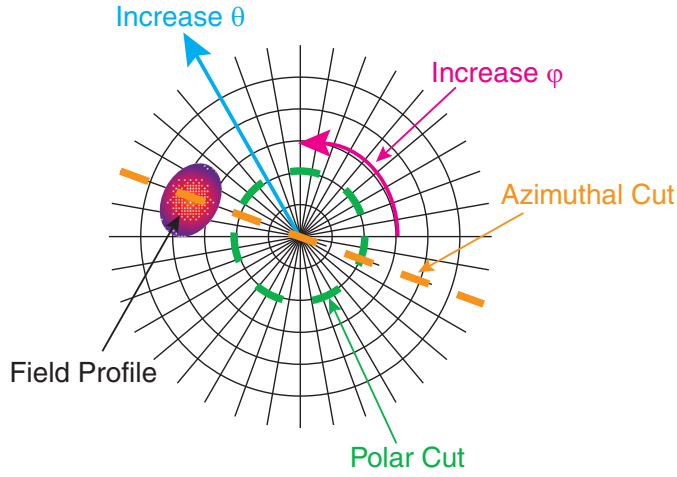
Note:

Far-field computation is supported only in three dimensions, and the main propagation direction of the incident light must be in the positive z-direction. The normal of the excitation plane must align with the z-axis.

Chapter 8: Extracting and Visualizing Results

Far Field

Figure 11 Illustration of polar plot for representation of far field



For a scattering grating array, the angles for the backscattered modes in the far-field zone can be computed by momentum matching:

$$(k_{n,m})_x = (k^{\text{inc}})_x + n \frac{2\pi}{L_x} \quad (39)$$

$$(k_{n,m})_y = (k^{\text{inc}})_y + m \frac{2\pi}{L_y} \quad (40)$$

In Equation 39 and Equation 40, n and m denote the mode order with respect to the x-direction and the y-direction. k^{inc} corresponds to the wavevector of the incident light, and $k_{n,m}$ stands for the wavevector of the scattering mode of order n and m .

Equation 39 and Equation 40 also can be expressed in polar coordinates to obtain the polar and the azimuthal scattering angles, θ^{sca} and ϕ^{sca} , for each mode, which are among the output quantities of the far-field feature:

$$\frac{2\pi}{\lambda} \sin \theta^{\text{sca}} \cos \phi^{\text{sca}} = \frac{2\pi}{\lambda} \sin \theta^{\text{inc}} \cos \phi^{\text{inc}} + n \frac{2\pi}{L_x} \quad (41)$$

$$\frac{2\pi}{\lambda} \sin \theta^{\text{sca}} \sin \phi^{\text{sca}} = \frac{2\pi}{\lambda} \sin \theta^{\text{inc}} \sin \phi^{\text{inc}} + m \frac{2\pi}{L_y} \quad (42)$$

For the computation of the far field of a finite array of size (n_x, m_y) , the near field resulting from the FDTD simulation is replicated accordingly before the near-to-far-field transformation is applied. With increasing array size, the scattering angles of the *a priori* calculated modes will converge with the maxima of the far-field intensity in the polar plot.

In this limit, the relative diffraction efficiency η_{rel} of a mode defined as:

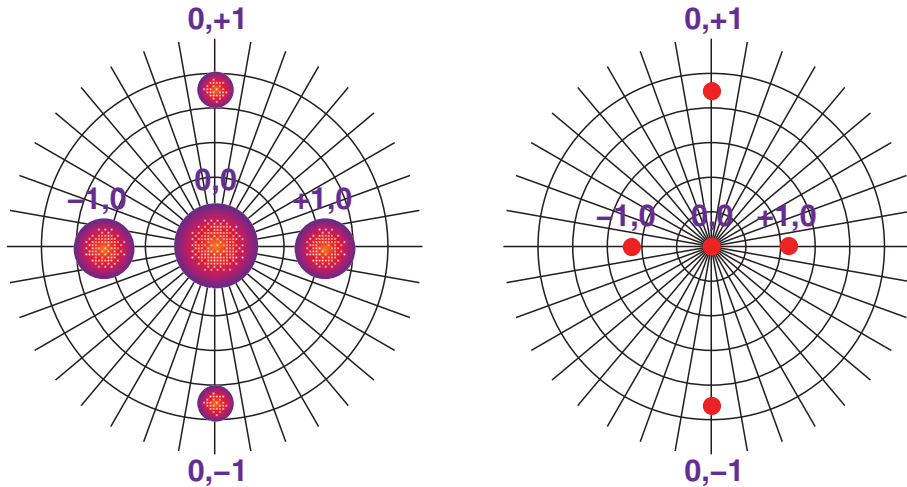
$$\eta_{\text{rel}} = \frac{P_{\text{mode}}}{P_{\text{total}}} = \frac{\int_{\Omega_{\text{mode}}} E \wedge H d\Omega}{\int_{\Omega_{\text{total}}} E \wedge H d\Omega} \quad (43)$$

where Ω denotes the solid angle, can be approximated by the maxima of the far-field modes:

$$\eta_{\text{rel}} = \frac{P_{\text{mode}}}{P_{\text{total}}} = \frac{S(\theta_{\text{mode}}^{\text{sca}}, \phi_{\text{mode}}^{\text{sca}})}{\sum_{\text{mode}_i} S(\theta_{\text{mode}_i}^{\text{sca}}, \phi_{\text{mode}_i}^{\text{sca}})} \quad (44)$$

In Equation 44, $S(\theta, \phi)$ represents the time-averaged Poynting vector at the scattering angles θ and ϕ . Figure 12 illustrates the computation of the relative diffraction efficiency using Equation 43 and in the limit of an infinite array size using Equation 44.

Figure 12 *Illustration of computation of relative diffraction efficiency: (left) the power of a mode is computed by integrating over a solid angle given by the expansion of the mode and (right) in the limit of an infinite array size, the power of a mode is determined by the maximum value of the far-field intensity*



Specifying the Far-Field Computation

The far-field computation can be performed at the end of the FDTD simulation similar to sensors and extractors, or in a postprocessing step in which a previously saved near field is loaded from a file. In both cases, the far-field computation and its output are controlled by the keywords of the respective `Farfield` section, which are described here.

Chapter 8: Extracting and Visualizing Results

Far Field

Information related to postprocessing only is given in [Postprocessing of Extractors, Sensors, and Far Field on page 89](#).

In its simplest form, a `Farfield` section requires only the specification of a name and a domain representing a plane in the near-field zone:

```
Farfield {
  Name = "farfield"
  BoxCorner1 = {0,0,0}
  BoxCorner2 = {2,3,0}
  # OR
  Region = "near-field-plane"
}
```

This section will trigger the computation of the far field at the default distance of 1 mm from the near-field plane. As a result, the far-field intensity as a function of the polar and the azimuthal diffraction angles is written to a TDR file named `farfield_eml.tdr`. In addition, the output from the computation of the far-field modes, corresponding to an infinite array size, and the integration of the far-field intensity over the entire hemisphere are saved in the `ResultFile` specified in the `Globals` section along with the output of the optional `Sensor` and `Detector` sections.

Additional keywords can be used to customize the previously mentioned output. With `PolarRange` and `AzimuthalRange`, the plotting of the far-field intensity as a function of diffraction angles can be restricted to a smaller solid angle than the hemisphere, and the angular discretization can be controlled:

```
Farfield {
  ...
  PolarRange = {30, 70, 0.5}
  AzimuthalRange = {90, 180, 2}
}
```

The first two entries of the range list specify the minimum angle and the maximum angle of the interval, respectively, and the third entry determines the angular discretization using an equidistant step size. All numbers are given in degrees. Only polar angles between 0° and 90° are supported where the polar angle is taken from the positive z-axis. The azimuthal angle can range from 0° to 360° and is measured from the positive x-axis in the direction of the positive y-axis.

It is also possible to save xy plots of the far-field intensity given a constant polar or azimuthal angle as they tend to be more amenable to quantitative analysis. Such angular cuts can be specified by either a range or an explicit list of angles:

```
Farfield {
  ...
  PolarCutRange = {0, 90, 30}
  AzimuthalCutList = {0, 45, 90, 180, 270}
}
```

Chapter 8: Extracting and Visualizing Results

Far Field

In this example, the far-field intensity as a function of the azimuthal angle is plotted for the polar angles 0° , 30° , 60° , and 90° , where the entries of the range list are the minimum polar angle, the maximum polar angle, and the step size, respectively. In addition, the far-field intensity as a function of the polar angle is plotted for the azimuthal angles specified in `AzimuthalCutList`.

To compute the total far-field power by integrating the far-field intensity over the entire hemisphere, you can specify a list of limited integration ranges. Each range consists of six numbers where the first three and the last three numbers correspond to the azimuthal and the polar angle range including the step size, respectively:

```
Farfield {  
  ...  
  IntegrationRange = { {0, 10, 0.2, 0, 360, 0.5}, {30, 45, 0.4, 80,  
    100, 1.} }  
}
```

This option can be useful to integrate over the expansion of a particular mode to obtain its power. The above example shows how integrating over mode (0, 0) and mode (0, 1) according to [Figure 12](#) could look like.

Note:

The far-field computation is performed independently for each of the different output quantities. For example, the choice for `PolarRange` and `AzimuthalRange` does not affect the computation of the angular cuts or the integration of the far-field intensity to obtain the far-field power. This also implies that the computational effort scales with the number of output quantities requested, for example, the number of integration ranges, and their specific discretization.

For FDTD simulations with periodic boundary conditions in the lateral direction, the near-field domain can be replicated in both the x- and y-direction to better approximate the actual physical extension of the device structure for the far-field calculation. How many times the near-field domain is replicated in the respective coordinate direction is set using the keyword `ArraySize`:

```
Farfield {  
  ...  
  ArraySize = {10, 20} # {nx, my}  
}
```

The radial distance at which the far field should be computed is given by the `Radius` keyword.

For a complete list of supported keywords in the `Farfield` section and their descriptions, see [Farfield on page 118](#).

Visualizing Results of the Far-Field Computation

This section summarizes the various output quantities produced by the far-field computation. They are saved either in the TDR file specified in the `Farfield` section or in the general PLT `ResultFile` specified in the `Globals` section. The name of the TDR file is derived from the `OutputFilePrefix` specified in the `Globals` section and the `Farfield` section name.

Table 9 Polar plot of far-field intensity saved in TDR file

Quantity	Description
<code>AzimuthalAngle</code>	Azimuthal diffraction angle. Unit: deg
<code>FarfieldIntensity</code>	Far-field intensity as a function of the azimuthal and polar diffraction angles. Unit: Wsr^{-1}
<code>FarfieldIntensityMask</code>	Auxiliary field indicating for which vertices the far-field intensity has actually been computed given the <code>AzimuthalRange</code> and the <code>PolarRange</code> selected in the <code>Farfield</code> section. All other vertices are assigned a zero default value. Unit: 1
<code>PolarAngle</code>	Polar diffraction angle. Unit: deg

Table 10 Polar cuts saved in PLT file

Quantity	Description
<code>AzimuthalAngle</code>	Azimuthal angle. Unit: deg
<code>FarfieldIntensity_<float>deg</code>	Far-field intensity as a function of the azimuthal angle for a constant polar angle of <code><float></code> degrees. Unit: Wsr^{-1}

Table 11 Azimuthal cuts saved in PLT file

Quantity	Description
<code>FarfieldIntensity_<float>deg</code>	Far-field intensity as a function of the polar angle for a constant azimuthal angle of <code><float></code> degrees. Unit: Wsr^{-1}
<code>PolarAngle</code>	Polar angle. Unit: deg

Chapter 8: Extracting and Visualizing Results

Far Field

Table 12 Far-field power for specified integration ranges saved in PLT file

Quantity	Description
AzimuthalAngleCenter	Center of the azimuthal angle range for the given integration range. Unit: deg
AzimuthalAngleWidth	Width of the azimuthal angle range for the given integration range. Unit: deg
IntegrationRangeIndex	Index to identify the far-field power corresponding to a specified integration range. It follows the order of the list entries given for the <code>IntegrationRange</code> keyword. The list index starts with number 1. Index 0 stands for the entire hemisphere as the integration range, which is always computed even if <code>IntegrationRange</code> is not specified. Unit: 1
PolarAngleCenter	Center of the polar angle range for the given integration range. Unit: deg
PolarAngleWidth	Width of the polar angle range for the given integration range. Unit: deg
Power	Far-field power for the given integration range. Unit: W

Table 13 Far-field modes for infinite array structure saved in PLT file

Quantity	Description
AzimuthalAngle	Azimuthal scattering angle ϕ^{sca} for the given mode as defined in Equation 41 and Equation 42 . Unit: deg
ModeIndex	Index to identify the far-field modes of an infinite array structure as defined in Equation 41 and Equation 42 . The index is defined by linearizing the 2D mode order where n is the fast-running index and m is the slow-running index. Unit: 1
ModeOrderX	Mode order n defined in Equation 41 . Unit: 1
ModeOrderY	Mode order m defined in Equation 42 . Unit: 1
PolarAngle	Polar scattering angle θ^{sca} for the given mode as defined in Equation 41 and Equation 42 . Unit: deg
RelativeDiffractionEfficiency	Relative diffraction efficiency for the given mode according to Equation 44 . Unit: 1

Postprocessing of Extractors, Sensors, and Far Field

To use the postprocessing feature of extractors, sensors, and the far field, the required underlying data must be saved in a file at the end of the FDTD simulation. This file can be loaded later by EMW to apply extractors, sensors, and the far-field computation without having to rerun the FDTD simulation, and it does not require an EMW license.

The quantities `AbsorbedPhotonDensity` and `PowerFluxDensity` are required for the postprocessing of extractors and sensors.

The syntax for saving a general results file for extractors and sensors is:

```
Save {
    Name = "n@node@_save"
    Quantity = {AbsorbedPhotonDensity, PowerFluxDensity}
    # domain specification (default entire device)
    BoxCorner1 = {0,0,0}
    BoxCorner2 = {4,6,8}
    # or
    Region = {"substrate"}
}
```

Note:

The keyword `Region` is not a supported entry for `Quantity` because the region information is saved by default in the general results file.

The postprocessing feature for the far-field computation requires only the quantity `CplxEMField` to be saved, which is an alias for the datasets `RealElectricField`, `ImagElectricField`, `RealMagneticField`, and `ImagMagneticField` to be written to the TDR file. For the reflected near-field, it is typically sufficient to save the quantity `CplxEMField` on a single plane in the near-field zone. However, at the expense of higher memory consumption and disk space usage, it is also possible to save the data in the entire 3D domain or a subdomain thereof, to allow for the selection of arbitrary near-field planes during postprocessing.

The syntax for saving a results file for far-field computation in postprocessing mode is:

```
Save {
    Name = "n@node@_save"
    Quantity = CplxEMField
    # plane at z=0 in near-field zone
    BoxCorner1 = {0,0,0}
    BoxCorner2 = {4,6,0}
    # or entire domain
    BoxCorner1 = {0,0,-4}
    BoxCorner2 = {4,6,8}
    # or
    Region = "near-field-region"}
}
```

Chapter 8: Extracting and Visualizing Results

Discrete Fourier Transform

The `Extractor`, `Sensor`, and `Farfield` sections can be specified at the same time during postprocessing, but since only one `LoadFile` can be read, the data therein must contain all the required quantities in a domain that includes all the domains specified in the different sections.

A `Save` section is like an `Extractor` section; however, it only supports `AbsorbedPhotonDensity`, `IncoherentAbsorbedPhotonDensity` (exclusively for incoherent FDTD simulations), `PowerFluxDensity`, and `CplexEMField` as `Quantity`. Further quantities allowed in extractors and sensors can be derived from those base quantities during postprocessing.

The command file used for postprocessing consists of a mandatory `Globals` section and the optional `Sensor`, `Extractor`, and `Farfield` sections:

```
Globals {
    OutputFilePrefix = "n@node@_"
    LoadFile = "n@node@_save_eml.tdr"
    ResultFile = "eml.plt"
}

Sensor {...}

Extractor {...}

Farfield {...}
```

Note:

If the `Globals` section contains the keyword `LoadFile`, then no other sections apart from `Globals`, `Sensor`, `Extractor`, and `Farfield` are allowed.

For details about the `OutputFilePrefix` keyword, see [Globals on page 124](#).

Discrete Fourier Transform

This section presents an overview of the discrete Fourier transform (DFT) feature that is implemented in EMW. A major motivation of the DFT is to obtain frequency responses at multiple frequencies from a single transient simulation. After obtaining electrical and magnetic fields at multiple frequencies, by executing the DFT from the time-domain simulation results, secondary quantities such as photon flux density are calculated at these frequencies of interest. This section covers the basic theory, practical considerations, and limitations of the DFT.

On-the-Fly DFT

Suppose $g(\mathbf{r}, t)$ represents the space- and time-dependent field values in the FDTD simulation, where $\mathbf{r} = (x, y, z)$ and $g = E_x, E_y, E_z, H_x, H_y, H_z$. Since the DFT in EMW is unrelated to spatial coordinates, the space-dependent variable \mathbf{r} can be omitted from the function g in the following derivation without loss of generality.

The continuous-time Fourier transform of $g(t)$ is:

$$G(\omega) = \int g(t) \exp(-i\omega t) \quad (45)$$

where $\omega = 2\pi f$. Within the FDTD framework, the above integration is approximated by the discrete-time Fourier transform:

$$G(\omega) = \sum_{n=0}^{N-1} g(n\Delta t) \exp(-i\omega n\Delta t) \quad (46)$$

During the FDTD simulation, the field value $g(t)$ is accumulated on the fly from the first time step at $t = 0$ until the last time step at $t = (N-1)\Delta t$.

The resolution of the discrete spectrum, that is, Δf , depends on both the number of DFT points (N) and the FDTD time step (Δt) by:

$$\Delta f = \frac{1}{N\Delta t} \quad (47)$$

Since the time step Δt is limited by the stability (Courant) criteria, the only way to increase the spectral resolution is to increase the number of time steps N .

Defining the DFT Wavelength Table

Light is often described in wavelengths instead of frequencies in optical simulations. Therefore, the `DFT` section in EMW defines a wavelength table for extracting the DFT fields. The DFT wavelength table is defined by:

$$\{\lambda | \lambda = \lambda_{\min} + l\Delta\lambda, \Delta\lambda = (\lambda_{\max} - \lambda_{\min}) / (N_\lambda - 1), 0 \leq l < N_\lambda\} \quad (48)$$

The wavelength resolution $\Delta\lambda$ is calculated from the user-defined input `MinWavelength` (λ_{\min}), `MaxWavelength` (λ_{\max}), and `NumberOfWavelengths` (N_λ).

Then, the corresponding DFT frequency table is:

$$\{f^{\text{tab}}\} = c_0 / \{\lambda\} \quad (49)$$

where c_0 is the vacuum speed of light.

Note:

In EMW, any wavelength-related keyword refers to the wavelength in vacuum.

Chapter 8: Extracting and Visualizing Results

Discrete Fourier Transform

The spectral resolution of the DFT frequency table might be different from that of the on-the-fly DFT. However, the former cannot be higher than the latter in order to obtain distinctive fields, that is, $\min(\Delta f^{\text{tab}}) \geq \Delta f$.

To access the DFT field $G(\omega)$ for a specific wavelength or for a range of wavelengths, different keywords can be used:

- **WavelengthList**: Specifies a comma-separated list of wavelengths from the wavelength table defined in the **DFT** section.
- **WavelengthRange**: Specifies a wavelength interval that includes at least one wavelength from the wavelength table defined in the **DFT** section. The interval boundaries do not have to coincide with any entry of the wavelength table.

The wavelengths specified using the keyword **WavelengthList** must match the wavelengths of the DFT wavelength table. Depending on the discretization of the wavelength table, wavelength entries with several digits after the decimal point can exist, and an exact match depends on the applied numeric precision. Therefore, the numeric precision can be controlled using the keyword **WavelengthPrecision**.

For a wavelength λ_1 specified in **WavelengthList** to be considered an exact match with a wavelength λ_i from the DFT wavelength table, the following condition must be fulfilled, where p is the value given for **WavelengthPrecision**:

$$|\lambda_1 - \lambda_i| < 0.5 \times 10^{-p} \quad (50)$$

The keywords **WavelengthList**, **WavelengthRange**, and **WavelengthPrecision** can be specified in the **Plot**, **Extractor**, and **Sensor** sections for DFT simulations. If both **WavelengthList** and **WavelengthRange** are specified at the same time, then the union of wavelengths is selected. The selected wavelengths can be only a subset of the table defined in [Equation 49](#). In such a case, EMW automatically calculates for only the selected wavelengths to save memory and computation time.

For examples using the keywords **WavelengthList** and **WavelengthRange**, see [EMW Syntax on page 95](#).

Note:

For DFT simulations, you must specify **WavelengthList** or **WavelengthRange** for extractors and sensors. Specifying neither of them in a **Plot** section selects the regular non-DFT field of the transient simulation with excitation wavelength as specified in the **Excitation** section if it exists or the wavelength corresponding to the automatically computed frequency given in [Equation 57 on page 95](#).

Normalizing Field Values

When the input signal is a sinusoid, its Fourier-transformed input spectrum is the sum of two delta functions. Since EMW deals only with linear materials (therefore, a linear system), the output spectrum consequently consists of delta functions.

It only makes sense to extract DFT fields at the frequency of the sinusoid f_c :

$$F[\cos[2\pi f_c(t - t_0)]] = \frac{1}{2}[e^{-i2\pi t_0(f - f_c)}\delta(f - f_c) + e^{-i2\pi t_0(f + f_c)}\delta(f + f_c)] \quad (51)$$

However, the output spectrum must be scaled by $e^{-i2\pi t_0(f - f_c)}/2$ to obtain the same result as in the non-DFT simulation at $f = f_c$.

When the input signal is a transient Gaussian pulse, the input spectrum is a Gaussian function with its center shifted to the modulation frequency f_c :

$$F\left[e^{-\frac{(t - t_0)^2}{2\sigma^2}} \cos(2\pi f_c t)\right] = \sqrt{\frac{\pi\sigma^2}{2}} \left[e^{-i2\pi t_0(f - f_c)} e^{-2[\pi\sigma(f - f_c)]^2} + e^{-i2\pi t_0(f + f_c)} e^{-2[\pi\sigma(f + f_c)]^2} \right] \quad (52)$$

where t_0 and σ are the delay and the full width at half maximum (FWHM), respectively. From the scaling property of the Fourier transform, the narrower the time-domain Gaussian pulse, the wider its spectrum. The shape of the spectrum can be designed by adjusting the FWHM.

The spectrum of a Gaussian pulse has its maximum at the modulation center frequency. The energy decays rapidly as the frequency shifts away from the center frequency.

To make the DFT useful for broadband simulation, the field values, the power flux densities, and the optical generation must be normalized by the magnitude of the input spectrum.

This normalization procedure ensures that the input power is the same for every frequency component, and EMW performs this automatically. What you obtain from the simulation is the normalized spectrum $G(\omega)/G_{\text{input}}(\omega)$ instead of $G(\omega)$. In an FDTD simulation, the excitation waveform might not be exactly sinusoidal or Gaussian, so the input spectrum $G_{\text{input}}(\omega)$ is obtained from on-the-fly DFT instead of the above analytic formulas.

Downsampling of DFT

In many practical cases, the sampling frequency intrinsic to FDTD ($1/\Delta t$) is much larger than the maximum DFT frequency (f_{max}), while the Nyquist–Shannon sampling theorem requires $f_{\text{sample}} > 2f_{\text{max}}$.

Chapter 8: Extracting and Visualizing Results

Discrete Fourier Transform

Based on this theorem and anti-aliasing considerations, consider a downsampling factor D such that $1/(D\Delta t) > 4f_{\max}$ or $D < \lambda_{\min}/(4c_0\Delta t)$. Equation 46 is rewritten as:

$$G(\omega) = \sum_{\tilde{n}=0}^{\left\lfloor \frac{N-1}{D} \right\rfloor} g(\tilde{n}D\Delta t) \exp(-i\omega\tilde{n}D\Delta t) \quad (53)$$

where $\left\lfloor \dots \right\rfloor$ denotes round towards zero. During the FDTD simulation, the field value $g(t)$ is accumulated on the fly by a time step of $D\Delta t$. If D is large (for example, greater than 100), downsampling will significantly reduce the computation time for the DFT calculation without the loss of accuracy.

Limitation of Single-Frequency Periodic Boundary Condition

The limitation on broadband simulation mainly comes from the single-frequency periodic boundary condition (PBC). For the sine–cosine method, the fields are copied from one periodic boundary to another with a phase shift that is dependent on the frequency.

Suppose $S_{x\min}$ and $S_{x\max}$ are a pair of periodic boundaries. The following relationship is enforced to ensure the periodicity in the x-direction:

$$g(\mathbf{r}, t)|_{\mathbf{r} \in S_{x\min}} = e^{-ik_x(x_{\max} - x_{\min})} g(\mathbf{r}, t)|_{\mathbf{r} \in S_{x\max}} \quad (54)$$

where:

- $k_x = \hat{x} \cdot \hat{k} 2\pi f_c / v$ is the x-component of the wave vector.
- $g = E_x, E_y, E_z, H_x, H_y, H_z$.
- v denotes the speed of light in the excitation medium.

Since the phase shift $k_x(x_{\max} - x_{\min})$ depends on the modulation (or center) frequency f_c of a frequency-shifted Gaussian pulse, it is only constant at f_c . At the other frequencies away from f_c , the excitation component must assume different incident angles for the PBC to maintain the same constant phase shift that was computed at f_c . Therefore, a broadband Gaussian excitation is equivalent to the sum of excitations with different frequencies and different incident angles.

For a structure that is doubly periodic in the x- and y-directions, suppose the incident plane is the x-plane, that is, $k_y = 0$, and the DFT frequency is f . By phase-matching, you find the incident angle associated with the DFT frequency is $\alpha = \sin^{-1}[\sin \alpha_c(f_c/f)]$. Similarly, if the incident plane is the y-plane, that is, $k_x = 0$, the incident angle associated with the DFT frequency is $\beta = \sin^{-1}[\sin \beta_c(f_c/f)]$.

However, when the incident plane is neither the x-plane nor the y-plane, the broadband injection angles (θ, ϕ) must be solved by the following nonlinear equations:

$$\begin{cases} \sin \theta \cos \phi = (f_c/f) \sin \theta_c \cos \phi_c \\ \sin \theta \sin \phi = (f_c/f) \sin \theta_c \sin \phi_c \end{cases} \quad (55)$$

When $\theta = \theta_c = 0$, the above equations hold for any DFT frequency f . Therefore, the PBC is independent of the frequency at normal incidence. In the case of normal incidence, it is unnecessary to calculate broadband injection angles even if the signal is broadband.

EMW Syntax

To set up a DFT wavelength table with 10 wavelengths, use:

```
DFT {
  MinWavelength = 350          # nm
  MaxWavelength = 750          # nm
  NumberOfWavelengths = 10
}
```

To extract DFT fields for the above wavelength table, use:

```
Plot/Extractor/Sensor {
  ...
  WavelengthRange = {250, 500}  # 350, 394.44, 438.89, 483.33 nm
}

Plot/Extractor/Sensor {
  ...
  WavelengthList = {750, 394, 528}  # 750, 394.44, 527.78 nm
}
```

Note:

Parentheses are used to define an interval, and braces are used to define a list.

If the `DFT` section is specified, you must use a pulsed excitation by specifying `Signal=Gauss` in the `PlaneWaveExcitation` section or the `TruncatedPlaneWaveExcitation` section (see [Temporal Dependency on page 61](#)). In this case, EMW automatically calculates the pulse width, the frequency, and the delay as follows:

$$\sigma = 2\sqrt{\ln 2} / \left(\pi c_0 \left(\frac{1}{\lambda_{\min}} - \frac{1}{\lambda_{\max}} \right) \right) \quad (56)$$

$$f = \frac{c_0}{2} \left(\frac{1}{\lambda_{\min}} + \frac{1}{\lambda_{\max}} \right) \quad (57)$$

$$t_0 = 3\sigma \quad (58)$$

Chapter 8: Extracting and Visualizing Results

References

If $\lambda_{\min} = \lambda_{\max}$, then NumberOfWavelengths must be 1, and:

$$\sigma = \frac{\lambda_{\min}}{c_0} = \frac{\lambda_{\max}}{c_0} \quad (59)$$

$$f = \frac{c_0}{\lambda_{\min}} = \frac{c_0}{\lambda_{\max}} \quad (60)$$

In [Equation 56](#), [Equation 57](#), [Equation 59](#), and [Equation 60](#), λ_{\min} , λ_{\max} , and c_0 denote MinWavelength, MaxWavelength, and the vacuum speed of light, respectively. If you specify σ using the keyword Sigma, or f using the keyword Frequency or Wavelength, or t_0 using the keyword Delay, EMW uses the user-defined value instead of the automatically calculated value.

To specify the downsampling rate of DFT, use:

```
DFT {  
    ...  
    DownSampling = <integer>  
}
```

If the user-specified downsampling rate is less than $\lambda_{\min}/(4c_0\Delta t)$, the user-specified value will be used. If the user-specified downsampling rate is greater than $\lambda_{\min}/(4c_0\Delta t)$, an error message will be issued because DFT accuracy will not be guaranteed. If the DownSampling keyword is not specified, EMW will automatically use $\lambda_{\min}/(4c_0\Delta t)$ as the downsampling rate.

Note:

Do not use the DFT with the CRI library. Instead, specify `epsilon_r` and `sigma` for each material in the parameter file, or consider using dispersive models (see [Dispersive Model Parameters on page 44](#)).

References

- [1] F. Alauzet and M. Mehrenberger, "P¹-conservative solution interpolation on unstructured triangular meshes," *International Journal for Numerical Methods in Engineering*, vol. 84, no. 13, pp. 1552–1588, 2010.
- [2] J. A. Kong, *Electromagnetic Wave Theory*, Cambridge, Massachusetts: EMW Publishing, 2008.
- [3] A. Taflove and S. C. Hagness, *Computational Electromagnetics: The Finite-Difference Time-Domain Method*, Boston: Artech House, 3rd ed., 2005.
- [4] C. A. Balanis, *Advanced Engineering Electromagnetics*, New York: John Wiley & Sons, 1989.

9

Parallelization

This chapter describes parallelization of EMW using multithreading and the message passing interface.

Overview of Parallelization

EMW supports parallel finite-difference time-domain (FDTD) computation for wave propagation, radiation, and scattering on shared-memory parallelization (SMP) systems, distributed processing (DP) systems, and hardware-accelerated systems.

SMP and DP systems are usually referred to as multicores and clusters, respectively. EMW performs differently on SMP and DP systems, depending on the computer architectures, the bandwidth of interconnects and memory buses, and the sizes and domain decomposition of the problems to be solved. You can also use SMP on DP systems, which is referred to as the mixed DP and SMP parallel mode.

Shared-Memory Parallelization

In general, the number of threads to be used in an EMW simulation can be specified either in the `Globals` section of the command file or by using the `--threads` command-line option. If you specify both, then the value specified on the command line overrides the value specified in the command file.

Examples

These examples illustrate both methods to set the number of threads to 2.

Specify the number of threads in the `Globals` section:

```
Globals {  
    ...  
    NumberOfThreads = 2    # or maximum  
    ...  
}
```

Chapter 9: Parallelization

Parallelization Using the Message Passing Interface

Specifying `NumberOfThreads = maximum`, which is supported only in the command file, sets the number of threads to the maximum number available on a given machine.

Specify the number of threads on the command line:

```
emw --threads 2 <command_file>
```

By default, the number of threads is set to 1. EMW supports another related command-line option `--max_threads`, which defines an absolute upper limit. This option applies to all Sentaurus tools offering shared-memory parallelization for better tool integration. For details, see the *TCAD Parallelization Environment Setup User Guide*.

Parallelization Using the Message Passing Interface

Distributed computation of FDTD can be activated using EMW with the message passing interface (MPI) [1]. EMW uses the MPICH library for MPI implementation [2]. In this methodology, domain decomposition is used, and each subdomain problem of FDTD is distributed to be solved by separate processes, which can run on a single machine or multiple machines of a cluster. In addition, it is possible to have each process use SMP to speed up the computation for each subdomain. This hybrid parallelization strategy is referred to as the mixed DP and SMP parallel mode.

The best performance on a cluster is achieved if you use cluster machines with special fast interconnections between machines. The MPI job also can be distributed over a network of machines, but the speedup is limited by the speed and bandwidth of the network connections between machines.

Depending on the available hardware resources and FDTD problem size, running a simulation in mixed DP and SMP parallel mode might lead to superior performance compared to a standard DP approach.

Domain Decomposition

The automatic domain decomposition (ADD) algorithm in EMW decomposes both 2D and 3D structures for optimal performance of the code in parallel mode.

Note:

The ADD algorithm applies only to the DP parallel mode of EMW (denoted as EMW MPI).

The algorithm chooses an integer decomposition of the number of processes to be as close as possible to the product of the square and cubic roots for 2D and 3D structures, respectively.

Chapter 9: Parallelization

Choosing Between Shared-Memory Parallelization and Distributed Processing

The scalability degrades when the number of processes is a large prime number. The best performance can be achieved for 2D and 3D structures when the number of processes can be expressed as a square and a cubic of an integer, respectively.

The minimum number of cells in each direction of each subdomain is set to ten except for the degenerated dimension in 2D structures. The ADD algorithm also does not decompose the simulation domain in the convolutional perfectly matched layer (CPML). Therefore, the number of processes cannot be arbitrarily large for a given problem. If the ADD algorithm cannot find a suitable decomposition, EMW terminates.

For a simulation domain that has a greater number of cells in one of the Cartesian directions, a domain decomposition in this direction is possible to achieve better speedup than the ADD algorithm. To enforce ADD (default) or domain decomposition in the x-, y-, or z-direction only, use the following syntax:

```
Globals { ...
    DomainDecomposition = Automatic | X | Y | Z
}
```

Each decomposed subdomain is solved by one MPI process.

Installing and Running EMW MPI on DP Systems

See the *TCAD Parallelization Environment Setup User Guide*.

Choosing Between Shared-Memory Parallelization and Distributed Processing

In SMP systems, many cores share one set of memory. The operating system manages the distribution of multiple threads to the cores, and multithreading is largely invisible to users. Threads are not bound to specific cores, and the number of threads can be greater than the number of cores on a node. However, when a simulation requires large amounts of memory and threads need to access memory modules belonging to different CPUs, the performance of SMP programs will be adversely affected because of the limited throughput of memory buses and cache thrashing.

To obtain the best performance on SMP systems, it is recommended to choose platforms with larger CPU caches and the latest memory bus architectures. Higher throughput memories might help to improve the speedup on SMP systems.

DP systems consist of a number of individual computing nodes connected by a communication network. DP programs are managed by an MPI process manager, which distributes the computing processes and is responsible for their communication to hardware resources on the computing nodes. The best performance is achieved if each node executes a number of processes that is less than or equal to the number of its cores.

Chapter 9: Parallelization

Hardware Acceleration Using Graphics Processing Units

Alternatively, in the mixed DP and SMP parallel mode, the best performance is achieved when the number of MPI processes (DP) equals the number of CPUs, and the number of threads (SMP) equals the number of cores per CPU.

In contrast to SMP programs, the scalability of DP programs is often good for a large number of nodes. If the size of the problem is sufficiently large, the communication (or data exchange) cost is much less than the computing cost for each subdomain, and DP programs definitely outperform SMP programs. However, solving a small problem by domain decomposition is not efficient because of the overhead of communication between MPI processes in DP systems.

Hardware Acceleration Using Graphics Processing Units

Hardware acceleration using graphics processing units (GPUs) has been used successfully in many fields of computational science and engineering to significantly increase the simulation performance. Due to its regular structure, the FDTD stencil is well suited to be implemented efficiently on a hardware accelerator such as a modern GPU.

To speed up FDTD simulations beyond the possibilities of SMP and DP systems, EMW can be run on NVIDIA® Tesla® GPUs [3]. EMW supports both single-GPU and multi-GPU simulations. Multi-GPU simulations can be run on a single machine with more than one GPU installed or on DP systems where each cluster machine contains at least one GPU.

After the GPU card has been installed, hardware-accelerated EMW simulations are as simple to set up and run as single-threaded simulations or simulations using SMP. Multi-GPU simulations are based on the domain decomposition method using MPI where the computation of the subdomains is handled by separate processes each accessing a dedicated GPU. For multinode multi-GPU simulations, the same considerations as outlined in [Parallelization Using the Message Passing Interface on page 98](#) apply.

Installing EMW GPU

To run EMW with hardware acceleration, denoted by EMW GPU, you must run it on a machine with a supported NVIDIA Tesla GPU accelerator [3] such as:

- A100
- K10
- K20
- K40
- K80

Chapter 9: Parallelization

Hardware Acceleration Using Graphics Processing Units

- P100
- V100

For details about the hardware specifications of the Ampere (A), Kepler (K), Pascal (P), and Volta (V) series of GPUs, see [\[3\]](#).

The following sections outline the steps required to set up the hardware and software to achieve the best performance.

Hardware and Software Setup

If you do not have access to a server with a supported GPU, you must first purchase one of the accelerators previously mentioned. They differ in performance and the size of the GPU device memory, where the latter determines the maximum FDTD problem size that can be simulated on the GPU.

Note:

GPU cards are installed using the Peripheral Component Interconnect Express (PCI-e) interface, and it is important to check that the assigned server fulfills the requirements regarding power supply and heat sink as most recent GPU models are not actively cooled.

After the GPU card has been physically installed, you must install the corresponding NVIDIA® driver software, which requires root privileges. It is recommended to install the latest driver version for the given GPU model. You can download the driver software from the NVIDIA website [\[3\]](#).

Optimization Settings

For high-performance computing workloads, changing the default settings of the GPU can increase performance. It is recommended to run the GPU in persistence mode, which can be activated for all GPUs by running the following command (requires root privileges):

```
nvidia-smi -pm 1
```

The NVIDIA System Management Interface (`nvidia-smi`) tool allows you to control the GPU and provides runtime statistics similar to the `top` program on Linux. It is shipped with the driver software and, by default, it is located in `/usr/bin`.

Memory bandwidth-limited applications such as FDTD can profit from NVIDIA GPU BOOST™ technology, which allows you to run the GPU with higher clock speeds as long as the maximum power and thermal limits are not violated. For an overview of the available clock speeds, run the following command:

```
nvidia-smi -q -d SUPPORTED_CLOCKS
```

Chapter 9: Parallelization

Hardware Acceleration Using Graphics Processing Units

To list the current, default, and maximum GPU clock speeds, run the following command:

```
nvidia-smi -q -d CLOCK
```

With this information, you can set the clock speed to its maximum value:

```
nvidia-smi -ac <memory, graphics>
```

where `memory` is the clock speed of the memory and `graphics` is the clock speed of the streaming multiprocessor. Note that only supported pairs of values as shown by these query commands can be set. For example, to set the maximum clock speed of the P100 PCI-e GPU, run the following command:

```
nvidia-smi -ac 715,1328
```

To change clock speeds, root privileges are required unless your IT administrator sets the application clocks commands to unrestricted:

```
nvidia-smi -acp UNRESTRICTED
```

Note:

You must run the GPU in persistence mode to maintain the previously set clock speeds after a reboot of the server. This means that you must execute the `nvidia-smi -pm` command before changing any clock settings.

Running EMW GPU

To run EMW GPU on an NVIDIA® GPU accelerated server, use the following command:

```
emw -hw nvidia [-n #GPUs] <command_file>
```

By default, simulations on a single server are run on a single GPU. If more than one GPU accelerator is attached to the server, then EMW GPU selects one of them, typically the fastest one. However, you can also select a specific GPU device explicitly by using a UNIX environment variable (for example, C shell syntax):

```
setenv CUDA_DEVICE_ORDER PCI_BUS_ID  
setenv CUDA_VISIBLE_DEVICES 1
```

The first environment variable ensures that the specified index agrees with the ones listed by the `nvidia-smi` command. The second environment variable hides all GPUs except the second one listed by the `nvidia-smi` command (listing of GPU devices starts with 0), which can be verified using `nvidia-smi` before running the simulation. To run a simulation on more than one GPU, you must specify the `-n` command-line option, indicating the number of GPUs to be used.

For such single-server multi-GPU simulations, you can also select specific GPU devices if more devices than that requested are available using the following syntax:

```
setenv CUDA_VISIBLE_DEVICES "1,3"
```

Chapter 9: Parallelization

Hardware Acceleration Using Graphics Processing Units

Multi-GPU simulations on a single server do not require any other setup steps. However, for multi-GPU simulations with GPU accelerators distributed across different servers in a cluster, an MPI host file must be provided to specify on which servers the GPUs are located.

Multi-GPU simulations are essentially MPI simulations with the restriction that each MPI process must connect to exactly one GPU on the server on which the respective process is running. The GPU memory requirements are the same for each process and, therefore, the simulation size is ultimately limited by the GPU with the smallest memory: the total available GPU memory is computed by multiplying the number of GPUs used by the memory of the GPU with the smallest memory. The same setup steps for running EMW MPI on DP systems, as outlined in *TCAD Parallelization Environment Setup User Guide*, Running TCAD Sentaurus Tools on a Cluster, also apply to multi-GPU simulations, in particular, to the first two steps.

Note:

Do not specify the `-mpi` command-line option as it is implied by specifying more than one GPU, for both single-node and multinode simulations.

General Considerations and Recommendations

When launching EMW GPU on a GPU-accelerated server, ensure that the targeted GPU has sufficient memory for the part of the program that is off-loaded to the GPU. Essentially, the arrays holding the electric and magnetic vector fields, several coefficient arrays, and some further auxiliary data must reside in GPU memory during the FDTD time-stepping.

Note:

The peak GPU memory footprint during an EMW GPU simulation is considerably lower than that of a standard EMW simulation because only data required for the actual FDTD time-stepping is allocated on the GPU. Temporary data needed during preprocessing and postprocessing is kept on the host. So, the amount of CPU memory required by a standard EMW simulation is not a direct indicator of how much GPU memory is required for an equivalent EMW GPU simulation.

The GPU memory footprint is determined not only by the tensor grid size, but also by the chosen boundary conditions (for example, the CPML layer thickness) and the material models such as `PEC` or `DispersiveMedia`. In particular, dispersive media can increase the memory footprint considerably if they cover a large percentage of the total number of tensor cells. The performance measured as computational throughput can also be affected by the additional computation necessary for the dispersive models. However, for typical CMOS image sensor simulations at wavelengths longer than 0.3 μm , only the metal lines have to be modeled dispersively. This limits the additional memory requirements to approximately 10% of the total GPU memory consumption for nondispersive simulations and maintains the computational throughput at a comparable level.

[Table 14](#) provides an overview of FDTD problem sizes supported by available memory sizes of common NVIDIA® Tesla® GPU accelerators. For multi-GPU simulations, the

Chapter 9: Parallelization

Hardware Acceleration Using Graphics Processing Units

accumulated memory of all GPUs determines the maximum supported problem size as the simulation is distributed across different GPUs using the domain decomposition method.

Table 14 Overview of maximum supported FDTD problem sizes for different NVIDIA GPU models (given values are upper limit estimates for standard simulations without dispersive media modeling)

NVIDIA GPU model type	Available GPU memory	Maximum FDTD problem size
K80	2 × 12 GB	2 × 160 or 1 × 310 Mcells
K40	12 GB	160 Mcells
P100	16 GB	220 Mcells
V100	16 GB / 32 GB	220 Mcells / 440 Mcells
A100	40 GB	550 Mcells

Performance

The performance of a GPU simulation strongly depends on the occupancy of its streaming multiprocessors. For small FDTD problem sizes, there is not sufficient work to do to exploit the high level of parallelism available on a GPU, and the computational throughput will be lower than for larger problems. As a general rule, you can expect near-peak throughput for problem sizes that require at least 25% of the available GPU memory.

The theoretical throughput of multi-GPU simulations scales with the number of GPUs. However, due to the data exchange at subdomain boundaries, practical performance is slightly lower for single-node simulations and can vary strongly for multinode simulations depending on the interconnection bandwidth between the nodes of a cluster.

Note:

EMW GPU simulations are accelerated mainly by off-loading the computation of the FDTD time-stepping to the GPU. However, during preprocessing and postprocessing, several tasks are handled by the CPU of the host machine, which can take advantage of SMP. Therefore, it is recommended to set `NumberOfThreads=4` or larger for best overall simulation time even for EMW GPU simulations.

Monitoring

To monitor the use of GPU resources for an EMW GPU simulation, run the `nvidia-smi` command.

Chapter 9: Parallelization

Licensing for Different Parallel Modes

Without any command-line options, it provides some basic usage information about the visible GPUs (see the environment variable `CUDA_VISIBLE_DEVICES` in [Running EMW GPU on page 102](#)) such as memory consumption, power usage, temperature, and utilization along with the process ID of the process, if any, accessing a specific GPU.

Limitations

EMW GPU supports the most commonly used simulation setups for CMOS image sensor simulations and reports an error if unsupported features are specified such as:

- Lower-order absorbing boundary conditions (`Higdon` and `Mur`)
- Normal periodic boundary conditions
- CPML boundary conditions all around
- Code V excitation
- Discrete Fourier transform

Note:

In principle, you can run several EMW GPU simulations concurrently, which offload computations to a GPU. However, by default, offloaded computations from different simulations are scheduled sequentially on the GPU, that is, the total simulation time will be the same as if the simulations were run one after the other. It is recommended to run only one simulation job at a time on the same GPU.

Licensing for Different Parallel Modes

Different license features are available for running EMW in parallel mode. The type of license feature to be checked out depends on whether you run EMW in SMP parallel mode, DP parallel mode, mixed DP and SMP parallel mode, or hardware-accelerated parallel mode.

SMP Parallel Mode

In SMP parallel mode, EMW first tries to check out a license for DP parallelization as it entitles the user to run the program with an unlimited number of threads. If it fails because the license is not available or already in use by another simulation, EMW checks out one license for SMP parallelization for every multiple of four threads specified by the user. The advantage of this order of license-checking is that DP licenses will be consumed first as they are not applicable to any other TCAD Sentaurus tools. In contrast, SMP licenses also are used by other tools such as Sentaurus Device and Sentaurus Process.

DP Parallel Mode

Running EMW in DP parallel mode requires one license for DP parallelization for every multiple of four MPI processes. For example, an EMW simulation distributed over n nodes using 18 MPI processes checks out five licenses for DP parallelization.

Mixed DP and SMP Parallel Mode

Since one license for DP parallelization allows the use of an unlimited number of threads, the DP licensing scheme described in [DP Parallel Mode](#) also applies to the mixed DP and SMP parallel mode.

Hardware-Accelerated Parallel Mode

Running EMW on hardware-accelerated systems requires one license for DP parallelization for every accelerator used by the simulation. For example, an EMW simulation on a single node using two available GPUs on that node or an EMW simulation distributed over two nodes using one GPU available on each of the two nodes checks out two licenses for DP parallelization. Since one license for DP parallelization allows the use of an unlimited number of threads, hardware-accelerated EMW simulations can run with an arbitrary number of threads used for pre- and postprocessing without incurring additional license consumption.

Controlling Parallel Licenses

You can further control the license checkout behavior described for the SMP, DP, and hardware-accelerated parallel modes by specifying the following keywords in the `Globals` section:

- The keyword `ParallelLicenses` determines which licenses EMW is generally allowed to check out. This keyword supports the options `MPI` (to select the `emwmpi` license) and `SMP` (to select the `sdevice-parallel` license).
- The keyword `UnavailableLicenseAction` controls the behavior of EMW if none of the allowed licenses is available. [Table 15](#) lists the options for this keyword.

Chapter 9: Parallelization

References

Table 15 *Options for UnavailableLicenseAction if insufficient parallel licenses are available*

Option	Description
abort	Terminate the simulation.
reduce	Reduce the number of threads used in SMP parallel mode to match the number of available parallel licenses.
serial	(Default) Continue the simulation in serial mode.
wait	Wait until enough parallel licenses become available.

For SMP simulations, the default values are:

```
ParallelLicenses = {MPI, SMP}
UnavailableLicenseAction = serial
```

For DP simulations, the default values are:

```
ParallelLicenses = {MPI}
UnavailableLicenseAction = abort
```

For hardware-accelerated simulations, the default values are:

```
ParallelLicenses = {MPI}
UnavailableLicenseAction = abort
```

The following example shows how to avoid checking out an emwmpi license for an SMP parallel simulation and to terminate if the sdevice-parallel license is not available:

```
Globals {
    ParallelLicenses = {SMP}
    UnavailableLicenseAction = abort
}
```

The ability to control the parallel license behavior can be useful, for example, when several users run Sentaurus Device and EMW parallel simulations using the same pool of licenses and start them through Sentaurus Workbench.

References

- [1] For more information, go to <https://www.mpi-forum.org>.
- [2] For more information, go to <https://www.mpich.org>.
- [3] For more information, go to <https://www.nvidia.com>.

A

Commands

This appendix describes the command keywords available for use with Sentaurus Device Electromagnetic Wave Solver.

Boundary

This section is mandatory and can occur six times.

Table 16 *Keywords of Boundary section*

Keyword	Unit	Default [range]	Description
AlphaMax = <float>	$\Omega^{-1} \text{m}^{-1}$	0.2	Maximum value of PML parameter <code>alpha</code> used in polynomial grading of PML tensor coefficient.
AlphaProfile = <float> { <float>, ... }	$\Omega^{-1} \text{m}^{-1}$	—	Profile of PML parameter <code>alpha</code> used in grading of PML tensor coefficient. The number of points of the profile must be double the size of the (PML) <code>Thickness</code> parameter.
KappaMax = <float>	1	15	Maximum value of PML parameter <code>kappa</code> used in polynomial grading of PML tensor coefficient.
KappaProfile = <float> { <float>, ... }	1	—	Profile of PML parameter <code>kappa</code> used in grading of PML tensor coefficient. The number of points of the profile must be double the size of the (PML) <code>Thickness</code> parameter.
OrderAlpha = <integer>	1	1	Order of polynomial grading for PML parameter <code>alpha</code> used in polynomial grading of PML tensor coefficient.

Appendix A: Commands

Boundary

Table 16 Keywords of Boundary section (Continued)

Keyword	Unit	Default [range]	Description
OrderKappa = <integer>	1	3	Order of polynomial grading for PML parameter <code>kappa</code> used in polynomial grading of PML tensor coefficient.
OrderSigma = <integer>	1	3	Order of polynomial grading for PML parameter <code>sigma</code> used in polynomial grading of PML tensor coefficient.
Sides = <identifier> { <identifier>, ... }	1	{ All }	Sides to which the boundary specification applies. Exclusive options are: <ul style="list-style-type: none"> • All • X • Xmin • Xmax • Y • Ymin • Ymax • Z • Zmin • Zmax
SigmaMax = <float>	$\Omega^{-1} \text{m}^{-1}$	—	Maximum value of PML parameter <code>sigma</code> used in polynomial grading of PML tensor coefficient. The default value is computed automatically from the grid spacing and refractive index.
SigmaProfile = <float> { <float>, ... }	$\Omega^{-1} \text{m}^{-1}$	—	Profile of PML parameter <code>sigma</code> used in grading of PML tensor coefficient. The number of points of the profile must be double the size of the (PML) <code>Thickness</code> parameter.
Thickness = <integer>	1	15	Thickness of PML boundary in terms of number of tensor-grid cells.

Table 16 Keywords of Boundary section (Continued)

Keyword	Unit	Default [range]	Description
Type = <identifier>	1	Periodic Oblique	Type of boundary condition. Exclusive options are: <ul style="list-style-type: none"> • CPML • Higdon • Mur • PEC • Periodic • PeriodicOblique • PMC

CodeVExcitation

This section is optional and can occur only once.

Table 17 Keywords of CodeVExcitation section

Keyword	Unit	Default [range]	Description
AmplitudeScaling = <float>	1	1	Scaling factor for electric field amplitude.
AnchorPoint1 = { <float>, <float>, <float> }	μm	—	Anchor point (of smallest coordinates) for the first data file.
AnchorPoint2 = { <float>, <float>, <float> }	μm	—	Anchor point (of smallest coordinates) for the second data file.
Delay = <float>	1	0	Number of signal periods before signal starts.
Frequency = <float>	s^{-1}	—	Signal frequency.
InputDataFile = "<string>" { "<string>", ... }	1	—	Names of two input data files from CODE V.
NRise = <float>	1	2	Number of signal periods before full amplitude is reached.
PhaseShift = <float>	deg	0	Linear phase shift for electric field.

Table 17 Keywords of CodeVExcitation section (Continued)

Keyword	Unit	Default [range]	Description
Signal = <identifier>	1	Harmonic	Sets transient form of the signal. Exclusive options are: <ul style="list-style-type: none"> • Gauss • Harmonic
Wavelength = <float>	nm	—	Signal wavelength.

ComplexRefractiveIndex

This section is optional and can occur multiple times.

Table 18 Keywords of ComplexRefractiveIndex section

Keyword	Unit	Default [range]	Description
CarrierDep = <identifier> { <identifier>, ... }	1	—	Controls carrier dependency of real and imaginary parts of complex refractive index. Exclusive options are: <ul style="list-style-type: none"> • Real • Imag
CRIModel = "<string>"	1	—	Name of user-provided complex refractive index model.
Extrapolate = <identifier>	1	No	Controls extrapolation behavior for interpolation of datasets in FieldDataFile specified in Globals section. Exclusive options are: <ul style="list-style-type: none"> • Yes • No
GainDep = <identifier> { <identifier>, ... }	1	—	Controls gain dependency of real part of complex refractive index. Exclusive options are: <ul style="list-style-type: none"> • RealLin • RealLog

Table 18 Keywords of ComplexRefractiveIndex section (Continued)

Keyword	Unit	Default [range]	Description
IgnoreMaterials = <identifier>	1	No	Controls whether interpolation between vertices of different materials is permitted when loading datasets from <code>FieldDataFile</code> specified in <code>Globals</code> section. Exclusive options are: <ul style="list-style-type: none"> • Yes • No
Material = "<string>" { "<string>", ... }	1	–	Names of materials to which the complex refractive index specification applies.
Region = "<string>" { "<string>", ... }	1	–	Names of regions to which the complex refractive index specification applies.
TemperatureDep = <identifier> { <identifier>, ... }	1	–	Controls temperature dependency of real part of complex refractive index. Exclusive options are: <ul style="list-style-type: none"> • Real • Imag
Type = <identifier>	1	Homogeneous	Controls type of complex refractive index distribution. Users can choose between a homogeneous (regionwise constant complex refractive index) and an inhomogeneous (complex refractive index may vary within a region) distribution, which is computed based on dependent data fields such as a temperature profile. Another option is to directly load an external complex refractive index profile. Exclusive options are: <ul style="list-style-type: none"> • Homogeneous • Inhomogeneous • FromFile
WavelengthDep = <identifier> { <identifier>, ... }	1	–	Controls wavelength dependency of real and imaginary parts of complex refractive index. Exclusive options are: <ul style="list-style-type: none"> • Real • Imag

Detector

This section is optional and can occur only once.

Table 19 Keywords of Detector section

Keyword	Unit	Default [range]	Description
BoxCorner1 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box that specifies the detector domain. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
BoxCorner2 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box that specifies the detector domain. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
BreakOnError = <identifier>	1	Yes	Specifies whether the simulation stops if an error (for example, floating-point exception) is encountered during the simulation. Final <code>Plot</code> or <code>Extractor</code> sections will still be executed. Exclusive options are: <ul style="list-style-type: none"> • Yes • No
FailureOnUnreached Tolerance = <identifier>	1	Yes	Specifies whether the exit code of the program will be set to a nonzero value if the simulation has not reached the specified tolerance. Exclusive options are: <ul style="list-style-type: none"> • Yes • No
Frequency = <float>	s^{-1}	–	Detector frequency. The default value is calculated from the excitation wavelength.
Region = "<string>"	1	–	Valid box or region name of the tensor grid where the detector is evaluated.
StartTick = <integer>	1	0	Number of tick steps when to start the detector evaluation.
StartTime = <float>	s	0	Simulation time when to start the detector evaluation.
StepX = <integer>	1	10	Detection is performed on every <integer>-th grid point in the x-direction only.

Appendix A: Commands

DFT

Table 19 Keywords of Detector section (Continued)

Keyword	Unit	Default [range]	Description
StepY = <integer>	1	10	Detection is performed on every <integer>-th grid point in the y-direction only.
StepZ = <integer>	1	10	Detection is performed on every <integer>-th grid point in the z-direction only.
Tolerance = <float>	1	0.001	The simulation is terminated when the maximum deviation is smaller than <float>.
Type = <identifier>	1	Periodic	Type of detector. Exclusive options are: <ul style="list-style-type: none">• Periodic• Decayed

DFT

This section is optional and can occur only once.

Table 20 Keywords of DFT section

Keyword	Unit	Default [range]	Description
DownSampling = <integer>	1	–	Downsampling rate of DFT.
MaxWavelength = <float>	1	–	End wavelength of the DFT wavelength table.
MinWavelength = <float>	1	–	Start wavelength of the DFT wavelength table.
NumberOfWavelengths = <integer>	1	–	Number of elements in the DFT wavelength table.

DispersiveMedia

This section is optional and can occur multiple times.

Table 21 Keywords of DispersiveMedia section

Keyword	Unit	Default [range]	Description
DeltaK = <float>	1	0	Activate dispersive model for $k + \text{DeltaK} \geq n$.
ExcludeMaterial = " <string>" { " <string>", ... }	1	—	Names of materials to which dispersive medium specification does not apply.
ExcludeRegion = " <string>" { " <string>", ... }	1	—	Names of regions to which dispersive medium specification does not apply.
InterfaceAveraging = <identifier>	1	No	Controls whether interface averaging is granted for dispersive regions or materials specified in this section. Exclusive options are: <ul style="list-style-type: none"> • Yes • No
Material = " <string>" { " <string>", ... }	1	—	Names of materials to which the dispersive medium specification applies.
Model = <identifier>	1	—	Selects dispersive model. Exclusive options are: <ul style="list-style-type: none"> • Debye • Drude • DrudeLorentz • DrudeModLorentz • Lorentz • ModLorentz • SingleDipoleDebye • SingleDipoleDrude
ModelParameters = <identifier>	1	ComputeFrom Complex Refractive Index	Specifies how dispersive model parameters are determined. Exclusive options are: <ul style="list-style-type: none"> • ComputeFromComplexRefractiveIndex • UserDefined

Appendix A: Commands

Extractor

Table 21 Keywords of DispersiveMedia section (Continued)

Keyword	Unit	Default [range]	Description
Region = "<string>" { "<string>", ... }	1	–	Names of regions to which the dispersive medium specification applies.

Extractor

This section is optional and can occur multiple times.

Table 22 Keywords of Extractor section

Keyword	Unit	Default [range]	Description
BoxCorner1 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box for which data must be extracted. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
BoxCorner2 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box for which data must be extracted. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
GridFile = "<string>"	1	–	Name of mixed-element grid file for output data.
Interpolation = <identifier>	1	Standard	Way of interpolating extraction data from tensor grid to the mixed-element grid. Exclusive options are: <ul style="list-style-type: none">• Standard• Simple• Conservative
Name = "<string>"	1	–	Defines the name of the extractor, which also is used for the output file name by appending <code>_eml.tdr</code> .
PlaneX = <float> <identifier>	μm	–	Extractor domain specified as a line (in two dimensions) or a plane (in three dimensions) perpendicular to the x-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.

Appendix A: Commands

Extractor

Table 22 Keywords of Extractor section (Continued)

Keyword	Unit	Default [range]	Description
PlaneY = <float> <identifier>	μm	–	Extractor domain specified as a line (in two dimensions) or a plane (in three dimensions) perpendicular to the y-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
PlaneZ = <float> <identifier>	μm	–	Extractor domain specified as a plane in three dimensions perpendicular to the z-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
Quantity = <identifier> { <identifier>, ... }	1	–	Specifies the quantities to extract. Exclusive options are: <ul style="list-style-type: none"> • AbsorbedPhotonDensity • AbsorbedPowerDensity • OpticalGeneration • PowerFluxDensity • AbsElectricField • RealElectricField • RealElectricFieldX • RealElectricFieldY • RealElectricFieldZ • ImagElectricField • ImagElectricFieldX • ImagElectricFieldY • ImagElectricFieldZ • RealMagneticField • RealMagneticFieldX • RealMagneticFieldY • RealMagneticFieldZ • ImagMagneticField • ImagMagneticFieldX • ImagMagneticFieldY • ImagMagneticFieldZ • Region

Appendix A: Commands

Farfield

Table 22 Keywords of Extractor section (Continued)

Keyword	Unit	Default [range]	Description
Region = "<string>" { "<string>", ... }	1	–	Regions for which data must be extracted. If not specified, data for the whole simulation domain will be extracted.
ScatteredFieldOnly = <identifier>	1	No	Specifies whether the extractor involves only scattered fields. Exclusive options are: <ul style="list-style-type: none">• Yes• No
WavelengthList = <float> { <float>, ... }	nm	–	Specifies a list of wavelengths to be used by the extractor for DFT-type simulations.
WavelengthPrecision = <integer>	1	0	Specifies the precision used for matching wavelengths given by WavelengthList and DFT wavelengths resulting from the specification in the DFT section. It also controls the number of digits after the decimal point displayed in the wavelength-identifying part of the extractor file name.
WavelengthRange = { <float>, <float> }	nm	–	Specifies the wavelength range to be used by the extractor for DFT-type simulations given by an interval (<min_wavelength>, <max_wavelength>).

Farfield

This section is optional and can occur multiple times.

Table 23 Keywords of Farfield section

Keyword	Unit	Default [range]	Description
ArraySize = { <integer>, <integer> }	1	{ 1 1 }	Number of times the near-field domain is replicated in each coordinate direction of the 2D plane of the near field.

Table 23 Keywords of Farfield section (Continued)

Keyword	Unit	Default [range]	Description
AzimuthalCutList = <float> { <float>, ... }	deg	–	Defines an explicit list of azimuthal angles. For each angle, an xy plot with constant azimuthal angle is created, whose x-axis denotes the polar angle and the y-axis denotes the far-field intensity. The list is in the format {AzimuthalAngle1, AzimuthalAngle2, ...}.
AzimuthalCutRange = { <float>, <float>, <float> }	deg	–	Defines a list of azimuthal angles. For each angle, an xy plot with constant azimuthal angle is created, whose x-axis denotes the polar angle and the y-axis denotes the far-field intensity. The list is in the format {MinAzimuthalAngle, MaxAzimuthalAngle, AzimuthalAngleStep}.
AzimuthalRange = { <float>, <float>, <float> }	deg	{0 360 1}	Range and discretization of azimuthal angles used for far-field computation. The list is in the format {MinAzimuthalAngle, MaxAzimuthalAngle, AzimuthalAngleStep}.
BoxCorner1 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box defining the near field to be used for the far-field computation. The specified box must be a plane in the 3D simulation domain. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
BoxCorner2 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box defining the near field to be used for the far-field computation. The specified box must be a plane in the 3D simulation domain. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.

Appendix A: Commands

Farfield

Table 23 Keywords of Farfield section (Continued)

Keyword	Unit	Default [range]	Description
IntegrationRange = { <float>, <float>, <float>, <float> } { { <float>, <float> }, <float>, <float>, <float> }, ... }	deg	–	Specifies a list of integration ranges and their discretization for computing the power integration. Each element of the list is in the format of {MinAzimuthalAngle, MaxAzimuthalAngle, AzimuthalAngleStep, MinPolarAngle, MaxPolarAngle, PolarAngleStep}. Note that the total power is always computed, and the corresponding integration range does not have to be specified explicitly.
Name = "<string>"	1	–	Defines the name of the far field, which also is used for the output file name by appending _eml.tdr.
PolarCutList = <float> { <float>, ... }	deg	–	Defines an explicit list of polar angles. For each angle, an xy plot with constant polar angle is created, whose x-axis denotes the azimuthal angle and the y-axis denotes the far-field intensity. The list is in the format {PolarAngle1, PolarAngle2, ...}.
PolarCutRange = { <float>, <float>, <float> }	deg	–	Defines a list of polar angles. For each angle, an xy plot with constant polar angle is created whose x-axis denotes the azimuthal angle and the y-axis denotes the far-field intensity. The list is in the format {MinPolarAngle, MaxPolarAngle, PolarAngleStep}.
PolarRange = { <float>, <float>, <float> }	deg	{ 0 90 1 }	Range and discretization of polar angles used for far-field computation. The list is in the format {MinPolarAngle, MaxPolarAngle, PolarAngleStep}.
Radius = <float>	μm	1000	Radius (distance) at which the far field is calculated.
Region = "<string>"	1	–	Near-field region to be used for the far-field computation. The specified region must be a plane in the 3D simulation domain.

Table 23 Keywords of Farfield section (Continued)

Keyword	Unit	Default [range]	Description
WavelengthList = <float> { <float>, ... }	nm	–	Specifies a list of wavelengths to be used by the far field for DFT-type simulations.
WavelengthPrecision = <integer>	1	0	Specifies the precision used for matching wavelengths given by WavelengthList and DFT wavelengths resulting from the specification in the DFT section. It also controls the number of digits after the decimal point displayed in the wavelength-identifying part of the far-field file name.
WavelengthRange = { <float>, <float> }	nm	–	Specifies the wavelength range to be used by the far field for DFT-type simulations given by an interval (<min_wavelength>, <max_wavelength>).

GaussianBeamExcitation

This section is optional and can occur only once.

Table 24 Keywords of GaussianBeamExcitation section

Keyword	Unit	Default [range]	Description
Amplitude = <float>	V m ⁻¹	1	Signal amplitude.
BeamCenter = { <float>, <float>, <float> }	μm	–	Coordinates of the center (focus) of the Gaussian beam.
BeamPower = <float>	W	2.08478e-15	Signal power of the Gaussian beam.
BeamRadius = <float>	μm	–	Radius (waist) of the Gaussian beam.
BoxCorner1 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box that specifies the excitation plane. Use min or max for the minimum or maximum device extents along the corresponding coordinate axis.

Appendix A: Commands

GaussianBeamExcitation

Table 24 Keywords of GaussianBeamExcitation section (Continued)

Keyword	Unit	Default [range]	Description
BoxCorner2 = { <float> min max, <float> min max, <float> min max }	μm	—	One of the two corners of a box that specifies the excitation plane. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
Delay = <float>	1	0	Number of signal periods before signal starts.
Direction = <identifier>	1	—	Illumination direction with respect to the coordinate system of the input tensor grid. This keyword automatically sets the excitation angles <code>Theta</code> , <code>Phi</code> , and <code>Psi</code> as well as the excitation plane <code>Plane [X,Y,Z]</code> . Exclusive options are: <ul style="list-style-type: none"> • <code>fromTop</code> • <code>fromBottom</code>
EllipticalPolarization Magnitude = <float>	1	0	Polarization ratio magnitude.
EllipticalPolarizationPhase = <float>	deg	0	Polarization ratio phase.
Frequency = <float>	s^{-1}	—	Signal frequency.
Intensity = <float>	Wcm^{-2}	1.32721e-07	Signal intensity.
NRise = <float>	1	2	Number of signal periods before full amplitude is reached.
Phi = <float>	deg	0	Azimuth angle of incidence.
PlaneX = <float> <identifier>	μm	—	Excitation line (in two dimensions) or plane (in three dimensions) perpendicular to the x-axis. For <code><identifier></code> , use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.

Table 24 Keywords of GaussianBeamExcitation section (Continued)

Keyword	Unit	Default [range]	Description
PlaneY = <float> <identifier>	μm	—	Excitation line (in two dimensions) or plane (in three dimensions) perpendicular to the y-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
PlaneZ = <float> <identifier>	μm	—	Excitation plane in three dimensions perpendicular to the z-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
Polarization = <float> <identifier>	1	0.5	Polarization, either <code>TE</code> or <code>TM</code> , or a value between 0 and 1.
PolarizationAngle = <float>	deg	45	Polarization angle.
Psi = <float>	deg	45	Polarization angle.
Region = "<string>"	1	—	Box that specifies the excitation plane.
Sigma = <float>	1	1	Width of transient Gaussian pulse in terms of number of signal periods.
Signal = <identifier>	1	Harmonic	Sets the transient form of the signal. Exclusive options are: <ul style="list-style-type: none"> • Gauss • Harmonic
Theta = <float>	deg	180	Polar angle of incidence.
Wavelength = <float>	nm	—	Signal wavelength.

Globals

This section is mandatory and must occur only once.

Table 25 Keywords of Globals section

Keyword	Unit	Default [range]	Description
CompressTDR = <identifier>	1	No	Controls whether output TDR files are compressed beyond the basic compression level. Exclusive options are: <ul style="list-style-type: none"> • Yes • No
CRIMIPath = "<string>"	1	.	Search path for shared objects used by complex refractive index model interface.
DomainDecomposition = <identifier>	1	Automatic	Overrides automatic MPI domain decomposition. Used to enforce domain decomposition in only one Cartesian direction. Exclusive options are: <ul style="list-style-type: none"> • Automatic • X • Y • Z
FieldDataFile = "<string>"	1	—	Data file containing datasets, for example, spatial temperature profile, on which the complex refractive index depends.
GridFile = "<string>"	1	—	Tensor grid file for the simulation.
LoadFile = "<string>"	1	—	Name of the results file to be loaded for postprocessing.
LogFile = "<string>"	1	—	Name of the log file. By default, the name of the command file is used with the extension <code>.log</code> . If the name ends with <code>.z</code> or <code>.gz</code> , a compressed log file will be written.
NumberOfThreads = <integer> <identifier>	1	1	Number of threads to be used for shared-memory parallelization. Either <code>NumberOfThreads=maximum</code> OR <code>NumberOfThreads=<integer greater than zero></code> .

Appendix A: Commands

Globals

Table 25 Keywords of Globals section (Continued)

Keyword	Unit	Default [range]	Description
OutputFilePrefix = "<string>"	1	" "	String to be prepended to all output file names including the log file.
OverSampling = <float>	1	1	Reduces the calculated maximum stable time step by the <code>OverSampling</code> factor.
ParallelLicenses = identifier> { <identifier>, ... }	1	{MPI SMP}	List of allowed licenses to be used by EMW when run in parallel mode. The default value for simulations started with <code>emw -mpi</code> is <code>ParallelLicenses = {MPI}</code> . Exclusive options are: <ul style="list-style-type: none"> • MPI • SMP
ParameterFile = "<string>"	1	—	Specifies the material parameter file.
ResultFile = "<string>"	1	result_ eml.plt	Name of result file.
TdrNativeTensorGridUnit = <identifier>	1	Automatic	Overrides automatic detection of length unit in input tensor grid. Its use may be necessary for tensor grids in [μm] with very small cell sizes or for tensor grids in [m] with very large cell sizes. Exclusive options are: <ul style="list-style-type: none"> • Automatic • m • um
Temperature = <float>	K	300	Global device temperature.
TimeStep = <float>	s	—	Overwrites calculated simulation time step.
TotalSimulationTime = <float>	s	+inf	Maximum simulation time after which the simulation is terminated.
TotalTimeSteps = <integer>	1	+inf	Maximum number of time steps to be performed.

Appendix A: Commands

Monitor

Table 25 Keywords of Globals section (Continued)

Keyword	Unit	Default [range]	Description
UnavailableLicenseAction = <identifier>	1	serial	Action to be taken when requested license is not available. The default value for simulations started with <code>emw -mpi</code> is <code>abort</code> . Exclusive options are: <ul style="list-style-type: none">• serial• reduce• wait• abort

Monitor

This section is optional and can occur multiple times.

Table 26 Keywords of Monitor section

Keyword	Unit	Default [range]	Description
EndTick = <integer>	1	+inf	Specifies the end of the active tick step interval.
EndTime = <float>	s	+inf	Specifies the end of the active time interval.
Frequency = <float>	s ⁻¹	—	Monitor frequency. The default value is calculated from the excitation wavelength.
Location = { <float>, <float>, <float> } { { <float>, <float>, <float> }, ... }	μm	—	Locations to monitor.
Name = "<string>"	1	—	Defines the name of the monitor, which also is used in the dataset name in the result file.

Table 26 Keywords of Monitor section (Continued)

Keyword	Unit	Default [range]	Description
Quantity = <identifier> { <identifier>, ... }	1	{AbsElectric Field}	Specifies the quantities to monitor. Exclusive options are: <ul style="list-style-type: none"> • AbsElectricField • CplxEMField • CplxElectricField • RealElectricField • RealElectricFieldX • RealElectricFieldY • RealElectricFieldZ • ImagElectricField • ImagElectricFieldX • ImagElectricFieldY • ImagElectricFieldZ • AbsMagneticField • CplxMagneticField • RealMagneticField • RealMagneticFieldX • RealMagneticFieldY • RealMagneticFieldZ • ImagMagneticField • ImagMagneticFieldX • ImagMagneticFieldY • ImagMagneticFieldZ
StartTick = <integer>	1	0	Specifies the beginning of the active tick step interval.
StartTime = <float>	s	0	Specifies the beginning of the active time interval.
TickStep = <integer>	1	—	Monitoring is performed every <integer> tick step only.
WavelengthList = <float> { <float>, ... }	nm	—	Specifies a list of wavelengths to be used by the monitor for DFT-type simulations.

Table 26 Keywords of Monitor section (Continued)

Keyword	Unit	Default [range]	Description
WavelengthPrecision = <integer>	1	0	Specifies the precision used for matching wavelengths given by WavelengthList and DFT wavelengths resulting from the specification in the DFT section. It also controls the number of digits after the decimal point displayed in the wavelength-identifying part of the monitor name in the result file.
WavelengthRange = { <float>, <float> }	nm	—	Specifies the wavelength range to be used by the plot for DFT-type simulations given by an interval (<min_wavelength>, <max_wavelength>).

PECMedia

This section is optional and can occur multiple times.

Table 27 Keywords of PECMedia section

Keyword	Unit	Default [range]	Description
ExcludeMaterial = "<string>" { "<string>", ... }	1	—	Names of materials to which PEC medium specification does not apply.
ExcludeRegion = "<string>" { "<string>", ... }	1	—	Names of regions to which PEC medium specification does not apply.
Material = "<string>" { "<string>", ... }	1	—	Names of materials to which the PEC medium specification applies.
Region = "<string>" { "<string>", ... }	1	—	Names of regions to which the PEC medium specification applies.

PlaneWaveExcitation

This section is optional and can occur only once.

Table 28 Keywords of PlaneWaveExcitation section

Keyword	Unit	Default [range]	Description
Amplitude = <float>	Vm^{-1}	1	Signal amplitude.
BoxCorner1 = { <float> min max, <float> min max, <float> min max }	μm	—	One of the two corners of a box that specifies the excitation plane. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
BoxCorner2 = { <float> min max, <float> min max, <float> min max }	μm	—	One of the two corners of a box that specifies the excitation plane. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
Delay = <float>	1	0	Number of signal periods before signal starts.
Direction = <identifier>	1	—	Illumination direction with respect to the coordinate system of the input tensor grid. This keyword automatically sets the excitation angles <code>Theta</code> , <code>Phi</code> , and <code>Psi</code> as well as the excitation plane <code>Plane [X,Y,Z]</code> . Exclusive options are: <ul style="list-style-type: none"> • <code>fromTop</code> • <code>fromBottom</code>
EllipticalPolarization Magnitude = <float>	1	0	Polarization ratio magnitude.
EllipticalPolarizationPhase = <float>	deg	0	Polarization ratio phase.
Frequency = <float>	s^{-1}	—	Signal frequency.
Intensity = <float>	Wcm^{-2}	1.32721e-07	Signal intensity.

Appendix A: Commands

PlaneWaveExcitation

Table 28 *Keywords of PlaneWaveExcitation section (Continued)*

Keyword	Unit	Default [range]	Description
NRise = <float>	1	2	Number of signal periods before full amplitude is reached.
NumberOfPlaneWaves = <integer>	1	1	Number of excitation plane waves.
NumericalAperture = <float>	1	0	Numerical aperture.
PartialCoherenceFactor = <float>	1	1	Partial coherence factor.
Phi = <float>	deg	0	Azimuth angle of incidence.
PlaneX = <float> <identifier>	μm	—	Excitation line (in two dimensions) or plane (in three dimensions) perpendicular to the x-axis. For <identifier>, use either min or max for the minimum or maximum device extents along the corresponding coordinate axis.
PlaneY = <float> <identifier>	μm	—	Excitation line (in two dimensions) or plane (in three dimensions) perpendicular to the y-axis. For <identifier>, use either min or max for the minimum or maximum device extents along the corresponding coordinate axis.
PlaneZ = <float> <identifier>	μm	—	Excitation plane in three dimensions perpendicular to the z-axis. For <identifier>, use either min or max for the minimum or maximum device extents along the corresponding coordinate axis.
Polarization = <float> <identifier>	1	0.5	Polarization, either TE or TM, or a value between 0 and 1.
PolarizationAngle = <float>	deg	45	Polarization angle.
Psi = <float>	deg	45	Polarization angle.
Region = "<string>"	1	—	Box that specifies the excitation plane.

Appendix A: Commands

Plot

Table 28 Keywords of PlaneWaveExcitation section (Continued)

Keyword	Unit	Default [range]	Description
Sigma = <float>	1	1	Width of transient Gaussian pulse in terms of number of signal periods.
Signal = <identifier>	1	Harmonic	Sets the transient form of the signal. Exclusive options are: <ul style="list-style-type: none"> • Gauss • Harmonic
Theta = <float>	deg	180	Polar angle of incidence.
Wavelength = <float>	nm	—	Signal wavelength.

Plot

This section is optional and can occur multiple times.

Table 29 Keywords of Plot section

Keyword	Unit	Default [range]	Description
BoxCorner1 = { <float> min max, <float> min max, <float> min max }	μm	—	One of the two corners of a box for which data must be plotted. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
BoxCorner2 = { <float> min max, <float> min max, <float> min max }	μm	—	One of the two corners of a box for which data must be plotted. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
Domain = <identifier>	1	Volume	Specifies whether the values for the volume or only for the surface will be plotted. In the latter case, the sidewalls can be selected with <code>Sides</code> . Exclusive options are: <ul style="list-style-type: none"> • Surface • Volume

Appendix A: Commands

Plot

Table 29 Keywords of Plot section (Continued)

Keyword	Unit	Default [range]	Description
EndTick = <integer>	1	+inf	Specifies the end of the active tick step interval.
EndTime = <float>	s	+inf	Specifies the end of the active time interval.
FinalPlot = <identifier>	1	Yes	Controls whether at the end of the simulation a final plot is created. Exclusive options are: <ul style="list-style-type: none"> • Yes • No
GridFile = "<string>"	1	—	Name of mixed-element grid file for output data.
Interpolation = <identifier>	1	Standard	Method of interpolating plot data from the tensor grid to the mixed-element grid. Exclusive options are: <ul style="list-style-type: none"> • Standard • Simple • Conservative
Name = "<string>"	1	—	Defines the name of the plot, which also is used for the output file name by appending <code>_eml.tdr</code> .
PlaneX = <float> <identifier>	μm	—	Plot domain specified as a line (in two dimensions) or a plane (in three dimensions) perpendicular to the x-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
PlaneY = <float> <identifier>	μm	—	Plot domain specified as a line (in two dimensions) or a plane (in three dimensions) perpendicular to the y-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.

Appendix A: Commands

Plot

Table 29 Keywords of Plot section (Continued)

Keyword	Unit	Default [range]	Description
PlaneZ = <float> <identifier>	μm	–	Plot domain specified as a plane in three dimensions perpendicular to the z-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
Quantity = <identifier> { <identifier>, ... }	1	{AbsElectric Field}	<p>Sets the list of quantities to be plotted. Exclusive options are:</p> <ul style="list-style-type: none"> • AbsElectricDisplacement • AbsElectricField • CplxEMField • CplxElectricField • RealElectricField • RealElectricFieldX • RealElectricFieldY • RealElectricFieldZ • ImagElectricField • ImagElectricFieldX • ImagElectricFieldY • ImagElectricFieldZ • AbsMagneticField • CplxMagneticField • RealMagneticField • RealMagneticFieldX • RealMagneticFieldY • RealMagneticFieldZ • ImagMagneticField • ImagMagneticFieldX • ImagMagneticFieldY • ImagMagneticFieldZ • AbsMagneticInduction • ConductionCurrentDensity • ElectricConductivityXX • ElectricConductivityYY • ElectricConductivityZZ

Appendix A: Commands

Plot

Table 29 Keywords of Plot section (Continued)

Keyword	Unit	Default [range]	Description
Quantity = <identifier> { <identifier>, ... }	1	{AbsElectric Field}	<ul style="list-style-type: none"> • MagneticConductivityXX • MagneticConductivityYY • MagneticConductivityZZ • RelativePermeabilityXX • RelativePermeabilityYY • RelativePermeabilityZZ • RelativePermittivityXX • RelativePermittivityYY • RelativePermittivityZZ • cplxRefIndex • CplxRefIndexXX • CplxRefIndexYY • CplxRefIndexZZ • cplxExtCoeff • CplxExtCoeffXX • CplxExtCoeffYY • CplxExtCoeffZZ • IncoherentRealEx • IncoherentRealEy • IncoherentRealEz • IncoherentRealHx • IncoherentRealHy • IncoherentRealHz • IncoherentImagEx • IncoherentImagEy • IncoherentImagEz • IncoherentImagHx • IncoherentImagHy • IncoherentImagHz • IncoherentIntensity • IncoherentOpticalGeneration • IncoherentAbsorbedPhotonDensity • Region
Region = "<string>" { "<string>", ... }	1	–	Regions for which data must be plotted. If not specified, the whole simulation domain will be plotted.

Appendix A: Commands

Plot

Table 29 Keywords of Plot section (Continued)

Keyword	Unit	Default [range]	Description
ScatteredFieldOnly = <identifier>	1	No	Specifies whether the plot involves only scattered fields. Exclusive options are: <ul style="list-style-type: none"> • Yes • No
Sides = <identifier> { <identifier>, ... }	1	{ All }	When Domain = Surface, Sides defines the sidewall where plotting will be performed. Exclusive options are: <ul style="list-style-type: none"> • All • X • Xmin • Xmax • Y • Ymin • Ymax • Z • Zmin • Zmax
StartTick = <integer>	1	+inf	Specifies the beginning of the active tick step interval.
StartTime = <float>	s	+inf	Specifies the beginning of the active time interval.
TickStep = <integer>	1	100	Plotting is performed every <integer> tick step only.
WavelengthList = <float> { <float>, ... }	nm	—	Specifies a list of wavelengths to be used by the plot for DFT-type simulations.
WavelengthPrecision = <integer>	1	0	Specifies the precision used for matching wavelengths given by WavelengthList and DFT wavelengths resulting from the specification in the DFT section. It also controls the number of digits after the decimal point displayed in the wavelength-identifying part of the plot file name.

Table 29 Keywords of Plot section (Continued)

Keyword	Unit	Default [range]	Description
WavelengthRange = { <float>, <float> }	nm	–	Specifies the wavelength range to be used by the plot for DFT-type simulations given by an interval (<min_wavelength>, <max_wavelength>).
WriteBehavior = <identifier>	1	Numbered	<p>Controls how consecutive plot instances are written to file. Exclusive options are:</p> <ul style="list-style-type: none"> • Numbered • Overwrite • OverwriteAndNumbered <p>Use <code>Numbered</code> to include a counter in the file name, which increases with each plot instance.</p> <p>Use <code>Overwrite</code> to have consecutive plot instances overwrite the same file.</p> <p>Use <code>OverwriteAndNumbered</code> to constantly overwrite a file containing the latest plot as well as to have a collection of all plots in separate files distinguished by a counter in the file name.</p>

PMCMedia

This section is optional and can occur multiple times.

Table 30 Keywords of PMCMedia section

Keyword	Unit	Default [range]	Description
ExcludeMaterial = "<string>" { "<string>", ... }	1	–	Names of materials to which PMC medium specification does not apply.
ExcludeRegion = "<string>" { "<string>", ... }	1	–	Names of regions to which PMC medium specification does not apply.

Appendix A: Commands

RTA

Table 30 Keywords of PMCMedia section (Continued)

Keyword	Unit	Default [range]	Description
Material = "<string>" { "<string>", ... }	1	–	Names of materials to which the PMC medium specification applies.
Region = "<string>" { "<string>", ... }	1	–	Names of regions to which the PMC medium specification applies.

RTA

This section is optional and can occur only once.

Table 31 Keywords of RTA section

Keyword	Unit	Default [range]	Description
EndTick = <integer>	1	+inf	Specifies the end of the active tick step interval.
EndTime = <float>	s	+inf	Specifies the end of the active time interval.
Log = <identifier> { <identifier>, ... }	1	{all}	Specifies quantities that are written as design-of-experiments parameters to the log file and that appear in the result table of Sentaurus Workbench. Exclusive options are: <ul style="list-style-type: none">• R• T• A• RTA• Rnorm• Tnorm• Anorm• all• none
LogPrecision = <integer>	1	3	Specifies the number of digits to the right of the decimal point of the design-of-experiments parameters written to the log file.

Appendix A: Commands

Save

Table 31 Keywords of RTA section (Continued)

Keyword	Unit	Default [range]	Description
Sensors = <identifier> { <identifier>, ... }	1	{all}	Type of sensors to be activated. Exclusive options are: <ul style="list-style-type: none">• R• T• A• all
StartTick = <integer>	1	+inf	Specifies the beginning of the active tick step interval.
StartTime = <float>	s	+inf	Specifies the beginning of the active time interval.
TickStep = <integer>	1	100	Sensing is performed every <integer> tick step only.
WavelengthList = <float> { <float>, ... }	nm	–	Specifies a list of wavelengths to be used by the sensor for DFT-type simulations.
WavelengthPrecision = <integer>	1	0	Specifies the precision used for matching wavelengths given by WavelengthList and DFT wavelengths resulting from the specification in the DFT section.
WavelengthRange = { <float>, <float> }	nm	–	Specifies the wavelength range to be used by the sensor for DFT-type simulations given by an interval (<min_wavelength>, <max_wavelength>).

Save

This section is optional and can occur multiple times.

Table 32 Keywords of Save section

Keyword	Unit	Default [range]	Description
BoxCorner1 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box for which data must be saved. Use min or max for the minimum or maximum device extents along the corresponding coordinate axis.

Appendix A: Commands

Save

Table 32 Keywords of Save section (Continued)

Keyword	Unit	Default [range]	Description
BoxCorner2 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box for which data must be saved. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
Name = "<string>"	1	–	Defines the name of the <code>Save</code> section, which also is used for the output file name by appending <code>_eml.tdr</code> .
Quantity = <identifier> { <identifier>, ... }	1	–	<p>Specifies the quantities to extract. Exclusive options are:</p> <ul style="list-style-type: none"> • AbsorbedPhotonDensity • IncoherentAbsorbedPhotonDensity • PowerFluxDensity • CplxEMField • CplxElectricField • RealElectricField • RealElectricFieldX • RealElectricFieldY • RealElectricFieldZ • ImagElectricField • ImagElectricFieldX • ImagElectricFieldY • ImagElectricFieldZ • CplxMagneticField • RealMagneticField • RealMagneticFieldX • RealMagneticFieldY • RealMagneticFieldZ • ImagMagneticField • ImagMagneticFieldX • ImagMagneticFieldY • ImagMagneticFieldZ
Region = "<string>" { "<string>", ... }	1	–	Regions for which data must be saved. If not specified, data for the whole simulation domain will be saved.

Table 32 Keywords of Save section (Continued)

Keyword	Unit	Default [range]	Description
WavelengthList = <float> { <float>, ... }	nm	–	Specifies a list of wavelengths to be used by the <code>Save</code> section for DFT-type simulations.
WavelengthPrecision = <integer>	1	0	Specifies the precision used for matching wavelengths given by <code>WavelengthList</code> and DFT wavelengths resulting from the specification in the <code>DFT</code> section. It also controls the number of digits after the decimal point displayed in the wavelength-identifying part of the save file name.
WavelengthRange = { <float>, <float> }	nm	–	Specifies the wavelength range to be used by the <code>Save</code> section for DFT-type simulations given by an interval (<min_wavelength>, <max_wavelength>).

Sensor

This section is optional and can occur multiple times.

Table 33 Keywords of Sensor section

Keyword	Unit	Default [range]	Description
BoxCenter = { <float>, <float>, <float> }	μm	–	Center of a nonaxis-aligned box.
BoxCorner1 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box that specifies the sensor domain. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
BoxCorner2 = { <float> min max, <float> min max, <float> min max }	μm	–	One of the two corners of a box that specifies the sensor domain. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.

Appendix A: Commands

Sensor

Table 33 Keywords of Sensor section (Continued)

Keyword	Unit	Default [range]	Description
BoxDimensions = { <float>, <float>, <float> }	μm	—	Dimensions of a nonaxis-aligned box.
CoordinateTransformation = <identifier>	1	Unified	Type of coordinate transformation. Exclusive options are: <ul style="list-style-type: none"> • Standard • Unified
Domains = <identifier>	1	Separate	Set operation on volumes defined by region and box corners for absorption and optical generation. Exclusive options are: <ul style="list-style-type: none"> • United • Intersecting • Separate
EndTick = <integer>	1	+inf	Specifies the end of the active tick step interval.
EndTime = <float>	s	+inf	Specifies the end of the active time interval.
Frequency = <float>	s^{-1}	—	Sensor frequency. The default value is calculated from the excitation wavelength.
Material = "<string>" { "<string>", ... }	1	—	Materials to which sensor applies. If neither <code>Material</code> nor <code>Region</code> is specified, data for the whole simulation domain is used.
Mode = <identifier> { <identifier>, ... }	1	{Integrate Average}	<code>Integrate</code> integrates a quantity over a surface or in a volume. <code>Average</code> divides the integrated value by the volume or surface area. Exclusive options are: <ul style="list-style-type: none"> • Integrate • Average
Name = "<string>"	1	—	Defines the name of the sensor, which also is used in the dataset name in the result file.

Appendix A: Commands

Sensor

Table 33 Keywords of Sensor section (Continued)

Keyword	Unit	Default [range]	Description
PlaneX = <float> <identifier>	μm	–	Sensor domain specified as a line (in two dimensions) or a plane (in three dimensions) perpendicular to the x-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
PlaneY = <float> <identifier>	μm	–	Sensor domain specified as a line (in two dimensions) or a plane (in three dimensions) perpendicular to the y-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
PlaneZ = <float> <identifier>	μm	–	Sensor domain specified as a plane in three dimensions perpendicular to the z-axis. For <identifier>, use either <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
Quantity = <identifier>	1	–	Specifies the integration quantity. Fluxes are integrated over surfaces; all others are integrated over volume. Exclusive options are: <ul style="list-style-type: none"> • <code>AbsorbedPhotonDensity</code> • <code>AbsorbedPowerDensity</code> • <code>OpticalGeneration</code> • <code>PowerFluxDensity</code> • <code>PhotonFluxDensity</code> • <code>IncoherentAbsorbedPhotonDensity</code> • <code>IncoherentOpticalGeneration</code>
Region = "<string>" { "<string>", ... }	1	–	Regions to which sensor applies. If neither <code>Region</code> nor <code>Material</code> is specified, data for the whole simulation domain is used.
RotationAngles = { <float>, <float>, <float> }	deg	{ 0 90 0 }	Rotation angles (theta, phi, psi) of a nonaxis-aligned box.

Appendix A: Commands

Sensor

Table 33 Keywords of Sensor section (Continued)

Keyword	Unit	Default [range]	Description
ScatteredFieldOnly = <identifier>	1	No	Specifies whether the sensor involves only scattered fields. Exclusive options are: <ul style="list-style-type: none"> • Yes • No
Sides = <identifier> { <identifier>, ... }	1	{ All }	If the quantity is a flux, specifies the sides for which integration is performed. Exclusive options are: <ul style="list-style-type: none"> • All • X • Xmin • Xmax • Y • Ymin • Ymax • Z • Zmin • Zmax
StartTick = <integer>	1	+inf	Specifies the beginning of the active tick step interval.
StartTime = <float>	s	+inf	Specifies the beginning of the active time interval.
TickStep = <integer>	1	100	Sensing is performed every <integer> tick step only.
WavelengthList = <float> { <float>, ... }	nm	—	Specifies a list of wavelengths to be used by the sensor for DFT-type simulations.
WavelengthPrecision = <integer>	1	0	Specifies the precision used for matching wavelengths given by WavelengthList and DFT wavelengths resulting from the specification in the DFT section.
WavelengthRange = { <float>, <float> }	nm	—	Specifies the wavelength range to be used by the sensor for DFT-type simulations given by an interval (<min_wavelength>, <max_wavelength>).

TruncatedPlaneWaveExcitation

This section is optional and can occur only once.

Table 34 Keywords of TruncatedPlaneWaveExcitation section

Keyword	Unit	Default [range]	Description
Amplitude = <float>	Vm^{-1}	1	Signal amplitude.
BoxCorner1 = { <float> min max, <float> min max, <float> min max }	μm	—	One of the two corners of a box that specifies the excitation plane. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
BoxCorner2 = { <float> min max, <float> min max, <float> min max }	μm	—	One of the two corners of a box that specifies the excitation plane. Use <code>min</code> or <code>max</code> for the minimum or maximum device extents along the corresponding coordinate axis.
Delay = <float>	1	0	Number of signal periods before signal starts.
Direction = <identifier>	1	—	Illumination direction with respect to the coordinate system of the input tensor grid. This keyword automatically sets the excitation angles <code>Theta</code> , <code>Phi</code> , and <code>Psi</code> as well as the excitation plane <code>Plane [X,Y,Z]</code> . Exclusive options are: <ul style="list-style-type: none"> • <code>fromTop</code> • <code>fromBottom</code>
EllipticalPolarization Magnitude = <float>	1	0	Polarization ratio magnitude.
EllipticalPolarization Phase = <float>	deg	0	Polarization ratio phase.
Frequency = <float>	s^{-1}	—	Signal frequency.
Intensity = <float>	Wcm^{-2}	1.32721e-07	Signal intensity.

Table 34 Keywords of *TruncatedPlaneWaveExcitation* section (Continued)

Keyword	Unit	Default [range]	Description
NRise = <float>	1	2	Number of signal periods before full amplitude is reached.
NumberOfPlaneWaves = <integer>	1	1	Number of excitation plane waves.
NumericalAperture = <float>	1	0	Numerical aperture.
PartialCoherenceFactor = <float>	1	1	Partial coherence factor.
Phi = <float>	deg	0	Azimuth angle of incidence.
PlaneX = <float> <identifier>	μm	—	Excitation line (in two dimensions) or plane (in three dimensions) perpendicular to the x-axis. For <identifier>, use either min or max for the minimum or maximum device extents along the corresponding coordinate axis.
PlaneY = <float> <identifier>	μm	—	Excitation line (in two dimensions) or plane (in three dimensions) perpendicular to the y-axis. For <identifier>, use either min or max for the minimum or maximum device extents along the corresponding coordinate axis.
PlaneZ = <float> <identifier>	μm	—	Excitation plane in three dimensions perpendicular to the z-axis. For <identifier>, use either min or max for the minimum or maximum device extents along the corresponding coordinate axis.
Polarization = <float> <identifier>	1	0.5	Polarization, either TE or TM, or a value between 0 and 1.
PolarizationAngle = <float>	deg	45	Polarization angle.
Psi = <float>	deg	45	Polarization angle.

Table 34 *Keywords of TruncatedPlaneWaveExcitation section (Continued)*

Keyword	Unit	Default [range]	Description
Region = "<string>"	1	–	Box that specifies the excitation plane.
Sigma = <float>	1	1	Width of transient Gaussian pulse in terms of number of signal periods.
Signal = <identifier>	1	Harmonic	Sets the transient form of the signal. Exclusive options are: <ul style="list-style-type: none"> • Gauss • Harmonic
Theta = <float>	deg	180	Polar angle of incidence.
Wavelength = <float>	nm	–	Signal wavelength.

B

Troubleshooting Simulations

This appendix provides troubleshooting help for Sentaurus Device Electromagnetic Wave Solver (EMW) simulations.

During simulations, EMW displays the most important issues in standard output. In Sentaurus Workbench, this is the *.out file. In addition, EMW writes a more detailed report to the log file set in the `Globals` section with `LogFile`, by default `<commandfilename>.log`.

The next sections discuss problems that could arise during simulations.

Unexpected Results

This section discusses unexpected results.

Did the simulation converge?

If the simulation converged successfully, you should see the following line in the .out file:

```
Detector: Termination criterion met after 809 steps.
```

Is the excitation correct?

You can double-check the excitation angle in the .out file:

```
Excitation angles in 3d specification: Theta = 90 deg, Phi = 90 deg,  
Psi = 0.0000e+00 deg.  
Excitation angles in 2d specification: Theta = 0.0000e+00 deg,  
Psi = 0.0000e+00 deg.
```

Are the material parameters correct?

In the log file, you can check all values for the refractive index and extinction coefficient:

```
Setting parameters for region "substrate".  
-----  
Complex refractive index: n = 3.939, k = 0.019767
```

Simulations Run Nonstop

This section discusses situations when simulations do not stop.

Did the simulation converge?

If the simulation converged successfully, you should see the following line in the .out file:

```
Detector: Termination criterion met after 809 steps.
```

Is the value of the time step reasonable?

In the .out file, you can find the chosen time step:

```
Time step set to 3.2810e-17s.
```

It is typically in the range $1\text{e-}18\text{ s} - 1\text{e-}17\text{ s}$. If it is much shorter, you probably have a very short edge of the mesh. See the *Sentaurus™ Mesh User Guide*.

Did the maximum deviation first decrease but then increase again?

If this occurs, then you can do the following:

1. From the log file, extract the time steps when the deviation starts to increase.
2. Perform some plotting around this time step to see where the problem is located.
3. Very often, the wave front hits a certain structure or boundary. If it is an absorbing boundary, try to improve the absorption by following the instructions in [Specifying Parameters of CPML Boundary Condition on page 28](#).

Simulator Terminates

If EMW terminates:

1. Check the `.out` file for error messages.
2. Remove freshly added features, or try to reduce your command file as much as possible, for example, remove all `Plot`, `Extractor`, `Detector`, `Sensor`, and `Save` sections.

Glossary

ADD

Automatic domain decomposition.

ADE

Auxiliary differential equation.

BC

Boundary condition.

CFL

Courant–Friedrichs–Lewy.

CPML

Convolutional perfectly matching layer.

CRI

Complex refractive index.

CRIMI

Complex refractive index model interface.

DFT

Discrete Fourier transform.

DP

Distributed processing.

EMW MPI

The distributed processing mode of EMW.

FDTD

Finite-difference time-domain.

GPU

Graphics processing unit.

LSF

Load Sharing Facility.

MPD

Management process or daemon.

MPI

Message passing interface.

NaN

Not a number.

Glossary

PEC

Perfect electric conductor.

PMC

Perfect magnetic conductor.

SMP

Shared-memory parallelization.

TFSF

Total-field scattered-field.