OpenParEM2D User's Manual

Version 2.1

April 2025

Brian Young

Contents

1 Introduction

OpenParEM2D is a full-wave electromagnetic simulator that solves the for the frequency-dependent complex propagation constant, characteristic impedance, and vector electric ($\overline{E}$) and magnetic ($\overline{H}$) fields of general 2D structures. It is a free and open-source project released under the GPL3 license.

OpenParEM2D solves the full set of Maxwell's Equations (hence, full-wave) in the frequency domain on the x-y plane with the assumption that the only z-dependence of the fields are $e^{-\gamma z}$. The solutions are mode of transmission lines and waveguides. The fundamental mode and optionally higher-order modes are solved.

OpenParEM2D is a command-line tool that performs just the electromagnetic calculations. To complete a project, other tools must be used to create a geometry, mesh the geometry, and plot results. The complete set of tools and operations is called a flow, and any number of flows are possible. The particular flow assembled for the development of OpenParEM2D is documented here.

This document covers how to use OpenParEM2D. Details on the theory, methodology, and accuracy of OpenParEM2D are covered in a separate document "OpenParEM2D_Theory_Methodology_Accuracy.pdf".

2 Features

OpenParEM2D features are listed in Table 1, while anticipated future upgrades are listed in Table 2.

Table 1: OpenParEM2D List of Features

- Full-wave frequency-domain solution of Maxwell's Equations
- Arbitrary 2D geometries
- Arbitrary order finite elements
- Parallel processing using the Message Passing Interface (MPI)
- Adaptive mesh refinement
- PEC, PMC, and impedance (perturbational) boundary conditions
- Calculations
  - complex propagation constant
  - characteristic impedance
  - dielectric loss
  - conductor loss
  - surface roughness loss
  - field distributions
- Dominant and higher-order modes
- Isotropic materials

Table 2: OpenParEM2D List of Anticipated Future Upgrades

- Eliminate null-space solutions
- Finite element order ramping
- Non-perturbational impedance boundary
- Anisotropic materials
- One or more graphical user interfaces (GUI)
- Microsoft Windows® port

3 Characteristic Impedance

From Maxwell's equations, the solution setup of a 2D cross-section of a transmission line or waveguide is an eigenvalue problem that on solving provides the complex propagation constant and electric field $\overline{E}$. The magnetic field $\overline{H}$ can be solved as a post-processing operation. Characteristic impedance is not a fundamental quantity when solving Maxwell's equations. Instead, it is calculated as a post-processing step on $\overline{E}$ and d $\overline{H}$ using any definition that meets the needs of the engineering design problem. The user must set up the calculation for the characteristic impedance through the definition of integration paths for voltages and currents and the selection of the definition of the characteristic impedance to use.

The specific calculation chosen for the characteristic impedance should be the one that is most useful in an engineering sense. The characteristic impedance is used to design for impedance matching to minimize reflections and for filter design, where reflections are controlled to produce a specific frequency response. So the best calculation to use is the one that produces the closest match between simulation and measurement. The best definition to use for one class of problems is not necessarily the best definition for another class of problems.

3.1 Definitions

Fundamentally, all solutions to the 2D problem are the modes of the structure, where each *mode* represents a unique solution to the eigenvalue problem with a unique field structure and complex propagation constant. For many users, only one mode is needed, which is referred to as the *dominant mode*. The electromagnetic modes are analogous to a string fixed on each end that is plucked, where there is a fundamental vibration plus harmonics. The fundamental tone is the dominant mode, and the harmonics represent the higher-order modes.

Each mode has its own unique field structure, so each also has its own unique characteristic impedance requiring a unique setup. In OpenParEM2D setups, configurations that apply directly to one given mode are called *modal* setups. In practice, this means that voltage and/or current integration paths specific to each mode must be defined.

For multiconductor transmission lines as seen in printed circuit boards and semiconductor packages, the modal setups are not known except for the special case of differential pairs (i.e. a multiconductor transmission line with just two symmetric signal conductors plus a ground plane). For multiconductor transmission lines, OpenParEM2D supports the definition of non-modal integration paths for the voltages and currents then applies an algorithm to compute the modal voltages and currents from the non-modal integration paths. These non-modal integration paths are called *line* setups.

Of the infinite number of possible definitions for the calculation of characteristic impedance, OpenParEM2D supports the three primary definitions, and it is at the user's discretion as to which definition to use for any given application. The three implemented definitions for the characteristic impedance, $Z_\circ$, are the power-voltage (PV) definition given by

$$Z_\circ = \frac{1}{2}\frac{VV^*}{P_z^*}, \tag{1}$$

the power-current (PI) definition given by

$$Z_\circ = 2\frac{P_z}{II^*}, \tag{2}$$

and finally, the voltage-current (VI) definition as

$$Z_\circ = \frac{V}{I}, \tag{3}$$

where V and I are the voltage and current, respectively, and P_z is the power propagating in the z-direction. The P_z term is always calculated for all modes, so the user does not need to set up anything for it. However, the voltage and current need further consideration.

3.2 Single-Mode Setups

In practice, most applications only use the dominant mode, for which the setups for the voltage and current are well known. For single-mode setups, integration paths for the voltage, the current, or both must be provided. If just the voltage path is provided, then only the PV definition can be calculated. Similarly, if just the current path is provided, then only the PI definition is available. If both are provided, then both definitions are calculated, and finally, if neither is provided, then no characteristic impedance is calculated. In the single-mode case, modal and line integration path setups are identical, use the same calculation, and produce the same result for the characteristic impedance.

3.3 Multi-Mode Setups

When more than one mode is requested, voltage and current integration paths must be defined by the user for the dominant and higher-order modes, and the definitions vary per mode. If definitions are not provided for a mode, then the characteristic impedance for that mode is not calculated.

The voltage and current paths must be provided by the user customized for each mode using modal setups. For the special case of multiconductor transmission lines, the user has the option of providing line setups instead of modal setups. With a line setup, non-modal voltages and currents are extracted from the 2D fields and combined to calculate the modal voltages and currents. This relieves the user of the need to define the modal integration paths, which are only known for differential pairs. Either setup ultimately calculates the modal voltages and currents and currents leading to modal characteristic impedances.

Consider the case of the differential pair shown in Fig. 1, which is marked with two current paths, `_P1` and `_P2`, and two voltage paths, `_P3` and `_P4`. For *modal* setups, one current path and/or one voltage path must be defined per mode. For the even mode (common mode), the currents on the conductors are equal, and the current is calculated over the total path of `_P1+_P2`. The voltages on the two conductors are equal, so the voltage is calculated on `_P3` or `_P4` or as 1/2 of the voltage calculated on the path `_P3+_P4`. For the odd mode (differential mode), the currents are equal and opposite, and the current is calculated as the current on `_P1` or `_P2` or as 1/2 the current calculated over the path `_P1−_P2`. The voltage is calculated as the total voltage the path `_P3−_P4`. For asymmetric differential pairs or multiconductor transmission lines with more than 2 conductors, the splits of the currents and voltages on the integration paths are not simple, so they are challenging to set up.

For line setups, combinations of the paths in Fig. 1 are not attempted. Instead, the voltages and currents are calculated for the individual paths as drawn. Since these are defined on a per-line basis instead of a per-mode basis, this setup is called the *line* setup. By themselves, the line voltages and currents are meaningless, but once combined, they yield the modal voltages and currents. For the differential pair in Fig. 1, two sets of voltages and currents are calculated using the line integration paths: one for the even mode and one for the odd mode. For the even mode, the computed line currents are I_1^e and I_2^e, and the line voltages are V_3^e and V_4^e. Similarly for the odd mode, the computed line values are I_1^o, I_2^o, V_3^o, and V_4^o. The even mode current is then $I^e = I_1^e + I_2^e$, and the even mode voltage is $V^e = 1/2(V_3^e + V_4^e)$. For the odd mode, $I^o = 1/2(I_1^o - I_2^o)$ and $V^o = V_3^o - V_4^o$.

For multiconductor transmission lines with line setups, OpenParEM2D applies a general algorithm to combine line currents and voltages to obtain modal currents and voltages. In the symmetric case of a differential pair, the algorithm produces the result just described. OpenParEM2D cannot know whether a drawing is a multiconductor transmission line, so if a line setup is applied by the user, then the algorithm is

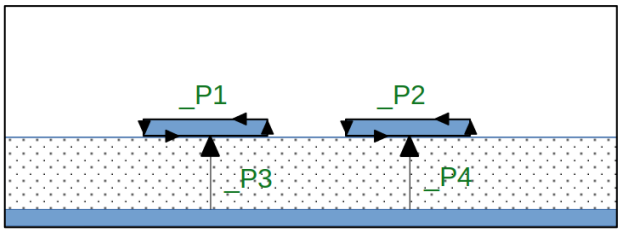


Figure 1: A differential pair marked with voltage and current integration paths.

applied whether or not it is appropriate. In general, modal setups should be used except for multiconductor transmission lines.

In summary, with a modal setup, the user provides integration paths for voltages and currents customized per mode to obtain the correct modal voltages and currents. For a multiconductor transmission line, the user has the option of providing a line setup with voltage and current paths defined per line, then OpenParEM2D applies an algorithm to combine the line voltages and currents to obtain modal voltages and currents. When properly set up, the two approached produce the same characteristic impedance for each mode.

4 Files

OpenParEM2D is a command-line tool that takes one and only one input, and that is a text project control file. To run serially on a single core or processor, the command is

```
$ OpenParEM2D project_name.proj
```

and in parallel it is

```
$ mpirun -q --oversubscribe -np N OpenParEM2D project_name.proj
```

where `project_name.proj` is the project control file. See Sec. 6.4 for details on the switch settings `-q`, `-oversubscribe`, and `-np`. To view a very short help message and the version number, execute

```
$ OpenParEM2D -h
```

All inputs are captured in the project control file including names of files that must be included. So setting up a project involves creating/editing a project control file and ensuring that the additionally required files are specified and available.

The required include files in a project control file define materials, mesh, and port definitions. Each of the required files are covered in detail in the following sections.

4.1 Project Control File

All aspects of the execution of OpenParEM2D are controlled by the project control file. There are no other inputs or command line options. The file consists of a list of keyword/value pairs that are documented in Appendix A. A simple but working control file is shown below.

```
#OpenParEM2Dproject 1.0
project.save.fields               false
mesh.file                         coax.msh
mesh.order                        2
mode.definition.file              coax_modes.txt
materials.global.path             ../
materials.global.name             global_materials.txt
refinement.frequency              high
refinement.variable               |Zo|
refinement.iteration.min          3
refinement.iteration.max          50
refinement.required.passes        1
refinement.tolerance              1e-5
frequency.plan.log                1e7,10e9,1
solution.modes                    1
solution.impedance.definition     PV
solution.impedance.calculation    modal
```

In this example setup, the mesh file "coax.msh" is generated using gmsh [1][2], the coax_modes.txt file is generated using the script "OpenParEM2D_save.py" in FreeCAD [3], and the materials file "global_materials.txt" is a hand-generated library file used across many projects. Keyword/value pairs can be added to the control

file to do things like outputting files to enable plotting with ParaView [4] [project.save.fields true] or calculate field values at specific (x,y) points [field.point 0,0.0008].

The names of the files generated by OpenParEM2D are based on the name of the project control file with the extension removed. So a control file called "my_project.proj" results in new files and directories being created called "my_project_results.csv", "ParaView_my_project", "temp_my_project", etc. The files are uniquely identified by the project name, so more than one project can exist in a single directory without file name conflicts.

The text-based control file has several advantages, with a few techniques listed here.

- Project control files tend to be similar across projects, so common practice is to copy a control file from an old project to a new one and then edit as necessary.
- A project can have multiple control files to explore simulation strategies. A simple comparison of project control files [e.g. Linux diff or sdiff] shows the differences in setup.
- Directory trees can be easily searched to find projects using or not using certain simulation features using the helper tool "proj_search". As completed projects accumulate, this feature becomes more useful.
- Comments can be used to add notes and keep track of changes or variations. In a GUI, the only option for the user is to check/uncheck a box or to change a number. With a text-based setup file, the change can not only be made, but the *reason* for the change can be given or the *history* of the setting can be maintained. This is a surprisingly capable feature that should be liberally used.

The frequencies to solve and the frequencies and order for adaptive mesh refinement are defined in a *frequency plan*. OpenParEM2D sets up frequency plans identically to OpenParEM3D, and the definition and use of frequency plans are covered in "OpenParEM3D_User_Manual.pdf".

4.2 Materials Files

Materials use the same materials files as described in "OpenParEM3D_User_Manual.pdf".

4.3 Mesh File

The supported mesh file formats are gmsh v.2.2 and the native format of MFEM [5][6]. When setting up a project, the general procedure is to use a CAD program that can output geometry data readable by gmsh, then use gmsh to mesh the structure and output a gmsh v2.2 mesh file. The mesh file is then specified in the project control file using the `mesh.file` keyword. To get the v.2.2 mesh output from gmsh, the format must be specified on the command line when starting gmsh as follows

```
$ gmsh -format msh22 &
```

One free and open-source CAD program that is supported by gmsh is FreeCAD. FreeCAD outputs a *.brep file that can be imported by gmsh.

4.4 Mode Definition File

Modes and boundaries are specified in a text file following the specification in Appendix B. The project control file specifies the mode definition file using the `mode.definition.file` keyword. Due to complexity, the mode definition file is very difficult to write by hand, so scripted generation is generally required. When using FreeCAD, the Python script "OpenParEM2D_save.py" can be used to generate the mode definition file.

5 Example Projects

An automated regression suite is in place as part of the release methodology. The projects in the suite are listed in Appendix C. The suite also represents a large number of worked examples that are ready to run with known answers. For example, for the regression project `WR90_rectangular_waveguide/WR90/WR90_order_4_refinement/WR90.proj`, the project can be run at the command line by executing

```
$ process.sh WR90.proj 5
```

where depending on the computer, a number smaller or larger than 5 could be appropriate for the number of cores to use. At successful completion, the file "WR90_test_results.log" can be viewed for summary results and "straight_test_test_results.csv" for a detailed pass/fail report. Results for the complex propagation constant, characteristic impedance, voltage, current, and power are available in "WR90_test_results.csv". Computed values for the fields are available in "WR90_test_fields.csv". To view the fields, set `project.save.fields true` before running the project, then run ParaView to setup and view fields.

The regression suite cases are set up for more accuracy than needed for engineering work to enable tight pass/fail criteria to be set. In many situations, looser settings are appropriate.

The accuracy demonstrations listed in Sec. D provide additional worked examples. These are set up for maximum accuracy and accept relatively slow run times.

6 Flow

The process of setting up and running a project from scratch involves several steps: draw the geometry via CAD or script, create a mesh, define ports and set boundary conditions [if needed], create a local materials file if the needed materials do not exist in the global library file, and finally run OpenParEM2D. Except for the OpenParEM2D step, any number of tools and techniques can be used to generate the needed files. One methodology is described here.

6.1 Geometry Definition Using FreeCAD

FreeCAD is an effective tool for building complex 2D geometries for simulations. It does have a significant learning curve, but tips are provided to help with that. Start FreeCAD with

```
$ freecad &
```

then start a new project with `File`→`New`. At first use, set some preferences starting with `View`→`Workbench`→`Draft` then `Edit`→`Preferences ...`. Click on the `General` icon then on the `General` tab and set the `Unit system:` to `Standard (mm, kg, s, degree)` and `Number of digits:` to 3. Next click on the `Draft` icon and then the `General settings` tab and set the `Internal precision level` to 10. Finally, click on the `Draft` icon and then the `Grid and snapping` and set the `Grid spacing` to 1 μm to enable drawing at μm scale as discussed next on the unit system in FreeCAD.

An important note about the unit system is that FreeCAD has internal scaling that does not seem to adhere to the length units of mm. Drawing in mm will generally lead to the mesh being output in m. This seems strange for its use here, but it must work in other scenarios. Or, perhaps a setting has been missed. Generally, drawings must be made in μm to get mm scaling in the final mesh, or the drawing can be done in mm then scaled by 0.001 in the x-, y-, and z- directions.

To continue with drawing, FreeCAD should be in the Draft Workbench by selecting `View`→`Workbench`→`Draft`. The next step is critical to avoid a very messy drawing experience. On the toolbar, click the `Auto` button then click `Top (XY)` to set the drawing plane. The drawing plane is where 2D primitives such as lines and rectangles are drawn.

To build a 2D model, primitives such as lines and rectangles are drawn on the drawing plane. It is important to use the snap functionality to ensure that objects are aligned. Once drawn, editing of an object is often easier by editing the text parameters rather that using mouse operations.

In addition to a typical copy operation, FreeCAD also has a clone operation. The clone operation is very powerful because edits to the parent object cascades to the child cloned objects. Use of clones does take some planning to ensure that changes to the parent are reflected in the clones in ways that make sense for the 2D model.

When the 2D model is complete, all of the components must be merged using the "boolean fragments" operation in the Part Workbench with `Part`→`Split`→`Boolean fragments`. The resulting object can be exported using `File`→`Export ...` to save with the BREP file format, which can be imported by gmsh for material assignment and meshing.

If the 2D model is built in mm, then the boolean fragment object can be cloned, and the clone can be scaled in 3 dimensions by 0.001. The scaled clone can then be exported as a BREP file. A scaled clone loses the connection to the parent object, so a change to the parent requires the scaled clone to be deleted and re-created with a new clone.

FreeCAD has vastly more capability than that described here, and there are very likely alternative or better ways to get things done than those described in these tips. Given its expansive capabilities and Python scripting support, it is highly likely that scripting could greatly simplify the process of building a 2D model for OpenParEM2D.

6.2 Mode and Boundary Annotation Using FreeCAD

The required mode definition file can be built by hand, but that is not generally practical. The FreeCAD Python script "OpenParEM2D_save.py" can be used to generate the needed mode definition file once the 2D model has been annotated with the needed mode and boundary information. The annotations are added as physical objects, such as lines, rectangles, and polylines, along with text objects providing instructions to "OpenParEM2D_save.py".

6.2.1 Paths (_P)

Paths are physical drawing elements used as voltage and current integration paths. Lines, rectangles, and polylines can be added to the drawing then marked as paths by editing the `Label` property in the property box to

```
_Ppathname
```

where `pathname` can be any text with alphanumeric characters. The paths are referenced by other annotations only by `pathname`, so the _P part is dropped.

6.2.2 Integration Paths (_M, and _L)

Integration paths for voltages and currents are set up using previously defined paths plus additional information indicating voltage or current plus [optionally] scale. An integration path is added to the drawing by placing a text object with any text, which is ignored. A simple period works well. The integration path information is entered by editing the `Label` property of the added text object located in the property box to

```
_Mmodenumber(voltage|current[,scale]){[+|-]pathname1,+pathname2,-pathname3,...}
```

or

```
_Lmodenumber(voltage|current[,scale]){[+|-]pathname1,+pathname2,-pathname3,...},
```

where `modenumber` is an integer, `scale` is a scale factor for the computed results [default=1], and the pathnames define the path. The pathnames string together, where + and − signs allow reversing direction, if needed. The two forms (*_M* or *_L*) are identical except for the prefix, which determines whether modal or line calculations are used to obtain the voltages and currents. For each modenumber, two entries are allowed: one for voltage and one for current.

OpenParEM2D solves for the dominant and higher-order modes, and the modes are sorted so that they are ordered from highest to lowest effective dielectric constant. The voltage and/or current integration paths defined for modenumber=1 are applied to the mode with the highest effective dielectric constant. The mode with the next highest effective dielectric constant is used with the integration paths defined with modenumber=2, and so on. It is required that modenumber start with 1 and increase sequentially.

6.2.3 Boundaries (_B)

Boundaries conditions are defined by placing a text object with any text, which is ignored, with the `Label` property in the property box edited to

```
_Bboundaryname(SI|PEC|PMC[,material]){[+|-]path,-path,+path,...}
```

where `boundaryname` can be any alphanumeric text, the boundary type is selected as one of `SI` for surface impedance, `PEC` for perfect electric conductor, and `PMC` for perfect magnetic conductor. Selecting `SI` requires the conductor material to be specified in the second argument [e.g. copper].

A PEC boundary forces the electric field tangential to the boundary to be exactly zero. This is the default boundary condition for any edge facing outside to an area that does not have a defined dielectric. The PEC boundary is useful for enforcing symmetry to reduce the size of the computational space for problems with anti-symmetric electric fields and symmetric magnetic fields.

A PMC boundary forces the magnetic field tangential to the boundary to be exactly zero. The PMC boundary is useful for enforcing symmetry to reduce the size of the computational space for problems with symmetric electric fields and anti-symmetric magnetic fields.

An SI boundary assumes that the conductor is thick compared to the skin depth at the simulation frequency. No penetration through the boundary is modeled.

6.3 Meshing Using gmsh

For meshing, gmsh is an effective tool that has a relatively straightforward interface. The library used by OpenParEM2D only supports the version 2.2 mesh format from gmsh, so it must be started calling for this format as

```
$ gmsh -format msh22 &
```

Open the BREP file exported from FreeCAD with `File→Open ...` then select a file with the `.brep` extension.

Before meshing, materials must be assigned to the surfaces in the imported model. In the cascading options, select ⊞`Geometry`→⊞`Physical groups`→⊞`Add`→`Surface`. A popup menu appears along with crossing dotted lines on the model indicating the surfaces. Add text to the popup for the material for a specific surface, then click a crossing line (or more, if all have the same material) for that surface, then enter `e` on the keyboard to complete the entry. A popup will appear to create a file with a `.geo` extension, and accept that. Continue until all surfaces have materials assigned, then enter `q` to quit. Note that sometimes entry by the mouse is disabled, indicated by a red box in the lower left-hand corner, so if the mouse becomes disabled, click that red box.

With the materials defined, in the cascading options select ⊞`Mesh`→`2D` to generate a mesh. If the mesh is acceptable, then in the cascading options select `Save` to save the mesh with an automatic `.msh` extension. Gmsh can be closed without needing to save any information because the `.geo` file is automatically saved. On re-opening a project, load the `.geo` to pick up the surface assignments. Meshing can then immediately proceed.

At the GUI level, meshing behavior can be modified by opening a control panel with `Tools→Options`. An options window will appear, then select `Mesh→General`. The primary controls are the 2D algorithm and the element size factor. At this point, it is up to experimentation to see what works best for the geometry at hand. After a mesh is generated, a change in settings is made by selecting in the cascading options ⊞`Geometry`→`Reload script` to clear the existing mesh, then make any changes to the meshing options and select ⊞`Mesh`→`2D` to generate a new mesh.

Generally, a sparse mesh is desirable for speed, but sparser meshes have lower quality, which impacts the ultimate speed and accuracy of the electromagnetic solution. The mesh quality can be viewed by selecting `Tools→Statistics→Update`. Several metrics are supplied, but the idea is to get these as close to 1 as possible, but generally, they will be far less than 1.

Once a mesh is complete, select `Files→Save Model Options` to save the settings to a file with an automatic `.opt` extension. The `.opt` is always applied so that new mesh generation can start where the last one left off. To start over from scratch, delete the `*.opt` file before starting gmsh.

Beyond the simple use described here, gmsh has a vast array of features and options plus scripting capabilities. With the simple tips above, meshing can get started while additional capabilities are explored over time.

6.4 Solution With OpenParEM2D

Once the mode definition file from FreeCAD and the mesh file from gmsh have been generated, the project setup file can be completed with links to these and to a materials file. With all of the files ready, a project file with the name `my_project.proj` can be executed by

```
$ mpirun -q --oversubscribe -np N OpenParEM2D project_name.proj
```

where `N` is the number of cores to use for the calculation. The switch `-q` is optional and eliminates extra messaging from the MPI system. The switch `--oversubscribe` is required when `N` is greater than half the number of available cores, but otherwise, it is optional. `N` should generally not exceed the number of available physical cores, and in most cases, best run times occur when `N` is less than the number of available cores. Experimentation is required to find the value of `N` that results in the lowest run times for a specific computer.

For very small problems, setting `N` too large can cause OpenParEM2D to hang at the function call within the MFEM library that sets up the finite element problem. The problem is that there is not enough data to spread among the number of requested processors. There is not a way for OpenParEM2D to check and report when `N` is too large. If OpenParEM2D hangs at the step `building finite element spaces ...`, then the remedy is to kill the job and re-start with a smaller `N`.

It is likely on initial attempts to run a new setup with OpenParEM2D that errors will be reported and the simulation stopped. The design of OpenParEM2D is to make no assumptions or corrections to input decks, so every error is reported even if it could be possible for OpenParEM2D to fix or adjust the setup. Correct the errors and try again.

OpenParEM2D uses a lock file to prevent data collisions from accidentally running the same project file more than once at a time. The project lock file has the form `.project_name.lock`. Should a running job exit unexpectedly, the lock file can be deleted as

```
$ rm .project_name.lock
```

so that the job can be restarted. Since the name starts with a period, it is a hidden file, so to see the lock file in a directory requires

```
$ ls -a
```

The goal is that OpenParEM2D always detects problems and exits gracefully with an error message with the lock file removed. Any other exit style is an issue that needs to be fixed with improved error checking, with one exception. For a large project that runs out of memory, OpenParEM2D will exit with cryptic messages from the MPI system without mentioning the memory issue and without removing the lock file.

When a job is running, progress is shown by extensive data sent to the terminal. This output should be redirected to a file if the simulation progress is to be logged.

6.5 Viewing Fields

To view the computed electromagnetic fields, OpenParEM2D outputs files for viewing with ParaView after every iteration and every frequency. To enable this output, set

```
project.save.fields true
```

in the project setup file. After loading the result in ParaView, a very useful predefined set of plots can be generated by running the macro "field_plot" from the `Macros` menu item.

Since the files are produced while OpenParEM2D is executing, ParaView can be used to view intermediate results. The one constraint is that ParaView will report errors if OpenParEM2D updates the files while ParaView is still reading them. If that happens, just try again.

The default behavior for ParaView is to enable the user to make several changes then click `Apply` to review the recomputed output. This is good behavior for very large datasets but inconvenient for small ones. A preference can be set to automatically implement changes immediately after every update. This setting can be found at `Edit`→`Settings ...`→`General`→`Auto Apply`.

After a ParaView file is opened, if the macro "field_plot" is not run, then nothing is shown. To show a result, scan down to the `Coloring` section where there is a dropdown box labeled `Solid Color`. Change

this setting to one of the field options. For vectors, the displayed field can be further refined by changing from plotting `Magnitude` to one of the field components.

Beyond the simple use described here and the tutorials, ParaView has a vast array of filters and customizations forming a very powerful visualization capability. With the simple tips above, visualization can get started while additional techniques are explored over time.

7 Results File

Computed results are saved to a csv file called "project_name_results.csv". A commented header line shows the content of each column. The only outputs from OpenParEM2D are this file, the ParaView output, and the data printed to the terminal during execution.

8 Tutorials

8.1 Rectangular Waveguide

WR90 rectangular is constructed, annotated, meshed, and solved. WR90 rectangular waveguide is 22.86 mm wide and 10.16 mm tall and has a recommended frequency range from 6.557 GHz to 13.114 GHz.

8.1.1 Drawing and Mode Annotation

- Create a directory for the project

```
$ mkdir rectangular_waveguide
$ cd rectangular_waveguide
```

- Start FreeCAD, open a new drawing, set preferences [if needed], set the drawing workspace to `Draft`, and set the drawing plane to `Top` as outlined in Sec. 6.1.
- Click `Drafting`→`Rectangle` or click on the rectangle icon on the toolbar and draw a rectangle of any size by clicking to start then clicking to end.
- Select the rectangle either on the drawing space or in the `Combo View` window.
- In the `Property` window, for property `Height` change the value to 10.16 um. Sec. 6.1 discusses why drawings are in μm to ultimately obtain a mesh in mm. For property `Width` change the value to 22.86 um.
- Again in the `Property` window, navigate through the properties `Placement`→`Position` then enter 0 for x, y, and z.
- Zoom to view the rectangle with `View`→`Standard Views`→`Fit All`.
- On the toolbar, click the ≫ to show the snap options, and click `Snap Midpoint`.
- On the toolbar, click the `Line` object then draw a line from the midpoint of the bottom of the long edge to the midpoint of the top of the long edge. Be sure that the white dot is showing to indicate that the line is snapping to the rectangle. This line object is used as the voltage integration path.
- Select the line object in the `Combo View` window, then click in the `Label` property and change the text to `_Pv`.
- On the toolbar, click the `Text` object then click anywhere on the drawing plane to add text. In the pop-up text box, add a period then click `Create text`. This text object is used to define the voltage path.
- Select the text object in the `Combo View` window, then click in the `Label` property and change the text to `_M1(voltage){v}`. This is a modal definition for the voltage.

- Save the drawing with the name `rectangular_waveguide` using `File`→`Save`.
- The completed drawing and annotation is shown in Fig. 2.
- Export the mode description file by selecting `Macro`→`Macros ...`→`OpenParEM2D_save.py`→`Execute`. Enter the name `rectangular_waveguide_modes.txt`, then select `Save`. Check the `Report view` window for errors.
- Select the rectangle, then `File`→`Export...`, make sure that the `BREP format` is selected, then save the file as `rectangular_waveguide.brep`.
- Save the drawing and exit FreeCAD.

8.1.2 Meshing

- Start gmsh as outlined in Sec. 6.3.
- Open the BREP file saved from the prior section by selecting `File`→`Open ...`→`rectangular_waveguide.brep`→`Ok`.
- Assign the material as air by clicking the options tree ⊞`Geometry`→⊞`Physical Groups`→⊞`Add`→`Surface`.
- In the pop-op window type `air` then select one of the crossing lines, which turns red. Press the keyboard `e` and a new pop-up appears. Click `Create new '.geo' file`. Finally, press the keyboard `q` to finish. [If the mouse does not select the dotted line, click the red box in the lower left to re-enable mouse input.]
- To mesh the geometry, in the options tree click ⊞`Mesh`→`2D`. A screenshot of the mesh is shown in Fig. 3.
- Save the mesh by selecting `File`→`Save Mesh`.
- Quit gmsh.

8.1.3 Solving

- Create the materials file `global_materials.txt` in any text editor and set the text contents to

```
#OpenParEMmaterials 1.0
Material
   name=air
   Temperature
      temperature=any
      Frequency
         frequency=any
         er=1.0006
         mur=1
         tand=0
         Rz=0
      EndFrequency
   EndTemperature
   Source
      Constantine A. Balanis, "Advanced Engineering Electromagnetics",
      John Wiley and Sons, 1989, p.79.
   EndSource
EndMaterial
```

- Create the project control file `rectangular_waveguide.proj` in any text editor and set the text contents to

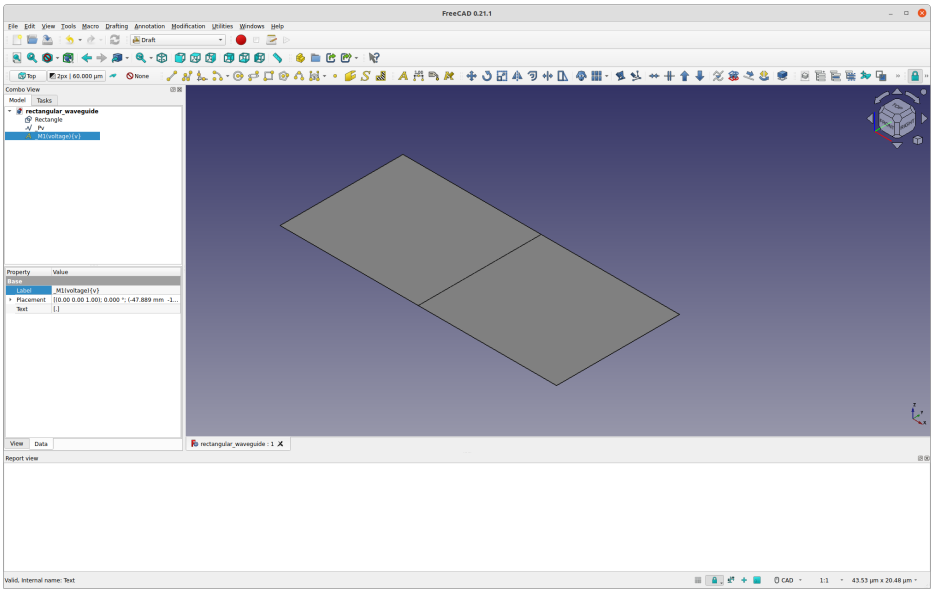


Figure 2: Drawing and annotation for a WR90 rectangular waveguide.

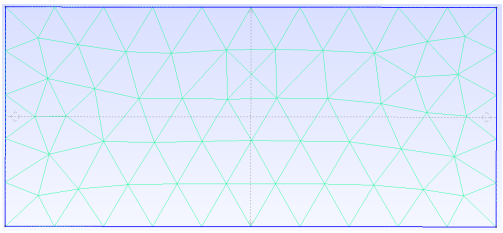


Figure 3: Screenshot of the meshed WR90 rectangular waveguide.

```
#OpenParEM2Dproject 1.0
project.save.fields            true
mesh.file                      rectangular_waveguide.msh
mesh.order                     3
mode.definition.file           rectangular_waveguide_modes.txt
materials.global.path          ./
materials.global.name          global_materials.txt
materials.local.path           ./
materials.local.name           //local_materials.txt
refinement.frequency           none
frequency.plan.point           10e9
solution.modes                 5
solution.modes.buffer          10
solution.impedance.definition  PV
solution.impedance.calculation modal
```

Note that iterative refinement is not needed for this problem since the mesh is fairly dense, 3^{rd}-order finite elements are used, and the fields vary slowly. See Appendix A for a complete list of available keyword/value pairs that can be included in the project control file.

- Run OpenParEM2D at the command line with

  ```
  $ OpenParEM2D rectangular_waveguide.proj
  ```

  to run serially with a single core or with

  ```
  $ mpirun -q --oversubscribe -np 4 OpenParEM2D rectangular_waveguide.proj
  ```

  to run in parallel with 4 cores. Substitute a larger or smaller core number as needed. The option `--oversubscribe` is not needed if the number of cores is less than or equal to half the number of available cores.

8.1.4 Results

The output includes the computed results

mode	frequency	er,eff	alpha,dB/m	beta/1000,rad/m	beta/ko	Zo(real)	Zo(imag)
1	1e+10	0.5706391	-1.0119426e-10	0.15832151	0.75540658	443.29902	-2.5441599e-11
2	1e+10	5.1589308e-27	1543.8729	1.5053553e-14	7.1825698e-14		
3	1e+10	1.0011881e-27	1974.2013	6.6315799e-15	3.1641556e-14		
4	1e+10	2.237198e-27	2307.0205	9.9131431e-15	4.7299027e-14		
5	1e+10	1.3660112e-27	2307.0213	-7.7461569e-15	-3.6959588e-14		

showing 1 propagating mode and 4 cutoff higher-order modes. Modes 4 and 5 are degenerate since they have the same complex propagation constant. An impedance is generated only for mode 1 since a voltage integration line is only generated for modenumber=1.

From "rectangular_waveguide_results.csv", beta/ko for mode 1 is 0.755406579379436 vs. the exact analytical result of 0.755406579840748, so the OpenParEM2D result with this setup is accurate to 9 decimal places. Alpha in dB/m for mode 2 is 1543.87289318585 vs. the exact analytical result of 1543.8726526439, so again a very close match. The comparison is similar for the remaining 3 modes. For the characteristic impedance for mode 1, the OpenParEM2D result is 443.299021764102 Ω vs. the exact analytical result of 443.299540513372 Ω.

Note that the results change slightly from run-to-run. This is due to the use of an iterative solution combined with MPI processing.

Fields can be viewed with ParaView as follows.

- Start ParaView at the command line with `$ paraview &`, then navigate to and open the fields result `rectangular_waveguide_frequency_1e+10_mode_1.pvd`

- Run `Macros`→`field_plot`. Wait a few moments.
- A wide variety of plots are available for viewing, both magnitude and vector.
- Scaling in ParaView does not work well for vector plots, so manual adjustment is always required.
- Click `E real vector` then enter 2e-6 for `Scale Factor`.
- The resulting field plot is shown in Fig. 4, where the electric field is vertically directed.

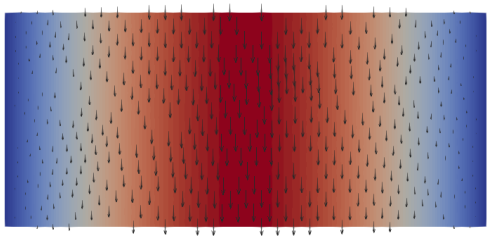


Figure 4: Re(Et) for the WR90 rectangular waveguide.

8.2 Coaxial Waveguide

RG401 type 0.250 semi-rigid coaxial cable is constructed, annotated, meshed, and solved. The RG401 coax is assumed to have an inner conductor diameter of 1.63 mm, an outer conductor inner diameter of 5.46 mm, a PTFE dielectric with $\epsilon_r = 2.1$ and a $\tan\delta = 0.0004$. The metals are assumed to be copper with $\sigma = 5.813 \times 10^7$ S/m.

8.2.1 Drawing and Mode Annotation

- Create a directory for the project

```
$ mkdir coaxial_waveguide
$ cd coaxial_waveguide
```

- Start FreeCAD, open a new drawing, set preferences [if needed], set the drawing workspace to `Draft`, and set the drawing plane to `Top` as outlined in Sec. 6.1.
- Click `Drafting`→`Polygon` or click on the polygon icon on the toolbar and draw a polygon of any size by clicking to start then clicking to end. This is the first polygon.
- Select the polygon either on the drawing space or in the `Combo View` window.
- In the `Property` window, for property `Radius` change the value to 0.815 um. Sec. 6.1 discusses why drawings are in μm to ultimately obtain a mesh in mm. Change the property `Faces Number` to 36. This is the center conductor.
- Again in the `Property` window, navigate through the properties `Placement`→`Position` then enter 0 for x, y, and z.
- Zoom to view the polygon with `View`→`Standard Views`→`Fit All`.
- On the toolbar, click the ≫ to show the snap options, and click `Snap Center`.
- Draw a second polygon snapped to the center of the first and configure it with the same settings so that it is a copy. Alternatively, copy the the first polygon to a second with the `Copy` and `Paste` commands under the `Edit` menu command. Do not use a cloned object because the Python script `OpenParEM2D_save.py` does not support clones.

- Change the Label property of the second polygon to _Pinner. Change the property Make Face to false [optional, but looks better]. This path is used for both the current integration path and the surface impedance boundary using copper for the center conductor.
- Draw a third polygon snapped to the center of the first with the same settings except with a radius of 2.73 um. This is the dielectric once the center conductor is removed.
- Copy and paste the third polygon to a fourth and change its Label property to _Pouter. Set its property Make Face to false, if desired. This path is used for the surface impedance boundary using copper for the outer conductor.
- Change the drawing workspace to Part. Select the third polygon then the first, then select Part→Boolean→Cut to cut the center conductor from the dielectric. Agree to the operation in the warning pop-up window. This object is the coax. Change the Label property to coax.
- Change the drawing workspace back to Draft.
- Draw a line from the outer conductor of the coaxial cable to the inner conductor. Change its Label to _Pv. This is the integration path for the voltage.
- Add a text object with the Label _M1(voltage){v}. This is the modal definition for the voltage.
- Add a text object with the Label _M1(current){inner}. This is the modal definition for the current.
- Add a text object with the Label _Bi(SI,copper){inner}. This is the surface impedance definition for the inner conductor.
- Add a text object with the Label _Bo(SI,copper){outer}. This is the surface impedance definition for the outer conductor.
- Save the drawing with the name coaxial_waveguide using File→Save.
- The completed drawing and annotation is shown in Fig. 5, where the _Pinner is highlighted in green and _Pouter is highlighted in yellow.
- Export the mode description file by selecting Macro→Macros ...→OpenParEM2D_save.py→Execute. Enter the name coaxial_waveguide_modes.txt, then select Save. Check the Report view window for errors.
- Select the coax object, then File→Export..., make sure that the BREP format is selected, then save the file as coaxial_waveguide.brep.
- Save the drawing and exit FreeCAD.

8.2.2 Meshing

- Start gmsh as outlined in Sec. 6.3.
- Open the BREP file saved from the prior section by selecting File→Open ...→coaxial_waveguide.brep→Ok.
- Assign the material as PTFE by clicking the options tree ⊞Geometry→⊞Physical Groups→⊞Add→Surface.
- In the pop-op window type PTFE then select one of the crossing lines, which turns red. Press the keyboard e and a new pop-up appears. Click Create new '.geo' file. Finally, press the keyboard q to finish. [If the mouse does not select the dotted line, click the red box in the lower left to re-enable mouse input.]
- To mesh the geometry, in the options tree click ⊞Mesh→2D. A screenshot of the mesh is shown in Fig. 6.

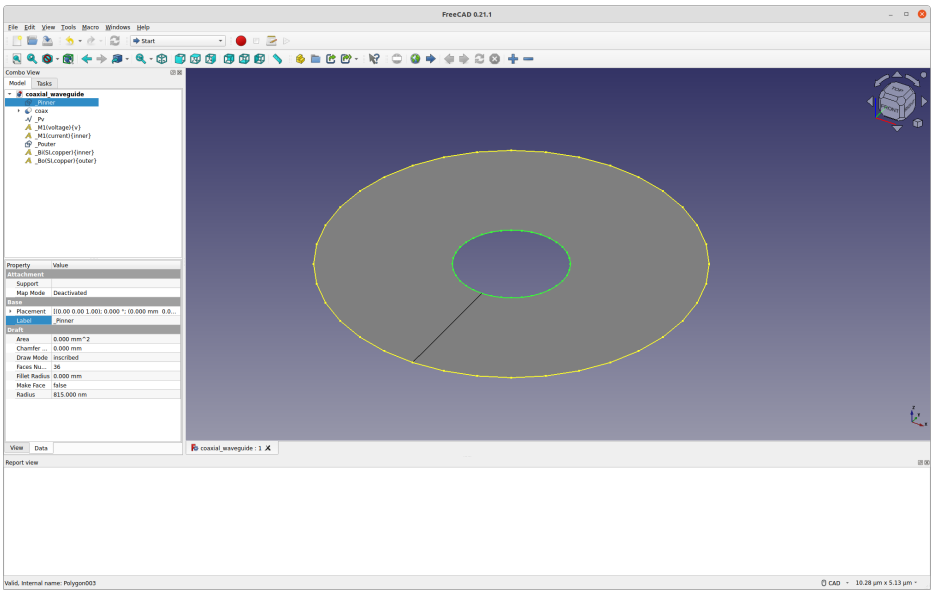


Figure 5: Drawing and annotation for an RG401 coaxial waveguide.

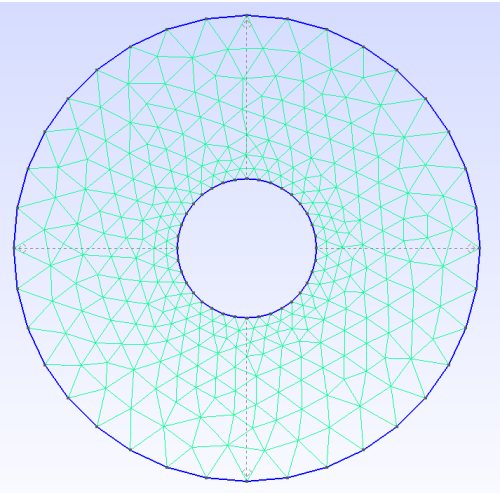


Figure 6: Screenshot of the meshed RG401 coaxial waveguide.

- Save the mesh by selecting `File`→`Save Mesh`.
- Quit gmsh.

8.2.3 Solving

- Create the materials file `local_materials.txt` in any text editor and set the text contents to

```
#OpenParEMmaterials 1.0
Material
   name=PTFE
   Temperature
      temperature=any
      Frequency
         frequency=any
         er=2.1
         mur=1
         tand=0.0004
         Rz=0
      EndFrequency
   EndTemperature
   Source
      generic numbers
   EndSource
EndMaterial
Material
   name=copper
   Temperature
      temperature=20
      Frequency
         frequency=any
         er=1
         mur=1
         conductivity=5.813e7
         Rz=0
      EndFrequency
   EndTemperature
   Source
      David M. Pozar, "Microwave Engineering," Addison-Wesley Publishing Company,
         1990, p.714.
   EndSource
EndMaterial
```

- Create the project control file `coaxial_waveguide.proj` in any text editor and set the text contents to

```
#OpenParEM2Dproject 1.0
project.save.fields            true
mesh.file                      coaxial_waveguide.msh
mesh.order                     3
mode.definition.file           coaxial_waveguide_modes.txt
materials.global.path          ./
materials.global.name          //global_materials.txt
materials.local.path           ./
materials.local.name           local_materials.txt
refinement.frequency           none
frequency.plan.point           10e9
solution.modes                 1
solution.modes.buffer          0
solution.impedance.definition  PV
```

```
solution.impedance.calculation modal
solution.temperature           20
debug.show.impedance.details   true
```

Note that iterative refinement is not needed for this problem since the mesh is fairly dense, 3rd-order finite elements are used, and the fields vary slowly. See Appendix A for a complete list of available keyword/value pairs that can be included in the project control file.

- Run OpenParEM2D at the command line with

```
$ OpenParEM2D coaxial_waveguide.proj
```

to run serially with a single core or with

```
$ mpirun -q --oversubscribe -np 4 OpenParEM2D coaxial_waveguide.proj
```

to run in parallel with 4 cores. Substitute a larger or smaller core number as needed. The option `--oversubscribe` is not needed if the number of cores is less than or equal to half the number of available cores.

8.2.4 Results

The output includes the characteristic impedance computed with all three definitions, shown as

```
Mode
     1  voltage (V): (5.13256,9.54287e-12)
        current (I): (0.102229,-2.04457e-05)
        Pz (Pz,avg): (0.263587,5.27173e-05)
Mode
     1  Impedance (VI): (50.2067,0.0100413)
                  (PV): (49.9706,0.00999411)
                  (PI): (50.4439,0.0100888)
```

The PV definition is very close to the designed value of 50 Ω. The output also includes the computed results

mode	frequency	er,eff	alpha,dB/m	beta/1000,rad/m	beta/ko	Zo(real)	Zo(imag)
1	1e+10	2.1000001	1.0990629	0.3037168	1.4491377	49.97056	0.0099941114

showing the effective dielectric constant as 2.1000001, which is very close to the lossless theoretical value of 2.1 for this homogeneous material.

The loss is 1.099 dB/m. Fairview Microwave on its datasheet for RG401 Type 0.250 part number FM-SR250CU-STR specifies the loss as 1.083 dB/m. These two loss figures are in very good agreement given that the exact dimensions nor material properties of the Fairview Microwave product are known. Simply using generic numbers gets very close.

Note that the results change slightly from run-to-run. This is due to the use of an iterative solution combined with MPI processing.

Fields can be viewed with ParaView as follows:

- Start ParaView at the command line with `$ paraview &`, then navigate to and open the fields result `coaxial_waveguide_frequency_1e+10_mode_1.pvd`
- Run `Macros`→`field_plot`. Wait a few moments.
- A wide variety of plots are available for viewing, both magnitude and vector.
- Disable all plots then enable `Ht real magnitude` and `H real vector`.
- Click `H real vector` then enter 1e-5 for `Scale Factor`.

- The resulting field is shown in Fig. 7, where the magnetic field circles the center conductor.

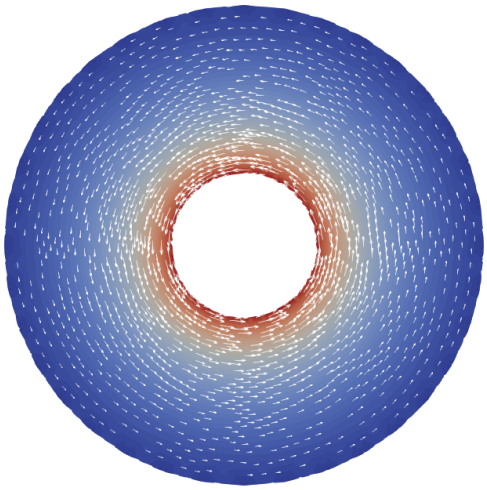


Figure 7: Re(Ht) for the RG401 coaxial waveguide.

8.3 Coupled Microstrip with Line Setup

The coupled microstrip pair from Fig. 8 in [7] is set up with line integration paths and solved for the even and odd characteristic impedances and propagation constants.

8.3.1 Drawing and Mode Annotation

- As for the prior tutorials, start a new project called `coupled_microstrip` with a new drawing in FreeCAD with the drawing plane set to `Top`.
- Draw a rectangle 1 um wide and 0.3 um tall [to ultimately result in 1 and 0.3 mm], and set the placement to x=-1.5 um and y=0. Zoom to fit. Set Label to `cut1`.
- Select the rectangle and copy it using `Edit`→`Copy` then `Edit`→`Paste`. Set Label to `_Pi1`. This is the current integration path for the first microstrip line.
- Select object `_Pi1` and copy it. Set the placement to x=0.5 um and y=0. Change Label to `cut2`.
- Select object `cut2` and copy it. Set Label to `_Pi2`. This is the current integration line for the second microstrip line.
- Select object `_Pi2` and copy it. Set Label to `substrate`. Set Height to 0.635 um and the width to 10 um. Set the position to x=-5 um and y=-0.635 um.
- Select object `substrate` and copy it. Set Label to `air`. Set Height to 3 um. Set the position with y=0.
- Set the drawing workspace to `Part`.
- Select `air` then `cut1`, then select `Part`→`Boolean`→`Cut`. Click `Yes` in the pop-up.
- Select the cut object then `cut2` and repeat the Boolean cut operation. Set Label to `air-metal`. The rectangular voids form the microstrip lines since all edges next to void spaces default to PEC.
- Select `View`→`Visibility`→`Hide all objects`.
- Save the drawing as `coupled_microstrip`.

- Select `substrate` and `air-metal` and press the space bar to view the objects.
- Set the drawing workspace to `Draft`.
- Set snap options to include `Snap Midpoint`.
- Draw a line from the center at the bottom of `substrate` to the center at the top of `substrate`. Set Label to `_Pv1`. Set the position to x=-1 um. This is the voltage integration line for the first microstrip line.
- Copy `_Pv1`, set Label to `_Pv2`, and set the position x=1 um. This is the voltage integration line for the second microstrip line.
- Add a text object with the Label `_L1(voltage){v1}`. This is the line definition for the voltage on the first microstrip.
- Add a text object with the Label `_L2(voltage){v2}`. This is the line definition for the voltage on the second microstrip.
- Add a text object with the Label `_L1(current){i1}`. This is the line definition for the current on the first microstrip.
- Add a text object with the Label `_L2(current){i2}`. This is the line definition for the current on the second microstrip.
- The drawing should look like that in Fig. 8, showing the substrate and air above the substrate along with the voltage integration lines.

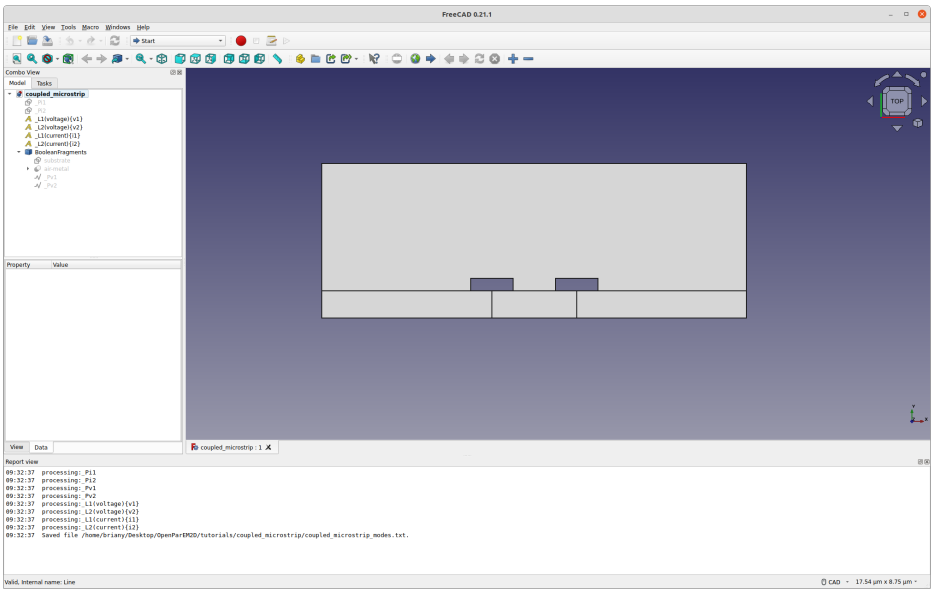


Figure 8: Coupled microstrip with annotations.

- Set the drawing workspace to `Part`.
- Select `substrate`, `air-metal`, `_Pv1`, and `_Pv2`, then select `Part`→`Split`→`Boolean Fragments`. By including the voltage integration lines, gmsh will include these an an edge in the mesh.
- Execute the macro `OpenParEM2D_save.py` to save the mode description file as `coupled_microstrip_modes.txt`.
- Save the drawing and exit FreeCAD.

8.3.2 Meshing

- Start gmsh and open the saved BREP file.
- Assign physical groups `air` to the surface representing air and `alumina` to the surface representing the substrate using the `Add Surface` methodology described in the prior tutorials.
- Mesh for 2D then save the mesh.
- Quit gmsh.

8.3.3 Solving

- Create the materials file `local_materials.txt` and set the text contents to

```
#OpenParEMmaterials 1.0
Material
   name=air
   Temperature
      temperature=any
      Frequency
         frequency=any
         er=1.0006
         mur=1
         tand=0
         Rz=0
      EndFrequency
   EndTemperature
   Source
      Constantine A. Balanis, "Advanced Engineering Electromagnetics",
      John Wiley and Sons, 1989, p.79.
   EndSource
EndMaterial
Material
   name=alumina
   Temperature
      temperature=any
      Frequency
         frequency=any
         er=9.8
         mur=1
         tand=0
         Rz=0
      EndFrequency
   EndTemperature
   Source
   EndSource
EndMaterial
```

- Create the project control file `coupled_microstrip.proj` set the text contents to

```
#OpenParEM2Dproject 1.0
project.save.fields           true
mesh.file                     coupled_microstrip.msh
mesh.order                    4
mesh.refinement.fraction      0.01
mode.definition.file          coupled_microstrip_modes.txt
materials.global.path         ./
materials.global.name         //global_materials.txt
materials.local.path          ./
```

```
materials.local.name           local_materials.txt
refinement.frequency           high
frequency.plan.point           40.11e9
refinement.variable            |Zo|
refinement.iteration.min       1
refinement.iteration.max       40
refinement.required.passes     1
refinement.tolerance           0.0001
solution.modes                 2
solution.modes.buffer          0
solution.impedance.definition  PI
solution.impedance.calculation line
debug.show.impedance.details   true
```

- Run OpenParEM2D with the proj file.

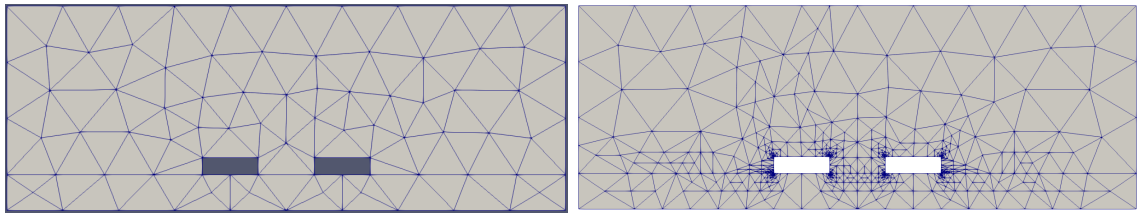
8.3.4 Results

OpenParEM2D calculates the characteristic impedances for common and differential modes, while [7] quotes results for the even and odd modes. To compare, the OpenParEM2D results are converted so that the even mode characteristic impedance is calculated as twice the common mode impedance, while the odd mode impedance is half the differential impedance. The results are shown in Table 3, where the [7] results are scaled off the printed plot. The results are in excellent agreement with a small difference in the odd mode characteristic impedance.

Table 3: Comparison of Results

Characteristic	Simulation of [7]	OpenParEM2D
$\beta_e/k_\circ$	2.93	2.94
$\beta_o/k_\circ$	2.82	2.82
$Z_{\circ e}, \Omega$	50.8	50.6
$Z_{\circ o}, \Omega$	43.4	42.6

Due to the sharp corners on the microstrip lines, adaptive mesh refinement is used in this solution. The initial and final refined meshes are shown in Fig. 9, where the refinement concentrates on the corners and the dielectric interface.

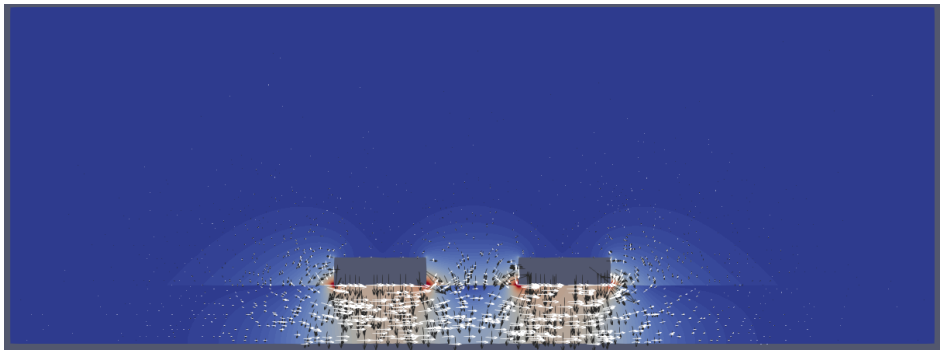


(a) Initial mesh as produced by gmsh.

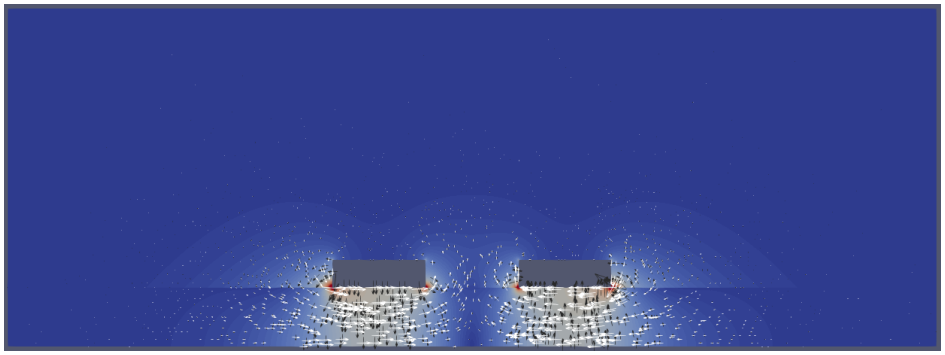
(b) Refined mesh after 26 iterations.

Figure 9: Initial and refined meshes for the coupled microstrip lines.

The fields can be viewed with ParaView, where pre-configured plots are generated using the macro `field_plot`. Plots for the common and differential modes are shown in Fig. 10. In these plots, the background shows $\mathrm{Re}(|\overline{E}|)$, while the black vectors show $\mathrm{Re}(\overline{E})$ and the white vectors show $\mathrm{Re}(\overline{H})$. Note that $\mathrm{Re}(\overline{E})$ and $\mathrm{Re}(\overline{H})$ do not use the same scale.



(a) Common mode.



(b) Differential mode.

Figure 10: Plots of the common-mode and differential-mode fields.

8.4 Coupled Microstrip with Modal Setup

The tutorial from Sec. 8.3 is re-worked using modal setups for the voltage and current integration paths. The setups are nearly identical, so just the changes are discussed.

8.4.1 Drawing and Mode Annotation

- Copy the tutorial project directory from `coupled_microstrip` to `coupled_microstrip_modal`.
- In FreeCAD, change `_L1(voltage){v1}` to `_M1(voltage,0.5){v1,v2}`. Mode 1 is the common-mode mode, so the voltages on the lines are equal due to symmetry. The modified annotation has OpenParEM2D integrate along both paths, so the voltages is 2× too high, so a scaling factor of 0.5 is applied.
- Change `_L2(voltage){v2}` to `_M2(voltage){v1,-v2}`. Mode 2 is the differential mode, so the voltages are equal and opposite. The modified annotation has OpenParEM2D integrate along the v1 path and then the v2 path in the reversed direction. The total voltage is then found without need for a scaling factor.
- Change `_L1(current){i1}` to `_M1(current){i1,i2}`. For the common mode, the total current is the sum of the currents on the two lines.
- Change `_L2(current){i2}` to `_M2(current,0.5){i1,-i2}`. For the differential mode, the currents are equal and opposite. The modified annotation has OpenParEM2D calculate the total currents on both lines then applies a scaling factor of 0.5.
- Execute the macro `OpenParEM2D_save.py` to save the mode description file as `coupled_microstrip_modes.txt`.

- Save the drawing and exit FreeCAD.

8.4.2 Meshing

The existing mesh is re-used.

8.4.3 Solving

In the proj file, change the value for the keyword `solution.impedance.calculation` from `line` to `modal`. Run OpenParEM2D.

8.4.4 Results

After 26 iterations, the results shown to the terminal are

```
-------------------------------------------------------------------------------------------------------------------------
mode     frequency           er,eff       alpha,dB/m  beta/1000,rad/m          beta/ko          Zo(real)          Zo(imag)
-------------------------------------------------------------------------------------------------------------------------
1        4.011e+10        8.6544292    2.1815834e-11        2.4730394        2.9418411        25.279915    2.7033135e-14
2        4.011e+10        7.9720613    1.3477717e-11        2.3735432        2.8234839        85.178997    3.0707696e-14
-------------------------------------------------------------------------------------------------------------------------
```

A run from the line setup [also after 26 iterations] in Sec. 8.3 shows nearly identical results as

```
-------------------------------------------------------------------------------------------------------------------------
mode     frequency           er,eff       alpha,dB/m  beta/1000,rad/m          beta/ko          Zo(real)          Zo(imag)
-------------------------------------------------------------------------------------------------------------------------
1        4.011e+10        8.6544292    -4.079956e-11        2.4730394        2.9418411        25.279915   -4.2724076e-14
2        4.011e+10        7.9720613    7.4035296e-12        2.3735432        2.8234839        85.127309    4.7404632e-14
-------------------------------------------------------------------------------------------------------------------------
```

where the only differences are the very minor changes due to the iterative nature of the solution interacting with MPI processing.

Impedance calculations are a post-processing step, so the field solution is the same for both setups. The differences are simply that the line setup calculates voltages and currents on partial paths then combines them to get final values, while the modal setup calculates voltages and currents on total paths to get final values without further processing. The two techniques must necessarily get the same results.

9 Techniques

9.1 Finite Element Order

The finite element order is set with the keyword/value pair `mesh.order N` in the project control file. `N` can be any integer and sets the order of the polynomial approximating the field in each finite element. All elements in a mesh have the same element order.

Higher values for `N` have potential advantages due to the polynomial accommodating larger field variations within any given element. The advantages of higher order include:

- Greater accuracy for a given mesh size.
- Smaller mesh size for a given accuracy.
- Better accuracy for frequency sweeps.
- Smoother fields for visualization.

However, there are also disadvantages of higher order:

- Slower run times for a given mesh size.
- Larger memory consumption.
- Slower mesh error calculation for a given mesh size.

- Harder convergence for the initial solution, although order-ramping may offer a fix, when implemented.

The disadvantages ramp quickly with increasing `N`. OpenParEM2D does not currently support GPU computing, which can alleviate the disadvantages of higher `N`.

Experience to date suggests the following guidelines for selecting `N`:

- Assuming a simulation runs to completion in an acceptable time, higher `N` is always better than lower `N`.
- Avoid `N=1`. Adaptive refinement is required and the number of iterations is just too great. Improved adaptive meshing algorithms might address this issue.
- Generally, `N=3` and `N=4` are good options.
- Choose the largest `N` that allows a quick first iteration when using adaptive refinement. What qualifies as *quick* depends on mesh size.
- In the first iteration, if it take more than a few seconds or requires a very large iteration limit [like 100,000], then decrease `N`. Conversely, if the first iteration is very quick, then try increasing `N`. Note that very slow convergence can be caused by poor mesh quality, so it is important to ensure that the mesh has good triangulation by avoiding long thin triangles.
- Problems with relatively smooth fields, such as those using waveguide, greatly benefit from larger N. For some problems, adaptive refinement may not be needed.
- Problems with sharp edges, such as edge-coupled stripline, benefit from a highly refined mesh around the edges. To keep run times reasonable, it can be beneficial to use `N=2`.
- If using iterative refinement, and the number of iterations is very large, then an increase in `N` could be in order. Conversely, if only a couple of iterations are needed, then a smaller `N` could be useful to force an increase in the number of iterations to help uncover potential areas where additional refinement would improve accuracy.
- ParaView supports `N` up to 6. If ParaView will be used to review field plots, then keep `N` to 6 or fewer.

9.2 Over-Meshing

All computations involve finite precision, and it is possible to demand more precision from a computer than it can provide, in which case accuracy can degrade. For finite element programs such as OpenParEM2D, excess precision requirements occur when a problem has far more mesh density than that required to accurately solve a problem. This is called *over-meshing*.

Consider an example of a problem minimally meshed such that the geometry is barely but accurately captured by the mesh and that first-order finite elements are used to solve the problem with adaptive mesh refinement. At the first iteration, the computed result will have poor accuracy because the linear finite elements cannot capture the curvature of the fields. Adaptive refinement subdivides the mesh in areas of high error, meaning areas where the linear approximation is poor, thereby improving the ability of the linear elements to describe the field. This process continues with additional refinements until the linear approximation provides a reasonable fit to the field curvature in all areas.

Now, what happens if adaptive refinement continues after the approximation of the linear finite elements already provides a good fit to the field curvature? Certainly, the fit of the field gets better. However, a problem begins to develop where the improvement starts to push down to lower decimal places. So with hypothetical numbers, the first iteration provides a 10% improvement, the second 1%, third 0.1%, fourth 0.01%, etc. Eventually, the required precision runs out of decimals for accurate calculation, and the computed solution starts losing accuracy. More iterations just makes it worse.

The same effect can happen by choosing mesh orders that provide too much variability required by the problem. For example, a perfectly linear field structure would be accurately described by both a linear finite element and a second-order finite element. However, for the second-order element, the computation must

accurately zero out the quadratic term, which requires extra digits of precision. Once a field is accurately captured by a finite element order, going substantially higher in order can actually decrease accuracy.

The key is to avoid relying on excessive numerical precision by not over-meshing a problem. The concept is the same whether low- or high- order finite elements are used. Generally, the higher the order of the finite elements, the less dense the mesh needs to be to avoid over-meshing.

Best practice is to simply mesh the problem with minimal settings and then let adaptive mesh refinement or stepping up the finite element order converge the solution to engineering accuracy then stopping. This practice also results in minimal run time. The goal is really to just not "play it safe" by targeting excessively high accuracies. See also the comments in the "Lossy Stripline" accuracy demonstration in the companion document "OpenParEM2D_Theory_Methodology_Accuracy.pdf".

9.3 Very Coarse Meshes

Very coarse meshes can cause convergence issues with higher-order modes. An example would be just 4 triangles to mesh a rectangular waveguide. A simple remedy should this occur is to set `mesh.uniform_refinement.count` to 1 or perhaps 2 to increase the mesh density. Otherwise, a new mesh with higher density is needed.

9.4 Mesh Quality

The largest aspect ratio of any element in the mesh is reported as a measure of mesh quality, with the output looking something like

```
mesh worst element aspect ratio: 2.2595 < target: 5
```

The target is just a reminder of a level that indicates a very good mesh. As the aspect ratio climbs above the target, convergence of the eigenvalue solution degrades and the resulting fields are not as good. Aspect ratios of 10 are still good, but by 20, the solution will often start showing signs of stress. Higher order finite elements have more trouble with high aspect ratios than lower order finite elements.

9.5 Adaptive Mesh Refinement

OpenParEM2D supports adaptive mesh refinement, where mesh elements with high errors are subdivided before re-solving the problem. Iterations stop when the convergence criteria are met.

The fraction of the mesh that is refined on any iteration is given by `mesh.refinement.fraction`. Mesh errors are calculated for each mesh element, sorted, and then the top `mesh.refinement.fraction` are selected for refinement. If high errors are concentrated in a few elements, such as for microstrip and stripline, then a lower `mesh.refinement.fraction` prevents low-error elements from being refined. A suggested value for `mesh.refinement.fraction` in this situation is 0.01. Conversely, if all of the elements have similar errors, such as for rectangular waveguide and coax, then a higher `mesh.refinement.fraction` can speed up convergence. A suggested value for `mesh.refinement.fraction` in this situation is the default value of 0.025.

The MFEM library does not support de-refinement for conformal meshes, and the flow presented here uses gmsh, which only supports conformal meshes. Once an element is refined, it stays refined even if later refinements improve the field such that it no longer needs to be refined. Since refinement is one-way, it is worthwhile to refine relatively slowly to minimize the number of relatively low-error elements that get refined. Ultimately, less refinement at each iteration produces more iterations but at times a faster overall run time.

Note that very low values of `mesh.refinement.fraction` should be matched with a reduced value of `solution.tolerance` to avoid pre-mature convergence. For example, if just one mesh element in a large mesh is refined, then clearly the answer cannot change very much and a smaller `solution.tolerance` would be in order.

9.6 Conductor Loss

Conductor losses are added with boundaries defined with "SI" along with a conductor material. The problem is first solved with all metals being ideal (PEC), and this solution includes dielectric losses. Then conductor losses are computed as a post-processing step and added to the dielectric loss to get the total loss. With this style of perturbational conductor loss calculation, field penetration through conductors is not supported. All metals must be thick compared to the skin depth for the conductor loss calculation to be accurate, and it is up to the user to ensure that this condition is met.

Conductor losses have a very small effect on the propagation constant. Since the eigenvalue problem solves for the propagation constant with PEC and PMC boundaries, the propagation constant does not include any effects due to conductor losses. OpenParEM3D does include the effect of conductor losses on delay, so if this effect needs to simulated, then a straight section of transmission line or waveguide can be solved for the delay with OpenParEM3D from which the propagation constant can be calculated.

9.7 Convergence Difficulties of the Eigenvalue Solution

Difficulty achieving convergence generally occurs in two scenarios: poor mesh quality and/or large frequency step. In cases where convergence is very slow, the best strategy is to re-mesh the problem to improve the mesh quality. The next best strategy is to reduce the order of the finite elements. In some cases, relaxing `solution.tolerance` can be sufficient, although tolerances as low as 1e-08 can start to impact accuracy.

Since the complex propagation constant from the prior iteration is used as the initial guess for the next iteration, large jumps in frequency can cause convergence issues since the larger the step, the less relevant is the initial guess. The remedy in this situation is to increase the frequency density.

A special case of a large frequency step is when adaptive mesh refinement occurs at a high frequency followed by a frequency sweep starting at a very low frequency. If convergence at the low frequency is proving difficult, then adaptive refinement can be added at lower frequencies to step down the frequency so that the initial guess is better. For example, suppose a problem is adaptively refined at 50 GHz with a subsequent sweep starting at 0.1 GHz. The complex propagation constant at 50 GHz is used as the initial guess at 0.1 GHz, for a 500:1 step down in frequency. To reduce the size of the step, adaptive mesh refinement at 50 GHz can be followed by adaptive mesh refinement at 5 GHz and then additionally at 0.5 GHz so that the maximum step down in frequency for the initial guess is never more than 10. [In a case like this, to avoid excessive refinement and very long run times, the convergence tolerance can be relaxed since multiple rounds of adaptive refinement is applied and good accuracy will still be obtained.]

9.8 Missing Higher-Order Modes

Along with the desired results, OpenParEM2D also finds the null space result, which is essentially the trivial field result of 0. The null space results are filtered out, and just the usable results are shown. To find N good results, OpenParEM2D has to solve for $N + M$ modes, where M can be 0 to 10 or higher. The count of extra modes to solve is given by `solutions.mode.buffer`, and a typical value is either 0 or 5, but it can be higher. If OpenParEM2D does not find as many modes as requested, simply increase `solutions.mode.buffer` and re-run. Since the eigenvalue solver must solve for $N + M$ modes, and more modes takes more time to solve, it is beneficial to run times to keep M as low as possible.

9.9 MPI and Iterative Solvers

MPI works by dividing a matrix by rows and spreading them across multiple cores. Exactly how the split is done depends on the current loading of the computer, so there is variability from run-to-run. The cores communicate with each other with packets of information and the timing of these also depend on loading, and the order of execution can change from run-to-run. OpenParEM2D uses iterative solvers for the eigenvalue solution of $\overline{E}$ and the linear solution for $\overline{H}$, and the progression of the iterations changes slightly because the data used at each iteration changes due to the variation in the data split and timing of MPI data packets. The net result is that the answers produced by OpenParEM2D changes slightly from run-to-run. If a single core is used, then there is no subdividing of the matrix nor MPI communication, so identical results are produced from every run.

A low-accuracy simulation demonstrates higher variation from run-to-run than a high-accuracy simulation. This behavior can be used to build confidence that a solution is converged by re-running the simulation with a different number of cores and seeing how much the answer changes. Changing the number of cores forces a change in the split of the matrices across cores and has more effect on the computed results. If the number of cores is not changed, it is possible that the split is identical with similar timings so that the final result does not actually change much.

9.10 Companion Tool builder

A companion tool called *builder* helps to quickly build projects for some common transmission line and waveguide types. The target use for builder is optimization of dimensions and materials to reach a target characteristic impedance. A simple text file with keyword/value pairs describes the physical aspects of the problem to be solved, then builder constructs all of the needed files for OpenParEM2D with the exception of the mesh, which still must be generated in gmsh. However, in gmsh the materials are already defined, so the process can be as simple as opening the geo file, meshing, then exiting. The template proj file can be touched up with the remaining specifics for a given simulation. Builder is documented in "builder_User_Manual.pdf".

A Control File Specification

OpenParEM2D is controlled by a text file consisting of keyword/value pairs. The rules for setting up the control file are listed below.

- The first line of the file must be `#OpenParEM2Dproject 1.0`
- One keyword/value pair per line
- Except for file names, each keyword has a default value
- Keyword/value pairs can appear in the file more than once, and excepting frequency.plan.*, the last entry is the one that is used. Note: No error message or warning is issued when keyword values are overwritten.
- All units are MKS: meter, Ohms, Hz, S/m, Celsius

Keyword	Value	Default	Description
project.save.fields	bool	false	Save the vector field results for viewing with ParaView
mesh.file	string		File name of the mesh file
mesh.order	int	1	Order of the finite elements
mesh.uniform_refinement.count	int	0	Number of times the mesh is uniformly refined before starting the simulation
mesh.refinement.fraction	double	0.025	Maximum fraction of the mesh to refine at each iteration
mesh.enable.refine	bool	true	Prevents re-ordering of DOFs when loading meshes. Used by OpenParEM3D, and otherwise, no effect.
mode.definition.file	string		File name for the boundary condition/mode file
materials.global.path	string	../	Path to a global materials file serving as a library. Can also be blank or ./ for the local directory
materials.global.name	string	global_materials	File name for the global materials file
materials.local.path	string	./	Path to a local materials file. Can also be blank.
materials.local.name	string	local_materials	File name for the local materials file
materials.check.limits	bool	true	Check the material values against range limits
refinement.frequency	string	highlow	Sets the frequency or frequencies on which adaptive mesh refinement is applied. Options: • none - refine at no frequencies and simulate with the initial mesh • all - refine at each frequency starting from the initial mesh • high - refine at the highest frequency then simulate at all frequencies with that mesh • low - refine at the lowest frequency then simulate at all frequencies with that mesh • highlow - refine at the highest then lowest frequencies then simulate at all frequencies with that mesh • lowhigh - refine at the lowest then highest frequencies then simulate at all frequencies with that mesh • plan - refine at the frequencies marked in the frequency plan then simulate at all frequencies with that mesh

Keyword	Value	Default	Description
refinement.variable	string	—gamma—	Variable on which to test convergence during adaptive refinement Options: • alpha - converge on the real part of the complex propagation constant (loss) • beta - converge on the imaginary part of the complex propagation constant (propagation constant) • —gamma— - converge on the magnitude of the complex propagation constant • —Zo— - converge on the magnitude of the characteristic impedance • Re(Zo) - converge on the real part of the characteristic impedance • Im(Zo) - converge on the imaginary part of the characteristic impedance
refinement.iteration.min	int	1	Minimum number of iterations to perform
refinement.iteration.max	int	10	Maximum number of iterations to perform
refinement.required.passes	int	3	The number of consecutive iterations that must meet the refinement tolerance
refinement.tolerance	double	0.001	Relative tolerance for convergence during adaptive refinement
refinement.refine.converged.modes	bool	true	Continue refining on modes even after initial convergence. Setting to false has not been thoroughly evaluated
frequency.plan.log	string	none	Adds solution frequencies to the frequency plan using a log scale with the comma-separated list start,stop,pointsPerDecade
frequency.plan.log.refine	string	none	Same as frequency.plan.log plus mesh refinement is applied at these frequencies
frequency.plan.linear	string	none	Adds solution frequencies to the frequency plan using a linear scale with the comma-separated list start,stop,step
frequency.plan.linear.refine	string	none	Same as frequency.plan.linear plus mesh refinement is applied at these frequencies
frequency.plan.point	double	none	Adds the given solution frequency to the frequency plan
frequency.plan.point.refine	double	none	Same as frequency.plan.point plus mesh refinement is applied at the given frequency
solution.modes	int	1	Number of modes to solve
solution.temperature	double	25	Temperature used for materials selection
solution.tolerance	double	1e-13	Tolerance for the eigenvalue solution and H field calculation
solution.iteration.limit	int	5000	Iteration limit for the iterative eigenvalue and Hfield solvers
solution.modes.buffer	int	5	Additional number of modes over solution.modes to solve. Increase this value if not all of the solution.modes solutions are found
solution.impedance.definition	string	none	Definition used to calculate the characteristic impedance Options: • none - skip the calculation • PV - use the power-voltage definition • PI - use the power-current definition • VI - use the voltage-current definition
solution.impedance.calculation	string	modal	Setup used for defining the voltage and current integration paths
solution.check.closed.loop	bool	true	Enables checking if current integration paths form closed loops
solution.accurate.residual	bool	false	Sets an input parameter to the eigenvalue solver that may produce higher accuracy results. In many cases, the eigenvalue solver will hang with this is set to true

Keyword	Value	Default	Description
solution.shift.invert	bool	true	Sets the eigenvalue solver to use the shift-and-invert method
solution.use.initial.guess	bool	true	Use the current eigenvalue solution as the initial guess in the following iteration
solution.shift.factor	double	1.0	Multiplier for the initial guess eigenvalue
solution.initial.alpha	double	0	Initial guess for alpha, where the complex propagation constant is alpha+jbeta
solution.initial.beta	double	0	Initial guess for beta, where the complex propagation constant is alpha+jbeta
output.show.refining.mesh	bool	false	Show details about the mesh during adaptive refinement
output.show.postprocessing	bool	false	Show details about the post-processing steps after the eigenvalue solution is found
output.show.iterations	bool	false	Show the eigenvalue solve iterations
output.show.license	bool	false	Show the license governing use of the software
test.create.cases	bool	false	Create test cases useful for setting up regression testing
test.show.audit	bool	false	Show summary results when performing regression testing with the program called "process". This keyword is not used by OpenParEM2D
test.show.detailed.cases	bool	false	Show detailed information about the test cases when regression testing with the program called "process". This keyword is not used by OpenParEM2D
debug.show.memory	bool	false	Show memory usage at strategic times to look for memory leaks. Not very comprehensive
debug.show.project	bool	false	Show the full set of project keywords and their values
debug.show.frequency.plan	bool	false	Show the full set of simulation and refinement frequencies
debug.show.materials	bool	false	Show the full set of materials from the material databases
debug.show.mode.definitions	bool	false	Show the setups used for mode definitions
debug.show.impedance.details	bool	false	Show all voltages, currents, powers, and impedance calculations
debug.skip.solve	bool	false	Apply all setup actions but skip solving
debug.tempfiles.keep	bool	false	Keep temporary files
field.point	double,double	none	Print out the field values at the x,y point in the cross section. Any number of field points can be specified. Primarily used for regression testing

B Boundary/Mode File Specification

Modes and boundaries are specified in a text file with information on locations and type. The file is generally referred to as the "mode definition file" even though it contains both mode and boundary specifications. Paths are first defined to provide physical locations, then modes and boundaries use the paths to complete the setups. Paths can be re-used across modes and boundaries.

```
// a comment

#OpenParEMmodes 1.0   // required on first real line

// All coordinates are in m.

// This block is informational use only.
// Only one File/EndFile block can be specified.
File
   name=string  // name of the file from which the lines are generated
EndFile

// Paths are used to define voltage integration lines, current integration loops,
// and locations for boundary conditions.
// Any number of paths can be defined.
// Paths are not required to be used.
// Names must be unique within all path definitions.
Path
   name=string
   point=(double,double)      // (x,y)
   point=(double,double)      // any number of points
   ...
   point=(double,double)
   closed=bool                // false if the path is open and true if the path is closed
EndPath

// For closed loops, do not duplicate the starting and stopping points.  Using closed=true
// closes the loop during calculations.

// Specify any number of boundaries.
// The physical placement of the boundary is defined by the paths.
// One path can be used if it is complete, and paths can be chained together to form more
// complex physical setups.
// Names must be unique within all boundary definitions.
Boundary
   name=string
   type=surface\_impedance|perfect\_electric\_conductor|perfect\_magnetic\_conductor
   material=string            // required for surface impedance
   path=name1                 // no sign means that the direction of the path is unchanged
   path-=name2                // minus sign means that the direction of the path is reversed
   ...
   path+=name3                // plus sign means that the direction of the path is unchanged
EndBoundary

// Specify any number of modes.
// Use with solution.impedance.calculation set to "modal".
// One Mode/EndMode block is applied to one solved waveguide/transmission line mode.  N
// modes requires N blocks. For N modes, all mode numbers are required from 1 to N, so
// mode definitions are required for modes 1, 2, ..., N. One path can be used if it is
// complete, and paths can be chained together to form more complex physical setups.
// One voltage definition and/or one current definition can be provided per mode number.
// 0 is not an allowed mode number
```

```
Mode
   mode=integer
   type=voltage|current
   [scale=double]  // default=1
   path=+name
   path-=name
   ...
   path=name
EndMode

// Specify any number of lines.
// Use with solution.impedance.calculate set to \line".
// One Line/EndLine block is applied to each conductor.  N conductors requires N blocks.
// For N lines, all line numbers are required from 1 to N, so line definitions are required
// for lines 1, 2, ..., N. One path can be used if it is complete, and paths can be chained
// together to form more complex physical setups. One voltage definition and/or one current
// definition can be provided per line number.
// 0 is not an allowed line number
Line
   line=integer
   type=voltage|current
   [scale=double]   // default=1
   path=+name
   path-=name
   ...
   path=name
EndLine

// The Mode/EndMode and Line/EndLine blocks are interchangeable.  Two block types are
// supported so that the block naming is coordinated with the impedance calculation type.
```

C Regression Suite

The projects are located in the installation area in the "regression/OpenParEM2D" directory.

Project	Description
coax_modal/coax.proj	Coax with comparison to analytical results using modal definitions
coax_line/coax.proj	Coax with comparison to analytical results using line definitions
coaxEighth/coax.proj	1/8 coax to test PMC boundary with comparison to analytical results
coaxEighthAlt/coax.proj	1/8 coax with re-located integration line with comparison to analytical results
differential_pair/diff_pair_modal/diffPair.proj	Differential pair with comparison to literature simulation using modal definitions
differential_pair/diff_pair_line/diffPair.proj	Differential pair with comparison to literature simulation using line definitions
differential_pair/diff_pair_modal_symmetry_even/diffPair.proj	1/2 differential pair using symmetry with comparison to literature simulation using modal definitions
differential_pair/diff_pair_modal_symmetry_odd/diffPair.proj	1/2 differential pair using symmetry with comparison to literature simulation using modal definitions
differential_pair/diff_pair_line_wide_spacing/diffPair.proj	Widely spaced differential pair using line definitions to test for accommodation of sign flips
WR90_rectangular_waveguide/WR90_PMC_symmetry/WR90half.proj	1/2 waveguide testing PMC boundary with comparison to analytical results
WR90_rectangular_waveguide/WR90/WR90_order_1_refinement_no_mesh_reuse/WR90.proj	Waveguide with higher-order modes with comparison to analytical results for 1^{st}-order finite elements
WR90_rectangular_waveguide/WR90/WR90_order_2_refinement_mesh_reuse/WR90.proj	Waveguide with higher-order modes with comparison to analytical results for 2^{nd}-order finite elements
WR90_rectangular_waveguide/WR90/WR90_order_3_no_refinement_no_mesh_reuse/WR90.proj	Waveguide with higher-order modes with comparison to analytical results for 3^{rd}-order finite elements
WR90_rectangular_waveguide/WR90/WR90_order_3_refinement_no_mesh_reuse/WR90.proj	Waveguide with higher-order modes with comparison to analytical results for 3^{rd}-order finite elements
WR90_rectangular_waveguide/WR90/WR90_order_3_mixed_refinement/WR90.proj	Waveguide with higher-order modes with comparison to analytical results for 3^{rd}-order finite elements
WR90_rectangular_waveguide/WR90/WR90_order_4_refinement/WR90.proj	Waveguide with higher-order modes with comparison to analytical results for 4^{th}-order finite elements
WR90_rectangular_waveguide/WR90/WR90_order_4_refinement_no_mesh_reuse_loss/WR90.proj	Waveguide with higher-order modes with comparison to analytical results for 4^{th}-order finite elements
WR90_rectangular_waveguide/WR90/WR90_order_5_refinement_mesh_reuse/WR90.proj	Waveguide with higher-order modes with comparison to analytical results for 5^{th}-order finite elements
WR90_rectangular_waveguide/WR90/WR90_order_6_norefinement/WR90.proj	Waveguide with higher-order modes with comparison to analytical results for 6^{th}-order finite elements
permeability/rectangular_waveguide/WR90.proj	Waveguide with higher-order modes with comparison to analytical results for $\mu_r = 2$
partially_filled_rect_waveguide/PartFilled_order_3_norefinement/PartFilled.proj	Partially filled rectangular waveguide with comparison to semi-analytical results for 3^{rd}-order finite elements
partially_filled_rect_waveguide/PartFilled_order_3_refinement/PartFilled.proj	Partially filled rectangular waveguide with comparison to semi-analytical results for 3^{rd}-order finite elements
partially_filled_rect_waveguide/PartFilled_order_4_norefinement/PartFilled.proj	Partially filled rectangular waveguide with comparison to semi-analytical results for 4^{th}-order finite elements
partially_filled_rect_waveguide/PartFilled_order_4_norefinement_widebandwidth/PartFilled.proj	Partially filled rectangular waveguide with comparison to semi-analytical results for 4^{th}-order finite elements
partially_filled_rect_waveguide/PartFilled_order_5_norefinement/PartFilled.proj	Partially filled rectangular waveguide with comparison to semi-analytical results for 5^{th}-order finite elements
partially_filled_rect_waveguide/PartFilled_order_6_norefinement/PartFilled.proj	Partially filled rectangular waveguide with comparison to semi-analytical results for 6^{th}-order finite elements
permeability/partially_filled_rect_waveguide/PartFilled.proj	Partially filled rectangular waveguide with comparison to semi-analytical results for $\mu_r = 2$

Project	Description
Lee_microstrip/full_line/Lee_microstrip.proj	Microstrip with comparison to literature simulation using line setups
Lee_microstrip/full_line_omit_voltage/Lee_microstrip.proj	Microstrip testing setup variation with comparison to literature simulation using line setup
Lee_microstrip/full_line_omit_current/Lee_microstrip.proj	Microstrip testing setup variation with comparison to literature simulation using line setup
Lee_microstrip/full_modal/Lee_microstrip.proj	Microstrip with comparison to literature simulation using modal setups
Lee_microstrip/full_modal_omit_voltage/Lee_microstrip.proj	Microstrip testing setup variation with comparison to literature simulation using modal setup
Lee_microstrip/full_modal_omit_current/Lee_microstrip.proj	Microstrip testing setup variation with comparison to literature simulation using modal setup
Lee_microstrip/half/Lee_microstrip.proj	1/2 microstrip testing PMC boundary with comparison to literature simulation
Simonovich_stripline/Simonovich_stripline.proj	Stripline with comparison to literature measurement

D Accuracy Studies

The projects are located in the installation area in the "regression/OpenParEM2D" directory. The cases are discussed in detail in "OpenParEM2D_Theory_Methology_Accuracy.pdf".

Project	**Description**
Simonovich_stripline_study/Simonovich_stripline.proj	Stripline with comparison to measurement and simulation from the literature
WR90_rectangular_waveguide/WR90/WR90_order_6_lossy_study/WR90_accuracy_run.proj	Lossy rectangular waveguide with comparison to exact results
WR90_rectangular_waveguide/WR90/WR90_order_6_study	Lossless rectangular waveguide with comparison to exact results
coaxEighth_study/coax_accuracy_run.proj	Coax with comparison to exact results
partially_filled_rect_waveguide/PartFilled_order_6_study/PartFilled_accuracy_run.proj	Partially filled rectangular waveguide with comparison to analytical results
differential_pair/diff_pair_study/diffPair.proj	Differential pair with comparison to simulation from the literature

References

[1] C. Geuzaine and J.-F. Remacle, "Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, 79(11), pp. 1309-1331, 2009.

[2] `https://gmsh.info` version 4.13

[3] `https://freecad.org` version 0.21

[4] `https://www.paraview.org`

[5] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, Tz. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini, "MFEM: A modular finite element methods library", *Computers and Mathematics with Applications*, vol. 81, 2021, pp. 42-74.

[6] `https://mfem.org`

[7] Frank Olyslager, Daniel De Zutter, and Krist Blomme, "Rigorous analysis of the propagation characteristics of general lossless and lossy multiconductor transmission lines in multilayered media", *IEEE Trans. Microwave Theory and Techniques*, vol. 41, no. 1, Jan. 1993, pp. 79-88.