OpenParEM3D User's Manual

Version 1.0

September 2024

Brian Young

Contents

1 Introduction

OpenParEM3D is a full-wave electromagnetic simulator that solves the for the frequency-dependent vector electric ($\overline{E}$) and magnetic ($\overline{H}$) fields of general 3D structures and post-processes the fields to produce scattering parameters (S-parameters) between 2D ports. It is a free and open-source project released under the GPL3 license.

OpenParEM3D solves the full set of Maxwell's Equations (hence, full-wave) in the frequency domain. Ports on the outer surface of the 3D space enable energy to enter and leave. It is assumed that the ports are 2D and represent transmission lines or waveguides with complex propagation constants. At each port, OpenParEM2D solves for the full-wave 2D fields, characteristic impedance of the dominant and optional higher-order modes, and the complex propagation constants and applies the fields in the 3D solution. Typically, port setups using the full-wave 2D solution in this manner are called "wave ports". After applying boundary conditions to the remaining outer surfaces, the full 3D problem is solved. Post-processing generates the S-parameters and optionally outputs files for viewing the fields.

OpenParEM3D is a command-line tool that does just the electromagnetic calculations. To complete a project, other tools must be used to create and mesh a 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 OpenParEM3D is documented here.

This document covers how to use OpenParEM3D. Details on the theory, methodology, and accuracy of OpenParEM3D are covered in a separate document "OpenParEM3D_Theory_Methodology_Accuracy.pdf".

2 Features

OpenParEM3D features are listed in Table 1, while anticipated future upgrades are listed in Table 2.

Table 1: OpenParEM3D List of Features

- Arbitrary ports with 2D transmission lines and waveguides
- Arbitrary 3D spaces driven by 2D multi-mode ports
- Full-wave frequency-domain solution of Maxwell's Equations
- Arbitrary order finite elements
- Parallel processing using the Message Passing Interface (MPI)
- Adaptive mesh refinement
- Impedance and radiation boundary conditions
- Isotropic materials with variable permittivity

Table 2: OpenParEM List of Anticipated Future Upgrades

- Antenna parameter calculations and radiation pattern plots
- Second-order radiation boundary condition
- Perfectly matched layer (PML) absorbing boundary condition
- Anisotropic materials
- Lumped ports
- One or more graphical user interfaces (GUI)
- Speed up mesh error calculation
- Microsoft Windows® port

3 Files

OpenParEM3D 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

```
$ OpenParEM3D project_name.proj
```

and in parallel it is

```
$ mpirun -q --oversubscribe -np N OpenParEM3D project_name.proj
```

where `N` is the number of cores to use for the calculation. 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 included files in a project control file are materials, mesh, and port definitions. Each of the required files are covered in detail in the following sections.

To view a very short help message and the version number, execute

```
$ OpenParEM3D -h
```

3.1 Project Control File

All aspects of the execution of OpenParEM3D 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.

```
#OpenParEM3Dproject 1.0
project.save.fields           false
mesh.file                     straight.msh
mesh.order                    6
port.definition.file          straight_ports.txt
materials.global.path         ../../../../OpenParEM2D/regression/
materials.global.name         global_materials.txt
materials.local.path          ./
materials.local.name          //local_materials.txt
refinement.frequency          none
frequency.plan.point          9e9
reference.impedance           0
```

In this example setup, the mesh file "straight.msh" is generated using gmsh [1], the "straight_ports.txt" file is generated using the script "OpenParEM3D_save.py" in FreeCAD [2], 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 [3] [project.save.fields true] or to turn on adaptive mesh refinement [ex. refinement.frequency high].

The names of the files generated by OpenParEM3D are based on the name of the project control file with the extension removed. So a control file called "my_project.proj" for a 2-port simulation results in new files being created called "my_project.s2p", "my_project_results.csv", "my_project_iterations.txt", 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.

3.1.1 Frequency Plan

Simulation frequencies are determined by the frequency plan, which is assembled using one or more of the keywords `frequency.plan.log`, `frequency.plan.linear`, and `frequency.plan.point`. The plan keywords can be listed any number of times to build up a set of frequencies to simulate. Overlaps are allowed, and duplicate frequencies are automatically eliminated. An example frequency plan with no refinement using

```
refinement.frequency  none
frequency.plan.linear 1e9,10e9,1e9
frequency.plan.linear 1e9,2e9,0.5e9
frequency.plan.log    0.1e9,1e9,3
frequency.plan.point  8.5e9
frequency.plan.point  9.5e9
```

results in the simulation being run at the frequencies

```
----------------------------------------
 frequency    refinement priority restart
----------------------------------------
      1e+08
2.15443e+08
4.64159e+08
      1e+09
    1.5e+09
      2e+09
      3e+09
      4e+09
      5e+09
      6e+09
      7e+09
      8e+09
    8.5e+09
      9e+09
    9.5e+09
      1e+10
----------------------------------------
```

Adaptive mesh refinement (AMR) is disabled by setting

```
refinement.frequency none
```

AMR is enabled by picking one of the keyword `refinement.frequency` options from the project control file specification in Appendix A. For all options except `plan`, the mesh is optimized at one or more of the frequencies in the frequency plan. Setting `refinement.frequency` to `highlow` results in the following frequency plan

```
----------------------------------------
 frequency    refinement priority restart
----------------------------------------
      1e+08   refine        2
2.15443e+08
4.64159e+08
      1e+09
    1.5e+09
      2e+09
```

```
      3e+09
      4e+09
      5e+09
      6e+09
      7e+09
      8e+09
    8.5e+09
      9e+09
    9.5e+09
      1e+10   refine        1     restart
----------------------------------------
```

The refinement frequencies are the highest and lowest in the frequency plan, with refinement first at 1e10 using the initial mesh then continuing with additional refinement at 1e08, as shown by the priority column.

The option `plan` enables fine control over the optimization frequencies by enabling refinement frequencies to be specified as the frequency plan is constructed using variations of the keywords `frequency.plan.log`, `frequency.plan.linear`, and `frequency.plan.point` with `.refine` tacked on. So in addition to the keywords `frequency.plan.log`, `frequency.plan.linear`, and `frequency.plan.point` there are also the keywords `frequency.plan.log.refine`, `frequency.plan.linear.refine`, and `frequency.plan.point.refine`. The behavior between the two versions is identical except that the `.refine` version marks those frequencies for refinement. Priority of refinement is set by the order of the `*.refine` keywords from top to bottom in the project control file. Modifying the the example frequency plan to

```
refinement.frequency        plan
frequency.plan.point.refine 8.5e9
frequency.plan.linear       1e9,10e9,1e9
frequency.plan.linear       1e9,2e9,0.5e9
frequency.plan.log.refine   0.1e9,1e9,3
frequency.plan.point        9.5e9
```

results in the frequency plan

```
----------------------------------------
 frequency    refinement priority restart
----------------------------------------
      1e+08   refine        2
2.15443e+08   refine        3
4.64159e+08   refine        4
      1e+09   refine        5
    1.5e+09
      2e+09
      3e+09
      4e+09
      5e+09
      6e+09
      7e+09
      8e+09
    8.5e+09   refine        1     restart
      9e+09
    9.5e+09
      1e+10
----------------------------------------
```

where refinement starts at 8.5e9 with the initial mesh, then continues at lower frequencies.

To check the frequency plan, include

```
debug.show.frequency.plan      true
```

in the project control file to print a frequency plan table as shown above.

3.2 Materials Files

Materials are specified in one or more materials files following the specification in Appendix B. At least one materials file is required to run OpenParEM3D, using either `materials.global.path` plus `materials.global.name` or `materials.local.path` plus `materials.local.name`. Typical simulation environments will have the bulk of the materials specified in a global file for general use across all projects, then a local materials file for materials specialized for the current project.

Note that material properties and information contained in copyrighted simulators are covered by the copyright of the simulator. It is a violation of the copyright to copy material parameters from these sources. Valid sources for material parameters are datasheets, books, and papers with appropriate citations in the Source/EndSource block.

A simple materials file consisting of just air is

```
#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
```

A more complicated entry for copper with surface roughness is

```
Material
   name=copper_prepreg
   Temperature
      temperature=any
      Frequency
         frequency=any
         er=1
         mur=1
         conductivity=5.813e7
         Rz=4.445e-6
      EndFrequency
   EndTemperature
   Source
      copper conductivity - use IPC spec at 20 degC - conductivity not provided
      in the paper Simonovich, A Practical Method to Model Effective
      Permittivity and Phase Delay Due to Conductor Surface Roughness
   EndSource
EndMaterial
```

3.3 Mesh File

The supported mesh file formats are gmsh v.2.2 and the native format of MFEM [4][5]. 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 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.

The MFEM mesh format is used by OpenParEM3D to transfer 2D port meshes to OpenParEM2D for port solution. MFEM meshes can be visualized using glvis [6].

3.4 Port Definition File

Ports and boundaries are specified in a text file following the specification in Appendix C. The project control file specifies the port definition file using the `port.definition.file` keyword. Due to complexity, the port definition file is very difficult to write by hand, so scripted generation is generally required. When using FreeCAD, the Python script "OpenParEM3D_save.py" can be used to generate the port definition file.

Port names are general text, such as "in" or "out", and they can also be numbers. Within each port are one or more modes that become the S-parameter ports in the computed S-parameter Touchstone® [7] file, and the S-parameter ports are numbered as integers starting with 1. Consider the example of a differential pair with an input port named "in" and an output port named "out". Each port has two modes: even and odd. So in total, there are two ports ("in" and "out") and four S-parameter ports ("in" even mode, "in" odd mode, "out" even mode, and "out" odd mode) leading to a 4-port S-parameter Touchstone file (with extension s4p). The port definition file has the generality to define any number of ports with any number or S-parameter ports ("Sport" in the specification).

When solving for multiple modes within a port, the modes are assigned to the S-parameter ports (again, "Sport"), in the order found with the 2D solution using OpenParEM2D. The ordering is dependent on setup with details provided in Sec. 5.2.5.

4 Example Projects

An automated regression suite is in place as part of the release methodology. The projects with short descriptions are listed in Appendix D. 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/straight/straight.proj`, at the command line in that project directory enter

```
$ process3D.sh straight.proj 12
```

where depending on the computer, a number smaller or larger than 12 could be appropriate for the number of cores to use. At successful completion, the file "straight_test_results.csv" can be viewed for the pass/fail report. Additional files are produced including the S-parameters in the csv file "straight_results.csv". Since this example does not renormalize the S-parameters, a Touchstone file is not produced. See Sec. 6 for more details. 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 that needed for engineering work to enable tight pass/fail criteria. In many situations, looser settings are appropriate.

In addition, detailed studies developed to demonstrate accuracy also represent good examples. These projects with short descriptions are listed in Appendix E. To run one of these projects, the instructions in Sec. 3 apply.

5 Flow

The process of setting up and running a project from scratch involves several steps: draw the geometry via CAD or script, 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 OpenParEM3D to obtain the S-parameters. Except for the OpenParEM3D step, any number of tools and techniques can be used to generate the needed files. One flow is described in detail here.

5.1 Geometry Definition Using FreeCAD

FreeCAD is an effective tool for building complex 3D 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 messing drawing experience: on the tool bar, 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. Successful use of FreeCAD critically depends on setting and changing the drawing plane as needed. With the drawing plane set to `Top`, primitives are drawn in the x-y plane. Changing the drawing plane is generally done by clicking on a surface in the 3D model to highlight it, then re-click the drawing plane button to reset the drawing plane to the highlighted plane, or select `Utilities`→`Select Plane`. Changes to the drawing plane are very frequently needed and are critical to successfully building a 3D model.

To build a 3D model, the general procedure is to set a drawing plane, draw a primitive 2D object, then extrude the object to create a 3D object in the Part Workbench (`View`→`Workbench`→`Part`). So a rectangle becomes a 3D box after extrusion. Once the 2D primitive is extruded, it becomes a child object of the extrusion. This is a critical point for successfully using FreeCAD. If the cross-section of the 3D extruded object needs to be changed, then edit the child object via its properties. *Do not re-draw anything.* Similarly, if two extruded objects are merged with a Boolean union operation, the two become child objects of the new merged object, so edits to a child extruded object bubbles up to the parent object. The process applies to any number of levels, with changes bubbling up to all higher levels. Redrawing objects to make changes is not an effective use of the way FreeCAD is intended to operate.

Deleting an object with children does not delete the children. So suppose a Boolean cut is made with two objects, the cut object is shown in the `Combo View` window with the two original objects shown as children. If the cut is incorrect, the cut object can be deleted, then the two original child objects are retained in the `Combo View` ready for further use.

FreeCAD has both a standard copy operation plus a clone operation. To copy an object, use the typical `Edit`→`Copy` and `Edit`→`Paste` sequence. 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 3D model.

FreeCAD has a number of workbenches tailored for specific sets of operations. To build a 3D model using the tips described here, two workbenches are used. First, the `Draft` Workbench is used to draw 2D primitives and to clone 2D and 3D objects. Second, the `Part` Workbench is used to build 3D objects by extruding 2D primitives and by applying Boolean operations to 3D objects, such as union, cut, and intersect. In the process of building a 3D model, it is necessary to frequently switch back-and-forth between the two workbenches.

When the 3D 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 ...` then saved using the BREP format, which can be imported by gmsh for material assignment and meshing.

If the 3D 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 3D model for OpenParEM3D.

5.2 Port and Boundary Annotation Using FreeCAD

The required ports definition file can be built by hand, but that is not generally practical. The FreeCAD Python script "OpenParEM3D_save.py" can be used to generate the needed ports file once the 3D model has been annotated with the needed port and boundary information. The annotations are added as physical objects, such as lines, rectangles, and polylines, along with text objects providing instructions to "OpenParEM3D_save.py".

5.2.1 Paths (_P)

Paths define physical drawing elements for use as port outlines and 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. Note that paths are defined on the outer surface of the 3D space, defined as a surface between a physical volume and a void where nothing is drawn.

5.2.2 Ports (_S, _M, and _L)

Ports are defined using a path and an impedance selection, and the information is added to the drawing by placing a text object with any text, which is ignored. A simple period works well. The port information is entered by editing the `Label` property in the property box to

```
_Sportname(PV|PI|VI){[+|-]pathname1,+pathname2,-pathname3,...}
```

where portname is any alphanumeric text, the characteristic impedance is selected from one of `PV`, `PI`, or VI, and the pathnames define the closed outline of the port. The pathnames string together to form a closed loop on a 2D plane, where + and − signs allow reversing direction, if needed. The characteristic impedance definitions are `PV` indicating the power/voltage definition, `PI` indicating the power/current definition, and VI indicating the voltage/current definition. Ports must use paths as defined in Sec. 5.2.1 on the outer surface of the 3D space.

Each of the ports are solved in OpenParEM2D for the dominant and higher-order modes, if called for. Only when every port is solved with just the dominant mode does the number of ports equal the number of S-parameter ports. For example, a straight section of microstrip has two ports, and with one propagating mode per port, there are just two S-parameter ports. On the other hand, a straight section of coupled microstrip also has two ports, but with an even and odd propagating mode per port, there are *four* S-parameter ports.

The modes at each port must be specified, and the modes become the final S-parameter ports. To define a mode at a port, place a text object with any text, and edit the `Label` property in the property box to

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

or

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

where the two forms are identical except for the prefix _*M* or _*L*. The two forms are interchangeable when there is one mode per port except that Touchstone files are not produced when using the _*M* form, so the _

L form is preferred in this case. For multimode ports, such as coupled lines, additional considerations are covered in Sec. 5.2.5.

The `portname` must align with an `_S` annotation, while `Sport` is the actual S-parameter port number starting with 1 and increasing sequentially without repeats, `voltage|current` selects whether the path defined by the path list should be used to calculate voltage or current, `net` is an optional net name, and `scale` is an optional scale factor with a default of 1. For a given port and Sport number, one definition for voltage plus one definition for current are allowed. As an example, for a straight section of microstrip, the annotations could look like

```
_Sin(PV){portin}
_Sout(PV){portout}
_Min(1,voltage){vlinein}
_Min(1,current){iloopin}
_Mout(2,voltage){vlineout}
_Mout(2,current){iloopout}
```

5.2.3 Boundaries (_B)

Boundaries are defined on the outer surface of the 3D space, as defined in Sec. 5.2.1, 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|radiation,material|[wave_impedance]){[+|-]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, `PMC` for perfect magnetic conductor, and `radiation` for radiation. Selecting `SI` requires the conductor material to be specified in the second argument [e.g. copper], while selecting `radiation` uses the impedance of free space unless the `wave_impedance` is specified. The paths must form a closed loop on a 2D plane.

A perfect electric conductor (PEC) boundary forces the electric field tangential to the boundary to be exactly zero. Unless overridden by `materials.default.boundary`, this is the default boundary condition for the outer surfaces of the 3D space wherever other port or boundary assignments are not made. 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 perfect magnetic conductor (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.

A surface impedance (SI) boundary assumes that the material is thick compared to the skin depth at the simulation frequency. No penetration through the boundary is modeled. The keyword pair
`materials.default.boundary some_metal` re-assigns the default boundary condition from PEC to a surface impedance using the conductor `some_metal`.

A radiation boundary implements a 1^{st}-order absorbing boundary condition for antenna input impedance simulations. Radiation boundaries need to be in the far field, so at least 1 or 2 wavelengths away from the radiating structure.

5.2.4 Differential Pairs (_D)

Single-ended S-parameters calculated with `impedance_calculation=line` in the port definition file can be converted to mixed-mode S-parameters by specifying differential pairs, where each differential pair is defined by placing a text object with any text, which is ignored, with the `Label` property in the property box edited to

```
_D(Sport_P,Sport_N)
```

where `Sport_P` is the Sport number for the positive half of the differential pair and `Sport_N` is the Sport number for the negative half.

5.2.5 Multimode Ports

Multimode ports often arise in the context of printed circuit boards, where a port captures more than one transmission line. Differential pairs are an especially common multimode port. OpenParEM2D and OpenParEM3D always process the fields at ports as modes, both dominant and higher-order modes. When calculating voltages and currents, the calculations are always done on the modal fields.

The voltage and current paths can be set up in one of two ways. In the first method, called the *modal setup*, the paths can be set up to directly calculate the modal voltages and currents. In this setup, the user must know the correct paths for the modal fields plus any scale factors to apply. For symmetric differential pairs, these are known, but otherwise, the setup is challenging. Consider the modal setup to be an expert mode applicable in some cases for some users. S-parameters produced using the modal setup cannot generally be directly used in circuit simulations because the simulators expect S-parameters referencing single lines, not modes.

In the second method, called the *line setup*, the paths are set up to calculate the voltages and currents per transmission line. This is the more intuitive setup for most users, but the resulting voltages and currents are neither the modal voltages and currents [because the paths are wrong] nor the isolated transmission line voltages and currents [because the fields are wrong, being modal fields]. Post-processing is applied to derive the correct isolated transmission line voltages and currents to obtain S-parameters that can be used in circuit simulators, and the algorithm is discussed in the document "OpenParEM3D_Theory_Methology_Accuracy.pdf".

Referring back to Sec. 5.2.2, there are two ways to define voltages and currents: `_M` and `_L`.
Using `_M` produces a port definition file [Sec. 3.4] marked for use using the modal definitions with `impedance_calculation=modal` when calculating S-parameters, while `_L` produces a port definition file marked for line definitions with `impedance_calculation=line` so that the appropriate processing can be applied to the calculated voltages and currents. When there is just one mode per port, there is no difference in calculations, so either `_M` or `_L` annotations can be used. As noted in Sec. 5.2.2, with a modal setup, no Touchstone is produced.

As an example, consider a differential pair of microstrip transmission lines. Paths for the port, currents, and voltages are shown in Fig. 1, where `_P1` and `_P2` are for currents [drawn as rectangles in FreeCAD] and `_P3` and `_P4` are for voltages [drawn as lines in FreeCAD]. The setup for the *line* setup for port "in" looks like

```
_Sin(PV){port}
_Lin(1,voltage){3}
_Lin(1,current){1}
_Lin(2,voltage){4}
_Lin(2,current){2}
```

and the resulting S-parameters are single-ended for the individual lines. For the *modal* setup,

```
_Sin(PV}{port}
_Min(1,voltage,comm_in,0.5){3,4}
_Min(1,current,comm_in){1,2}
_Min(2,voltage,diff_in){3,-4}
_Min(2,current,diff_in,0.5){1,-2}
```

and the resulting S-parameters are the even and odd modes.

The single-ended S-parameters can be converted to mixed-mode S-parameters using `Differential Pair` definitions. In this case, the *line* setup looks like

```
_Sin(PV}{port}
_Lin(1,voltage){3}
_Lin(1,current){1}
_Lin(2,voltage){4}
_Lin(2,current){2}
_D(1,2)
```

where the results for S-parameters ports (Sports) 1 and 2 are processed to modal S-parameters for even and odd modes. In this symmetric layout, after mixed-mode conversion, the S-parameters are directly comparable to those produced by the modal setup. Note that mixed-mode calculations cannot be applied to *modal* setups.

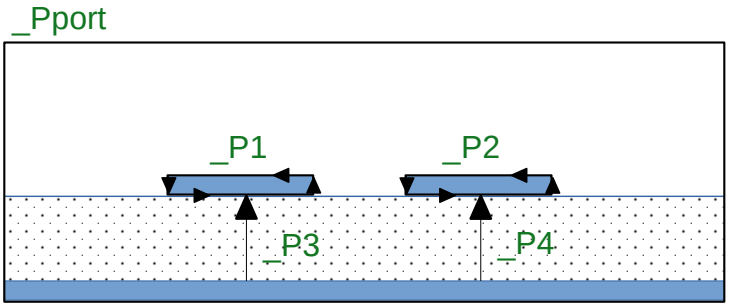


Figure 1: Path Definitions for Microstrip Differential Pair

5.3 Meshing Using gmsh

For meshing, gmsh is an effective tool that has a relatively straightforward interface. The library used by OpenParEM3D 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 volumes in the imported model. In the cascading options, select ⊞`Geometry`→⊞`Physical groups`→⊞`Add`→`Volume`. A popup menu appears along with yellow dots on the model indicating the volumes. Add text to the popup for the material for a specific volume, then click a yellow dot (or more, if all have the same material) for that volume, 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 volumes 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, so if the mouse becomes disabled, click that red box.

With the materials defined, in the cascading options select ⊞`Mesh`→`3D` 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 volume assignments. Meshing can then immediately proceed.

At the GUI level, meshing 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 3D algorithm, where `Delaunay` and `Frontal` are the principle options, 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`→`3D` to generate a new mesh.

Generally, a sparse mesh is desirable for speed, but sparser meshes have lower quality, which impact 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.

5.3.1 Plotting Attributes

The assignments of attributes to the mesh can be visually checked using ParaView using a csv file called `projectName_attributes.csv` that is generated as a project completes. To view the attributes, execute the

following steps.

- Edit the project control file to add the keyword/value pair `debug.show.port.definitions true`, then run the OpenParEM3D project.
- Start ParaView.
- Open the attributes file `projectName_attributes.csv`. When the pop-up window appears, choose `CSV Reader` then `Ok`.
- Add a filter to process the data with `Filters`→`Alphabetical`→`Table To Points`.
- In the Properties window, for `X Column` select `X`, for `Y Column` select `Y`, and `Z Column` select `Z`.
- Close the `Spreadsheet1` view by clicking the ☒.
- Select the `TableToPoints1` filter. A white dot matrix of the geometry will appear.
- Change the `Coloring` from `Solid Color` to `Attribute`.
- Change the `Representation` from `Surface` to `Points Gaussian`.
- Adjust the point size with `Gaussian Radius` to improve readability.
- The points show the attributes assigned to the boundary mesh elements by color coding.
- Review the OpenParEM3D output to find the attributes assigned to the boundaries. These can be checked against the ParaView plot to ensure that the assignments occurred as expected.

5.4 Solution With OpenParEM3D

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

```
$ mpirun -q --oversubscribe -np N OpenParEM3D 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 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 OpenParEM3D 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 OpenParEM3D to check and report when `N` is too large. If OpenParEM3D hangs at the step `building finite element spaces ...`, then the remedy is to kill the job and restart with a smaller `N`.

It is likely on initial attempts to run a new setup with OpenParEM3D that errors will be reported and the simulation stopped. The design of OpenParEM3D is to make no assumptions or corrections to input decks, so every error is reported even if it could be possible for OpenParEM3D to fix or adjust the setup. In OpenParEM3D v1.0, there are over 600 individual checks with customized error messages. Correct the errors and try again.

OpenParEM3D uses a lock file when running to prevent data collisions from accidentally running the same project control 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 OpenParEM3D 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, OpenParEM3D will exit with cryptic messages from the MPI system without mentioning the memory issue and without removing the lock file. This typically occurs at the stage of `solving E field in 3D volume ...` or `solving H field in 3D volume ...`, so an unexplained exit at this stage is most like caused by running out of memory.

When a job is running, progress is shown by extensive data sent to the terminal. In addition, there are two continuously updated files that show progress: `project_name_results.csv` and `project_name_iterations.csv`. They can be view as scrolling log files by opening two new terminals, then after the files appear, run

```
$ tail -f project_name_results.csv
```

in one terminal and

```
$ tail -f project_name_iterations.txt
```

in the other. The three terminals together enable a comprehensive view of execution progress.

5.5 Viewing Fields

To view the computed electromagnetic fields, OpenParEM3D outputs files for viewing with ParaView after every iteration and every frequency. To enable this output, set

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

in the project control file.

Since the files are produced while OpenParEM3D is executing, ParaView can be used to view intermediate results. The one constraint is that ParaView will report errors if OpenParEM3D updates the files while ParaView is still reading them. If that happens, just try again in ParaView.

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 change. This setting can be found at `Edit`→`Settings ...`→`General`→`Auto Apply`.

When a ParaView file is first opened, 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. The displayed field can be further refined by changing from plotting `Magnitude` to one of the field components.

ParaView uses filters to modify how the field is shown. A very useful one is the "Slice" filter. To enable the slice filter, select `Filter`→`Alphabetical`→`Slice`. The slice can be modified by its controls to explore the fields in the interior of the 3D model.

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.

6 S-parameter Output Files

Computed S-parameters are always output in a custom csv file with the name `project_results.csv`. In addition to this file, S-parameters *may* be output in either Touchstone 1.1 format or Touchstone 2.0 format.

If the S-parameters have not been renormalized to a reference impedance using the keyword `reference.impedance`, then a Touchstone output file is not generated. Port impedances are in general frequency dependent, but the Touchstone file formats do not support frequency-dependent reference impedances. To satisfy the formats, then renormalization must be done through providing a non-zero value for `reference.impedance`.

Note that Touchstone outputs are verified for a number of test cases against the Touchstone golden parser `tschk2` from the IBIS Open Forum [7]. The OpenParEM3D developers do not have access to circuit

simulation software to perform additional testing. Users should be especially careful when using Touchstone outputs until sufficient confidence has accumulated.

6.1 Modal Setups

Modal S-parameters are not output as a Touchstone file because the 1.1 format is for single-ended S-parameters and the 2.0 format is for single-ended or mixed-mode S-parameters. Modal S-parameters may or may not be mixed-mode, plus there is the issue that the 2.0 format requires identifying common and differential modes, which OpenParEM3D cannot do on its own, so it would be possible for these to be reversed.

6.2 Line Setups

For unnormalized S-parameters computed with a line setup, the S-parameters are still modal and are treated in the same way as S-parameters computed with a modal setup. In short, a Touchstone file is not output.

If a reference impedance is specified using the keyword `reference.impedance`, then the modal S-parameters are converted to single-ended S-parameters and renormalized in the process. The results are output as a Touchstone 1.1 file.

6.2.1 Mixed-Mode S-parameters

Renormalized S-parameters using line setups can optionally be converted to mixed-mode S-parameters using `DifferentialPair/EndDifferentialPair` blocks. Mixed-mode S-parameters are always output as Touchstone 2.0 file. It is up to the user to ensure that the modes are actually symmetric so that they are true common and differential modes.

7 Tutorials

7.1 Rectangular Waveguide

A straight section of WR90 rectangular is constructed, annotated for 2-port S-parameter simulation, 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.

7.1.1 Drawing and Port 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. 5.1.
- Click `Drafting`→`Rectangle` or click on the rectangle icon on the tool bar 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 5.1 discusses why drawings are in μm to ultimately obtain a mesh in mm. For property `Width` change the value to 22.86 um.
- Zoom to view the rectangle with `View`→`Standard Views`→`Fit All`. A reference screen shot is shown in Fig. 2

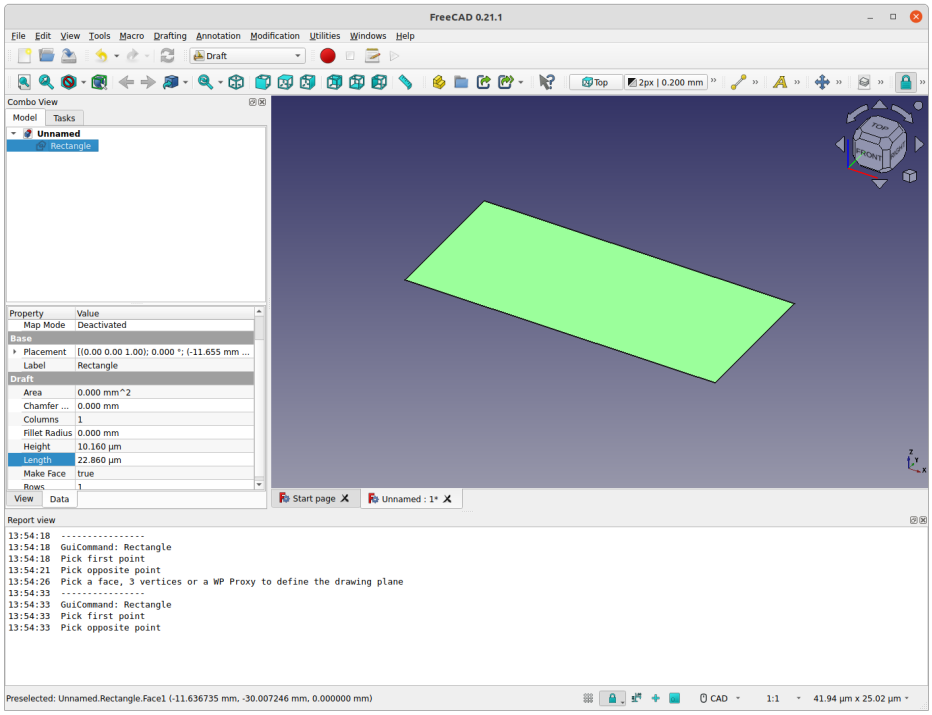


Figure 2: Screen shot of a rectangle sized for WR90 waveguide using μm for ultimate mm scaling.

- Switch to the `Part` Workbench with `View`→`Workbench`→`Part`.
- Select the rectangle then select `Part`→`Extrude ...`, then enter 100 um in the `Along` box. Click `Ok`.
- Select the item `Extrude` and view the `Properties` window to see the extrusion length in the `Length Fwd` value box. If the length of the extrusion needs to be changed, change it here by editing the value of `Length Fwd`.
- For learning purposes, change the extrusion length for the rectangle to 90 um and zoom to fit all. A reference screen shot is shown in Fig. 3.
- In the `Combo View` window, click the small sideways triangle next to the `Extrude` item to show the `Rectangle` item. Select the `Rectangle` item. In the `Property` window, change the value of `Label` to `_Pportin`. This rectangle marks the input port.
- Change to the `Draft` workspace by selecting `View`→`Workbench`→`Draft`.
- Save the drawing with the name `rectangular_waveguide` using `File`→`Save`.
- Click on the end of the long extrusion that is opposite of `_Pportin` then change the drawing plane to this plane by selecting `Utilities`→`Select Plane`. Alternatively, click the menu bar item as shown in Fig. 4 to change from `Top` to `Custom`.
- Draw a rectangle by snapping to diagonal corners of the end of the extrusion, then change the value of `Label` to `_Pportout`. This rectangle marks the output port.
- Hover the cursor over the `_Pportout` in the `Combo View` window and the rectangle will highlight as shown in Fig. 5.
- Expand the `lock` icon and select `Snap Midpoint`. The screenshot in Fig. 6 shows the selection of the snapping to the midpoint of lines.

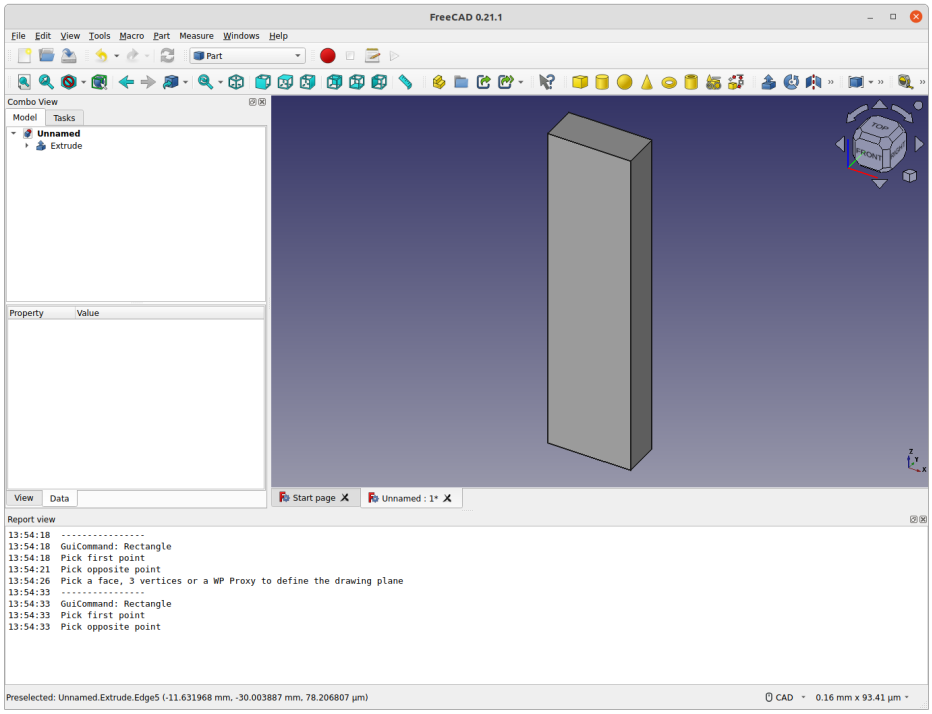


Figure 3: Screen shot of the rectangle extruded 90 um.

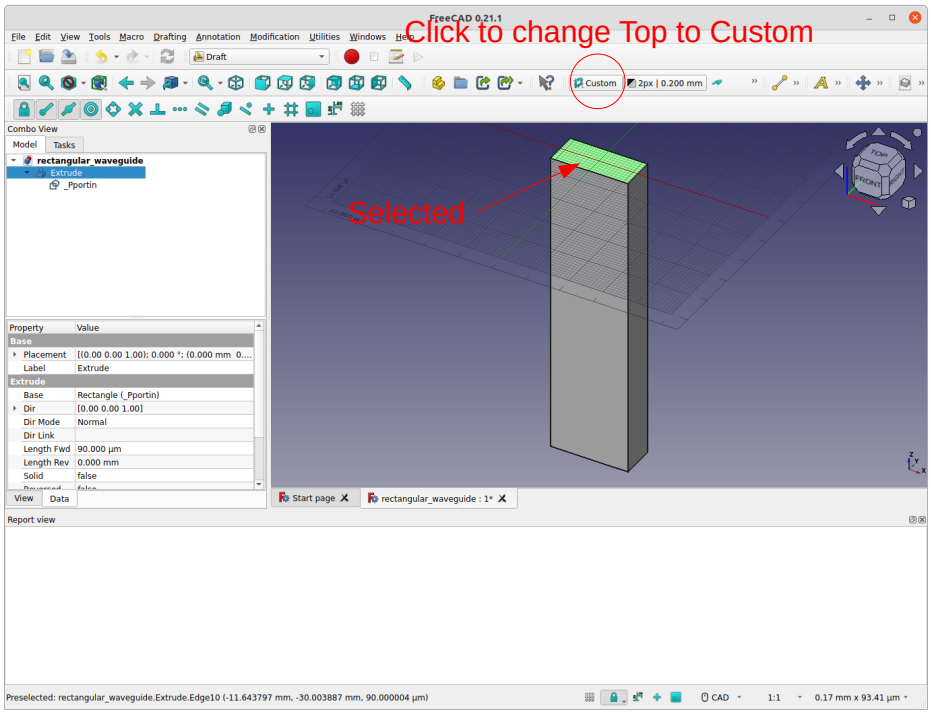


Figure 4: Setting the drawing plane to the end of the extrusion.

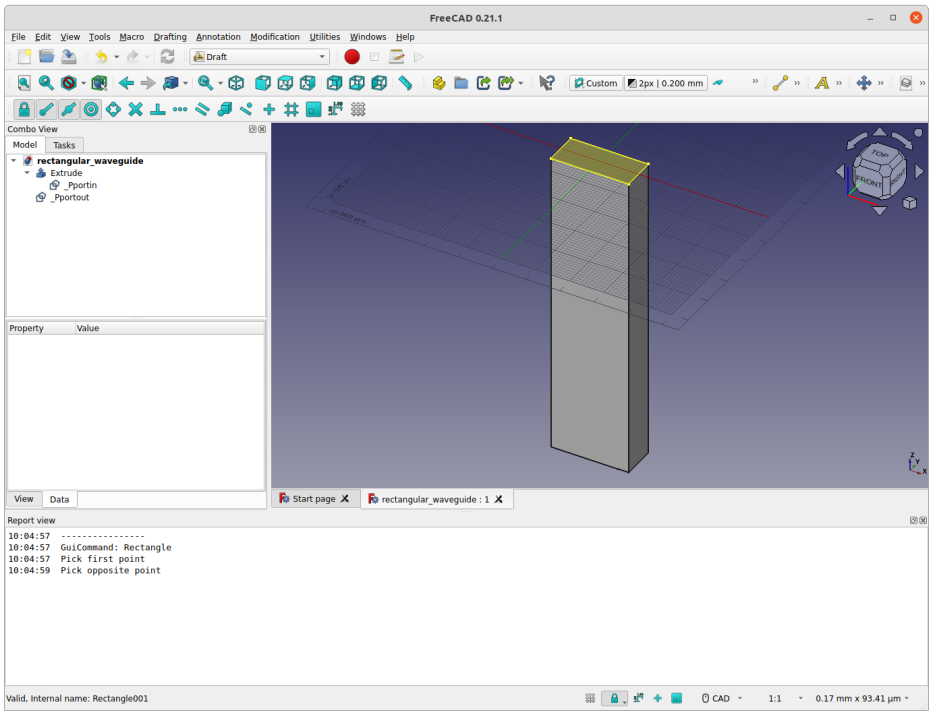


Figure 5: Hovering over the port object _
`Pportout` in the `Combo View` window highlights the output port.

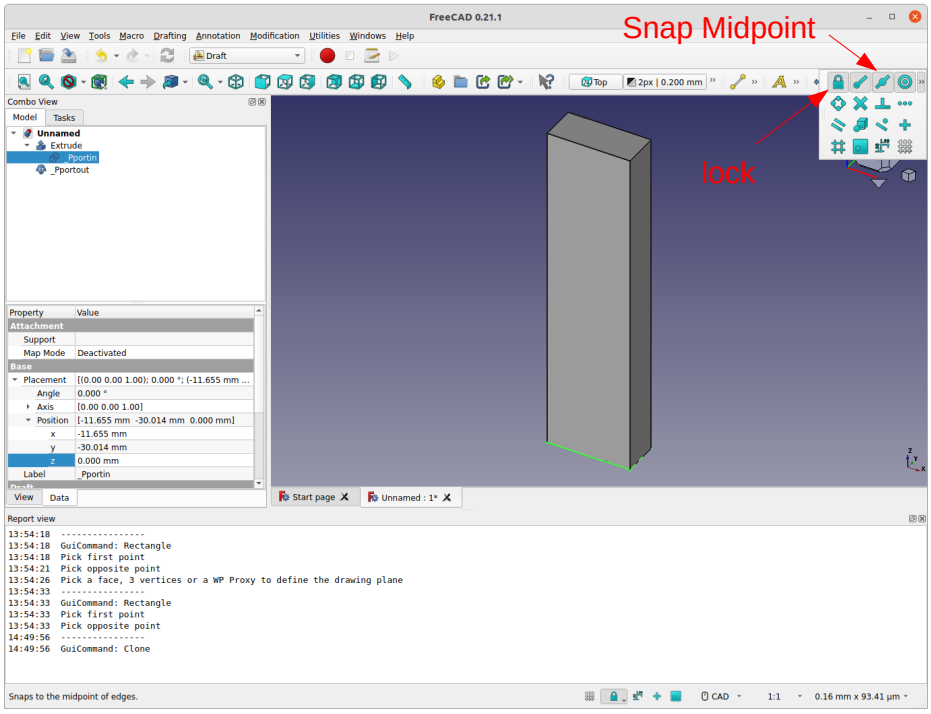


Figure 6: Snapping selection showing snapping to line ends and centers and to face centers.

- View the input port with `View→Standard views→Bottom`. Select line drawing with `Drafting→Line` and draw a line from the center of the top of the rectangle to the bottom (i.e. directed in the +y direction). Make sure it snaps to the line by only clicking when a white circle appears on the snap point. It is critical that lines snap accurately to the geometry to avoid later errors in OpenParEM3D when things do not line up. This line is the voltage integration line for the input port. Change this line's `Label` to `_Pvin`.
- View the orthogonal view by selecting `View→Standard views→Home`. Draw a line from the bottom of the output port [long side] to the top of the port [long side] (i.e. directed in the +y direction). Again, ensure that the line snaps to the middle of the side of the rectangle. This line is the voltage integration line for the output port. Change this line's `Label` to `_Pvout`.
- Add a text item to the drawing by selecting `Annotation→Text` and click anywhere in the drawing. Enter a period in the popup text box, then click `Create text`. Select the new text object in the `Combo View` window, and change its `Label` to `_Sin(PV){portin}`. This adds the input port using the power-voltage definition for characteristic impedance using the rectangle `_Pportin`.
- Repeat the adding of a text item, or copy/paste the existing one, and change its `Label` to `_Sout(PV){portout}` to add the output port.
- Add a text item and change its `Label` to `_Min(1,voltage){vin}` to add the S-parameter port 1 to port "in" using voltage calculation on the path defined by the line with label `_Pvin`.
- Add a text item and change its `Label` to `_Mout(2,voltage){vout}` to add the S-parameter port 2 to port "out" using voltage calculation on the path defined by the line with label `_Pvout`.
- Save the drawing since it is complete.
- Select the `Extrude` object in the `Combo View` window, then export it with `File→Export ...` then `Save`, removing the `-Extrude` part of the name. The default format is `BREP` but can be verified before saving by checking the format drop-down.
- Export the ports description file by selecting `Macro→Macros ...→OpenParEM3D_save.py→Execute`. Enter the name `rectangular_waveguide_ports.txt`, then select `Save`. Check the `Report view` window for errors.
- A screen shot of the completed drawing with a successfully exported ports description file is shown in Fig. 7.
- Save the drawing and exit FreeCAD.

7.1.2 Meshing

- Start gmsh as outlined in Sec. 5.3.
- Open the BREP file saved from the prior section by selecting `File→Open ...→rectangular_waveguide.brep→Ok`.
- Click and drag in the window to change the view.
- Assign the material as air by clicking the options tree `⊞Geometry→⊞Physical Groups→⊞Add→Volume`.
- In the pop-op window type `air` then select the yellow dot, 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 dot, click the red box in the lower left to re-enable mouse input.]
- To mesh the geometry, in the options tree click `⊞Mesh→3D`. A screenshot of the mesh is shown in Fig. 8.
- Save the mesh by selecting `File→Save Mesh`.
- Quit gmsh.

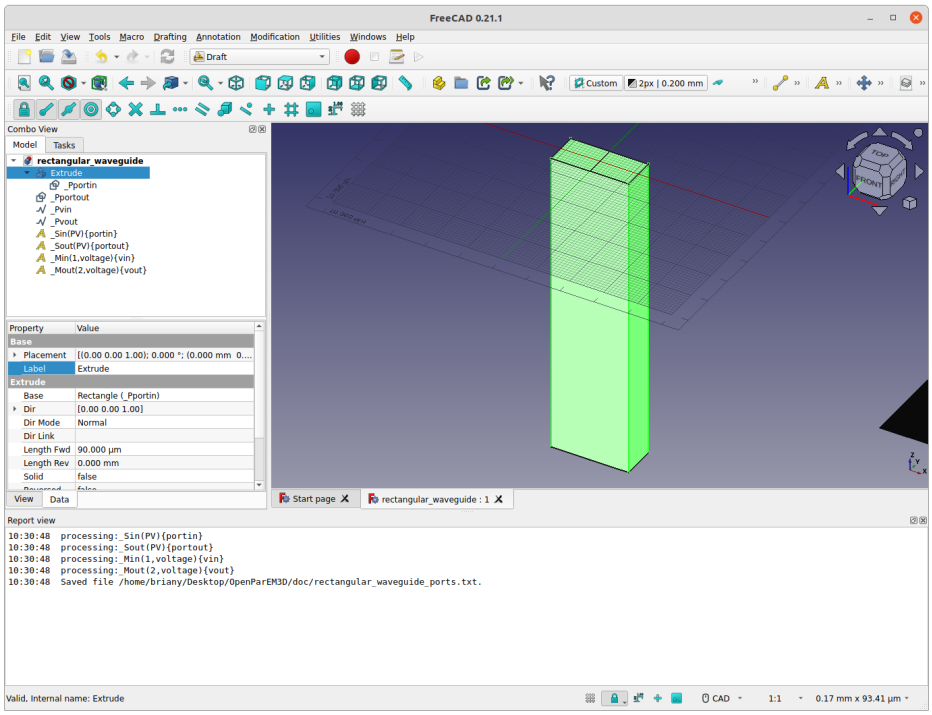


Figure 7: Screenshot of the completed drawing with exported ports description file.

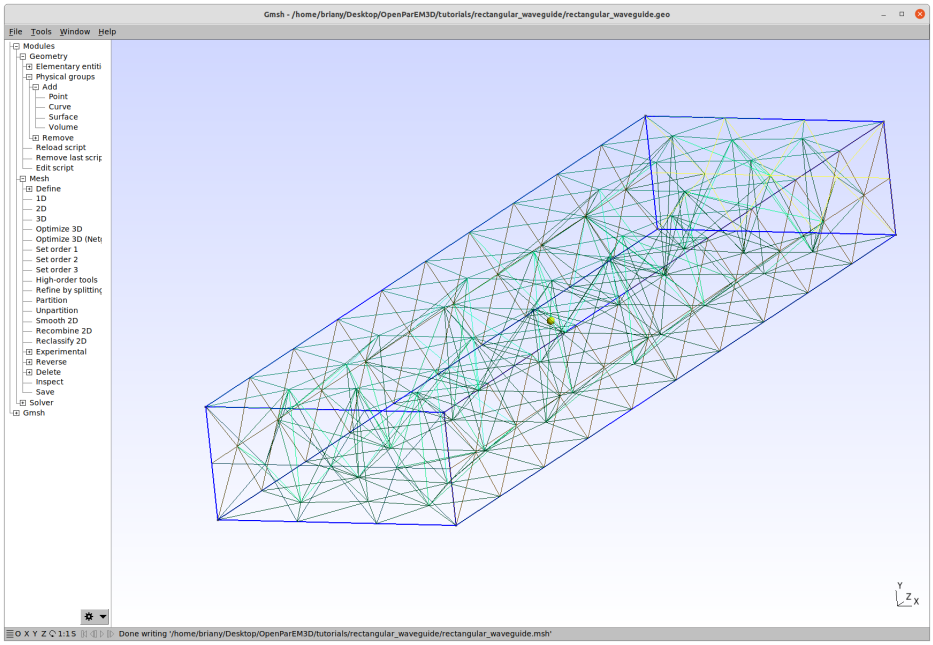


Figure 8: Screenshot of the meshed straight section of rectangular waveguide.

7.1.3 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

```
#OpenParEM3Dproject 1.0
project.save.fields            true
mesh.file                      rectangular_waveguide.msh
mesh.quality.limit             100
mesh.order                     4
port.definition.file           rectangular_waveguide_ports.txt
materials.global.path          ./
materials.global.name          global_materials.txt
materials.local.path           ./
materials.local.name           //local_materials.txt
refinement.frequency           none
refinement.iteration.min       1
refinement.iteration.max       5
refinement.required.passes     1
refinement.relative.tolerance  1e-4
refinement.variable            S
frequency.plan.point           10e9
reference.impedance            0
```

  Note that iterative refinement is not needed for this problem since the mesh is fairly dense, 4^{th}-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 OpenParEM3D at the command line with

```
$ OpenParEM3D rectangular_waveguide.proj
```

  to run serially with a single core or with

```
$ mpirun -q --oversubscribe -np 12 OpenParEM3D rectangular_waveguide.proj
```

  to run in parallel with 12 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.

- View the 2-port S-parameters, such as `$ more rectangular_waveguide.s2p`, with the results looking very similar to

```
! 2-port S-parameter data
! Sport 1 net1
! Sport 2 net2
# GHz S DB R 0
!freq dBS11 angS11 dBS21 angS21 dBS12 angS12 dBS22 angS22
10 -78.3715096672794 -55.5217603830515 0.000533264064487618 -96.4088426326485 0.000533264064487618 -96.4088426326485 -78.3907308077556 -52.6413868985043
```

  These are serial results, and running in serial should produce almost identical results. If running in parallel, due to the nature of MPI processing and iterative solution, the results can vary slightly.

  Note the very high return loss since this is just a straight section of waveguide and the very slight positive insertion loss of 0.00053 dB since there are no conductor or dielectric losses.

  From the result of the 2D simulations scrolled in the output when running OpenParEM3D, β=158.32151 rad/m. Since the waveguide is 90 mm long, the expected phase shift is $\theta = -158.32151 * 0.090 = -96.404°$, which is in very close agreement to the 3D computed value of $-96.409°$, or 0.0051%.

- Start ParaView at the command line with `$ paraview &`, then navigate to and open the fields result `rectangular_waveguide_frequency_1e+10_Sport_1.pvd`
- Click in the window and move the mouse to change the view.
- Click on the box `Solid Color` then select `gridReE`. Click `Apply` if nothing happens.
- Click on the box `Magnitude` and select `Y` to view the y-component of the electric field. A screen shot is shown in Fig. 9.

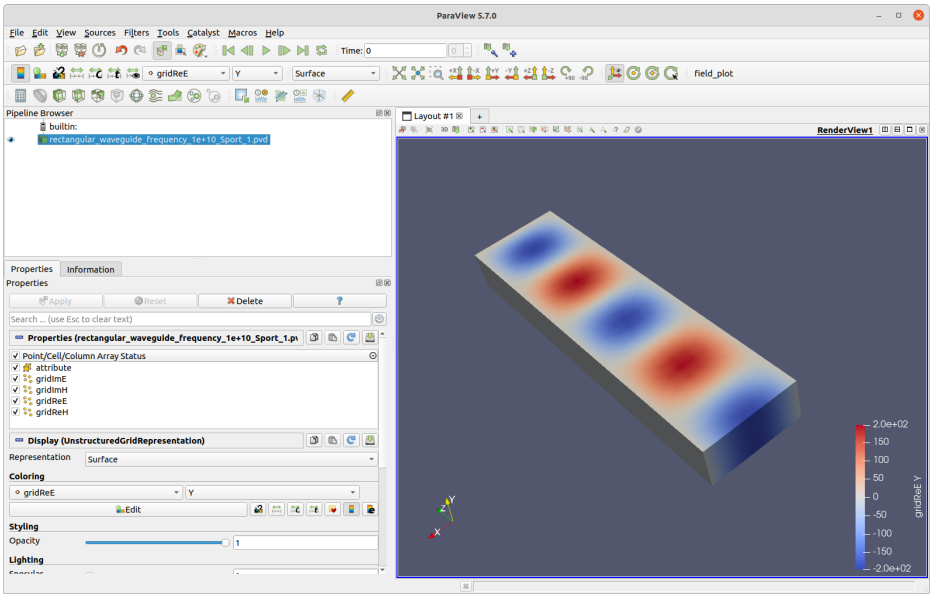
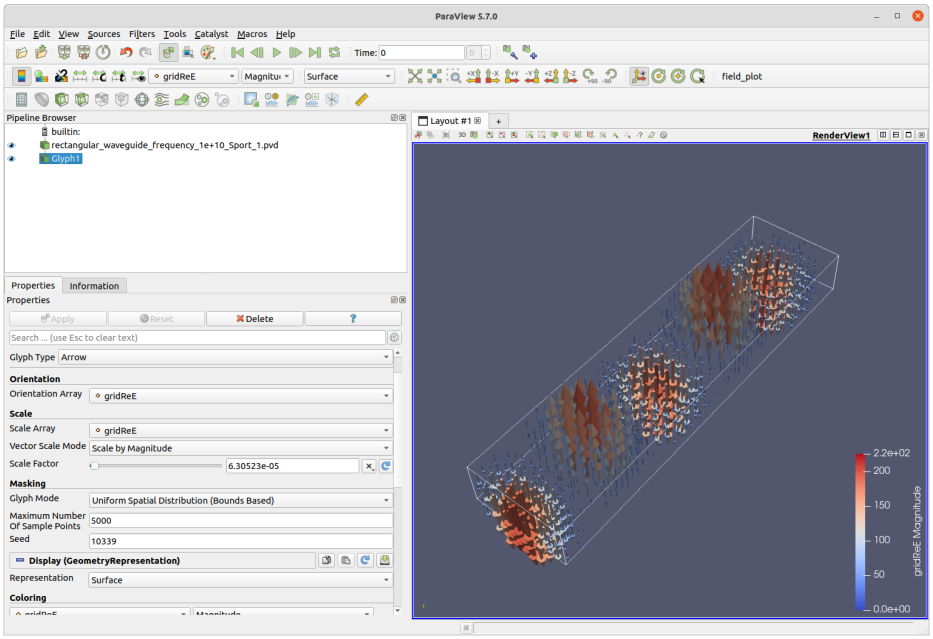


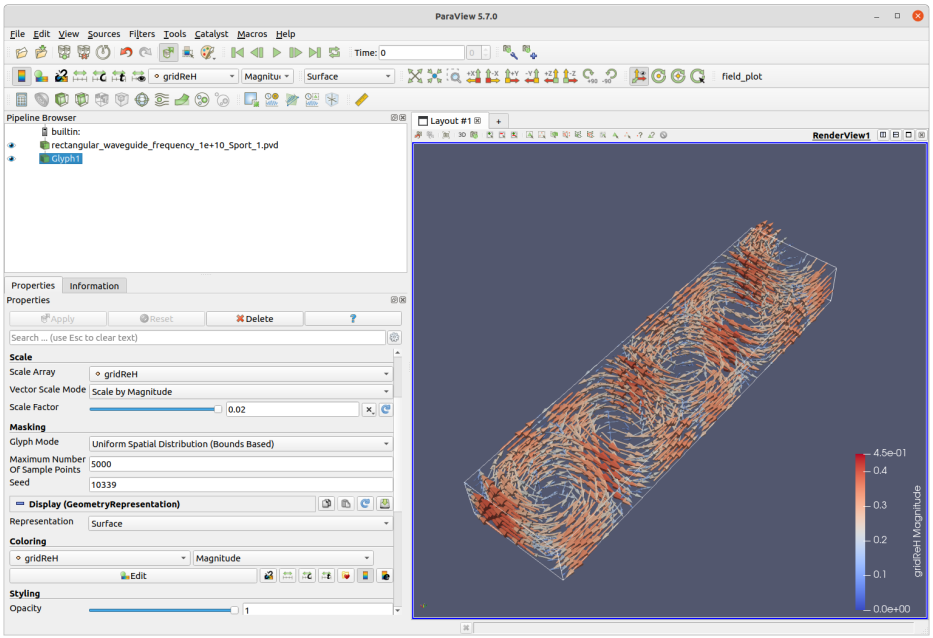
Figure 9: Re(E_y) for the rectangular waveguide when driven by port 1.

- Vector plots can be viewed by selecting `Filters`→`Alphabetical`→`Glyph`.
- Ensure that the `Glyph1` item in the `Pipeline Browser` is selected and that any other items are not.
- Set `Orientation Array` to `gridReE` and the `Scale Array` to `gridReE`.
- Set `Scale Factor` to 6.3e-05.
- An outline of the geometry can be created by enabling `rectangular_waveguide_frequency_1e+10_Sport_1.pvd`, then change `Representation` from `Surface` to `Outline`.

- A similar plot for $\overline{H}$ can also be made.
- Screenshots for $\overline{E}$ and $\overline{H}$ are shown in Fig. 10.



(a) Re($\overline{E}$)



(b) Re($\overline{H}$)

Figure 10: Vector fields for the rectangular waveguide when driven by port 1.

7.2 Microstrip Step in Width

A straight section of microstrip with a step in width is constructed, annotated for 2-port S-parameter simulation, meshed, and solved.

7.2.1 Drawing and Port Annotation

- Create a directory for the project

```
$ mkdir microstrip_step
$ cd microstrip_step
```

- 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. 5.1.

- Four rectangles are needed to eventually form the substrate, the input microstrip line, the output microstrip line, and the air above. Add four rectangles and modify the properties to get them started. Continuing the work-around with FreeCAD, the drawing is done in μm to end up with a mesh in mm.

  - Substrate: `label`→`substrate2D`; `height`→`0.254um`; `length`→`2.54um`
  - Input Microstrip: `label`→`input2D`; `height`→`0.025um`; `length`→`0.254um`
  - Output Microstrip: `label`→`output2D`; `height`→`0.025um`; `length`→`0.508um`
  - Air: `label`→`air2D`; `height`→`1.27um`; `length`→`2.54um`

- Position `substrate2D` to the origin by changing the properties `Base`→`Placement`→`Position`→`x` to 0 and similarly for `y` and `z`.

- Enable snapping to center as shown in Fig. 6.

- Select the object `air2D` then move it to snap the center of the bottom to the center of the top of the object `substrate2D` using `Modification`→`Move`. To do this, bring the mouse near the desired snap location until the while dot appears to show that the snap point is active, click and release the mouse, drag to the desired location until another white dot appears, then click to complete the move. It is critical to click for the moves only with the white dots showing to achieve proper alignment.

- Similar to `air2D`, select the object `input2D` and move/snap the bottom center to the top center of `substrate2D`. Repeat for `output2D`.

- If an object is in the way and prevents picking up the snap points of another object, hide the object by selecting it and pressing the space bar. Unhide by selecting it in the `Combo View` window and pressing the space bar.

- The project should look like the screen shot in Fig. 11.

- Save the drawing as `microstrip_step`.

- Change the workspace from `Draft` to `Part`.

- Select all four rectangles, then `Part`→`Extrude ...`→`Along` and enter `10um` to extrude each box 10 μm.

- Change the label of each extrusion to match the label of the underlying rectangle, so for example, `Extrude` is changed to `substrate`.

- Put the display in a convenient view by `View`→`Standard Views`→`Home`.

- Select the object `substrate` and hide it by pressing the space bar.

- Select `input` and change the property `Length fwd` to `5um` to make it half as long. Repeat for `output`.

- Change back to the `Draft` workspace, then select `output` and move it 5 μm so that the near end snaps to the far end of `input`. The two microstrip lines should be co-linear and end-to-end, as shown in Fig. 12.

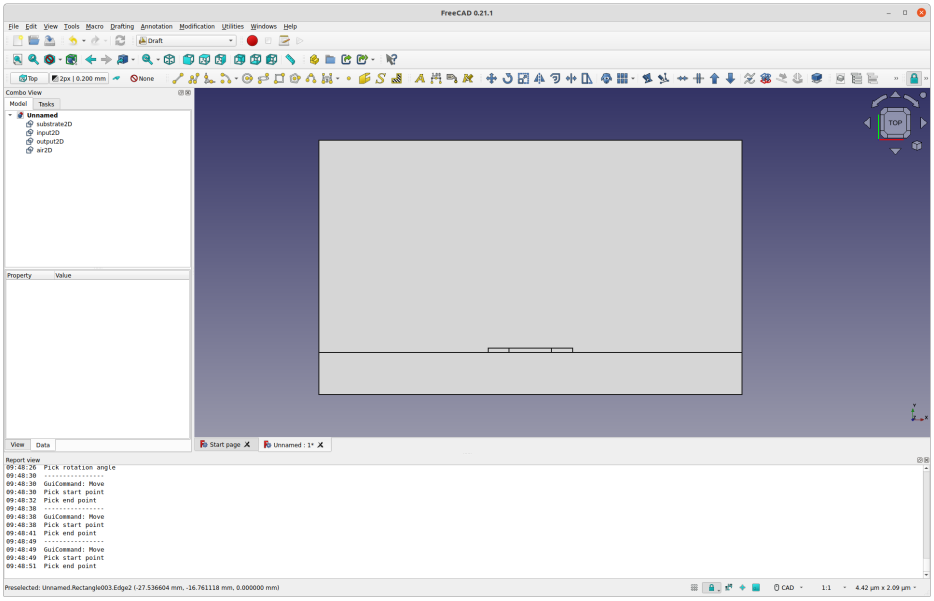


Figure 11: Screenshot of the 2D setup for the microstrip-step-in-width tutorial.

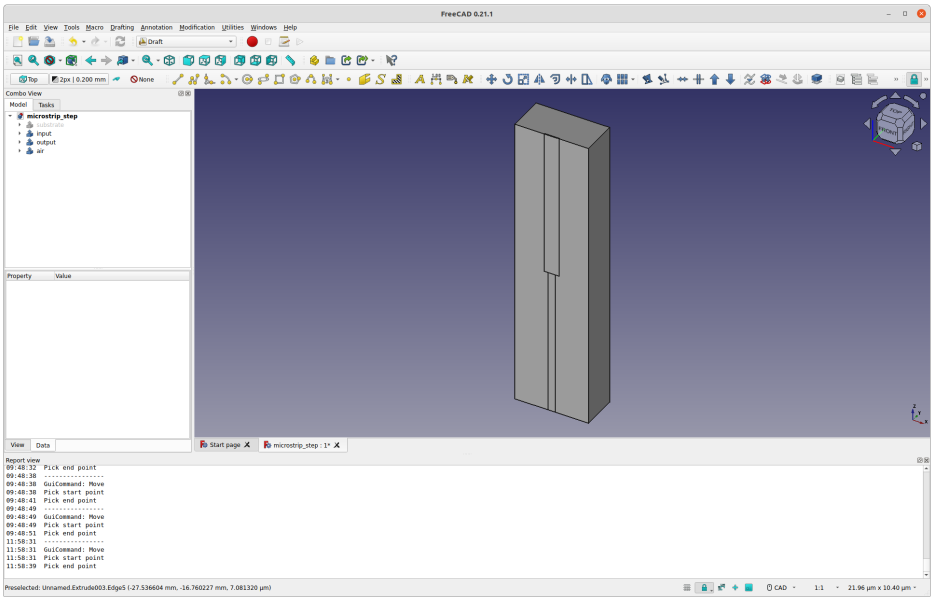


Figure 12: Screenshot of the 3D setup after extrusion and movement of one microstrip segment.

- All surfaces *outside* the drawn geometry default to perfect electric conductors, so metal. *Outside* means a void where there is no geometry defined. To create the microstrip lines, the drawn microstrips can be Boolean subtracted from the air to create the needed voids. To do this, switch back to the `Part` workspace, select `air`, then add `input` to the selection by selecting it with the `Ctrl` key pressed, then select `Part`→`Boolean`→`Cut`. Select the newly created cut item then add `output` to the selection and repeat the cut operation. Change the label of the final cut object to `air_metal`.
- Select `substrate` and press the spacebar to enable viewing it.
- Select `air_metal` and `substrate`, then merge them by selecting `Part`→`Split`→`Boolean fragments`.
- Select the Boolean fragments object then export it a BREP file with the name `microstrip_step.brep`.
- At this point, there is a 3D substrate and 3D air with a void where the microstrip metal exists. It is ready for port annotation.
- Set up the port at the top for adding annotations by setting the workspace to `Draft`, selecting `View`→`Standard views`→`Top`, clicking on one of the rectangle faces to select it, and selecting the plane as the drawing plane with `Utilities`→`Select Plane` or by clicking the related tool bar button.
- To form the output port outline, draw a rectangle covering the full cross section and change the label to `_Pportout`.
- To add the output port current integration path, draw a rectangle over the microstrip and change the label to `_PIout`.
- To add the output port voltage integration path, draw a line from the center of the bottom of the substrate to the center of the microstrip line at the interface to the substrate and change the label to `_PVout`. Ensure that snapping-to-center is on for this operation.
- For the input port, selecting `View`→`Standard views`→Bottom, select a rectangle face, and set the drawing plane. Add rectangles for `_Pportin` and `_PIin` and a line for `_PVin`. Note that the directions of the voltage lines between the two ports must be the same to avoid an extra 180° phase shift, so both need to go from the ground to the strip or from the strip to the ground. If the strip-to-ground direction is drawn, then the characteristic impedance using the V/I definition will be negative [which does not hurt anything].
- Add text objects with labels to complete the ports with labels `_Sin(PV)portin`, `_Min(1,voltage)Vin`, `_Min(1,current)Iin`, `_Sin(PV)portout`, `_Mout(2,voltage)Vout`, and `_Mout(2,current)Iout`. Since the PV definition for characteristic impedance is called out, the current mode definitions are not used. The input microstrip port is S-parameter port 1, while the output microstrip port is S-parameter port 2.
- Export the ports description file by selecting `Macro`→`Macros ...`→`OpenParEM3D_save.py`→`Execute`. Enter the name `microstrip_step_ports.txt`, then select `Save`. Check the `Report view` window for errors.
- A screen shot of the completed drawing with successfully export ports description file is shown in Fig. 13.
- Save the drawing and exit FreeCAD.

7.2.2 Meshing

- Start gmsh and open the saved BREP file.
- Assign materials by clicking the options tree ⊞`Geometry`→⊞`Physical Groups`→⊞`Add`→`Volume`.
- In the pop-op window type `air` then select the yellow dot showing the area of air. When selected, the dot turns red. Press the keyboard `e` and a new pop-up appears. Click `Create new '.geo' file`.

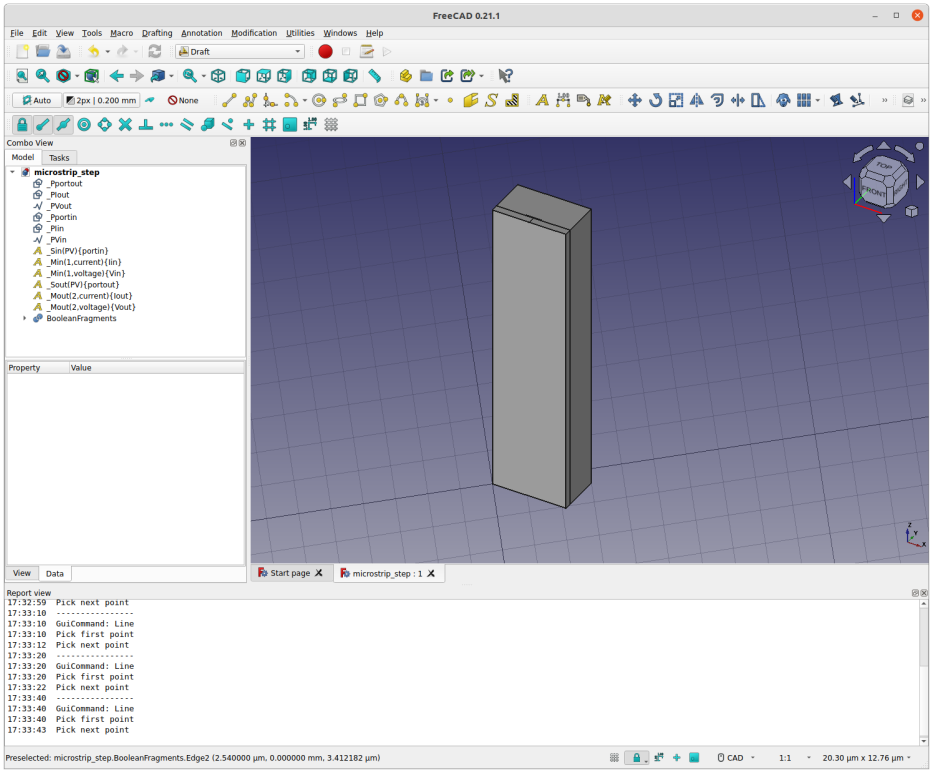


Figure 13: Screenshot of the completed 3D setup.

- Change the text from `air` to `alumina` then select the yellow dot showing the substrate. Press the keyboard `e`.
- If the mouse does not select the dots, click the red box in the lower left to re-enable mouse input.
- Since both materials are assigned, press the keyboard `q` to finish.
- The default settings for this problem result in port meshes that are not very good with long thin triangles. To address that, select `Tools`→`Options` to pop up an options control window. Then select `Mesh`→`General`→`2D algorithm`→`Packing of parallelograms`. This choice is made simply by trying the various options to see which one produces the best mesh that avoids long thin triangles at the ports.
- Mesh the geometry by selecting in the options tree click ⊞`Mesh`→`3D`. A screenshot of the mesh is shown in Fig. 14.
- Save the meshing options with `File`→`Save Model Options`. The options file is automatically loaded when loading the `*.geo` file to enable exactly reproducing a mesh, if needed, or to experiment with different settings based on the previous set.
- Save the mesh by selecting `File`→`Save Mesh`.
- Quit gmsh.

7.2.3 Solving

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

```
#OpenParEMmaterials 1.0
Material
   name=air
```

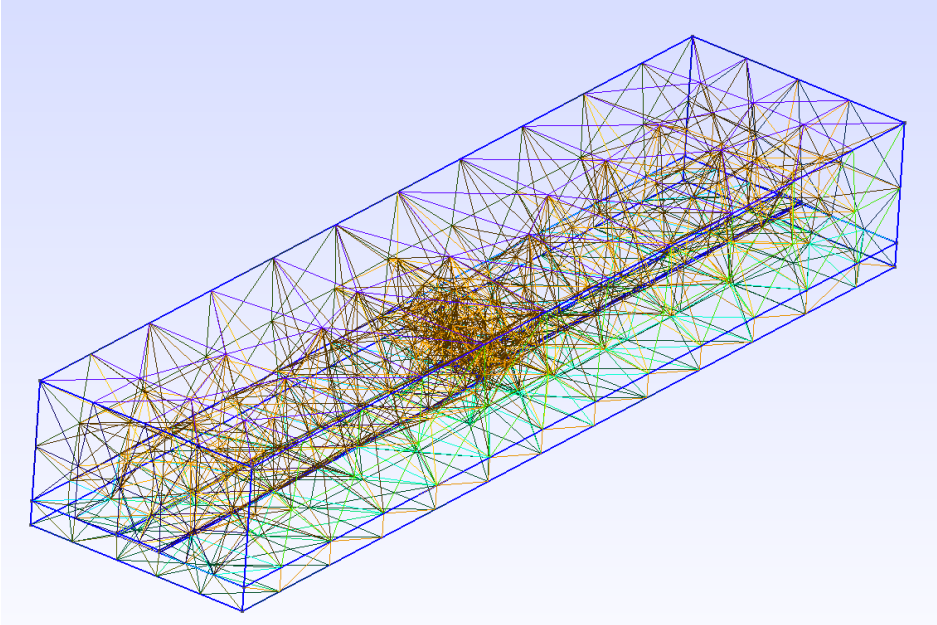


Figure 14: Screenshot of the meshed microstrip step in width.

```
   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=0.001
         Rz=0
      EndFrequency
   EndTemperature
   Source
      Generic numbers for testing.
   EndSource
EndMaterial
```

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

```
#OpenParEM3Dproject 1.0
project.save.fields            true
mesh.file                      microstrip_step.msh
mesh.quality.limit             100
mesh.order                     3
port.definition.file           microstrip_step_ports.txt
materials.global.path          ./
materials.global.name          //global_materials.txt
materials.local.path           ./
materials.local.name           local_materials.txt
refinement.frequency           high
refinement.iteration.min       1
refinement.iteration.max       40
refinement.required.passes     1
refinement.relative.tolerance  0.01
refinement.absolute.tolerance  1e-06
refinement.variable            SandH
frequency.plan.point           10e9
reference.impedance            0
```

Adaptive refinement is applied due to the step and the edges of the microstrip lines. With 3rd order elements, extensive refinement is not required. See Appendix A for a complete list of available keyword/value pairs that can be included in the project control file. Note that the reference impedance in the setup file is set to 0, so no renomalization of the S-parameters is performed.

- Run OpenParEM3D at the command line with

  ```
  $ OpenParEM3D microstrip_step.proj
  ```

  to run serially with a single core or with

  ```
  $ mpirun -q --oversubscribe -np 12 OpenParEM3D microstrip_step.proj
  ```

  to run in parallel with 12 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.

- View the 2-port S-parameters with any suitable command with the results looking very similar to

```
! 2-port S-parameter data
! Sport 1 net1
! Sport 2 net2
# GHz S DB R 0
!freq dBS11 angS11 dBS21 angS21 dBS12 angS12 dBS22 angS22
10 -14.599053663356 -117.699095892363 -0.1442073391584 52.408887125916 -0.1442073391584 52.408887125916 -14.3596394213266 42.8362354034876
```

  The output from OpenParEM3D shows the results from the 2D port simulations, and the impedances for the input and output ports are 48.1 Ω and 33.5 Ω, respectively. Considering just the transmission lines, the return loss from these impedances is 20*log10(abs(33.5-48.1)/(33.5+48.1))=-14.9 dB, which is close to the computed value of -14.6 dB that also includes the loading of the step discontinuity.

- Start ParaView and open the results file for port 1.
- Add a filter by selecting `Filters`→`Alphabetical`→`Slice`.
- In `Plane Parameters` select `Y Normal`.
- Change the y-component of the origin to 0.0002.
- Change `Solid Color` to `gridReE` and change the value shown from `Magnitude` to `Y`. In the buttons below, change the scale to be symmetric with a range of -4000 to +4000.
- A screen shot of the resulting Re(E_y) field just under the microstrip traces is shown in Fig. 15.

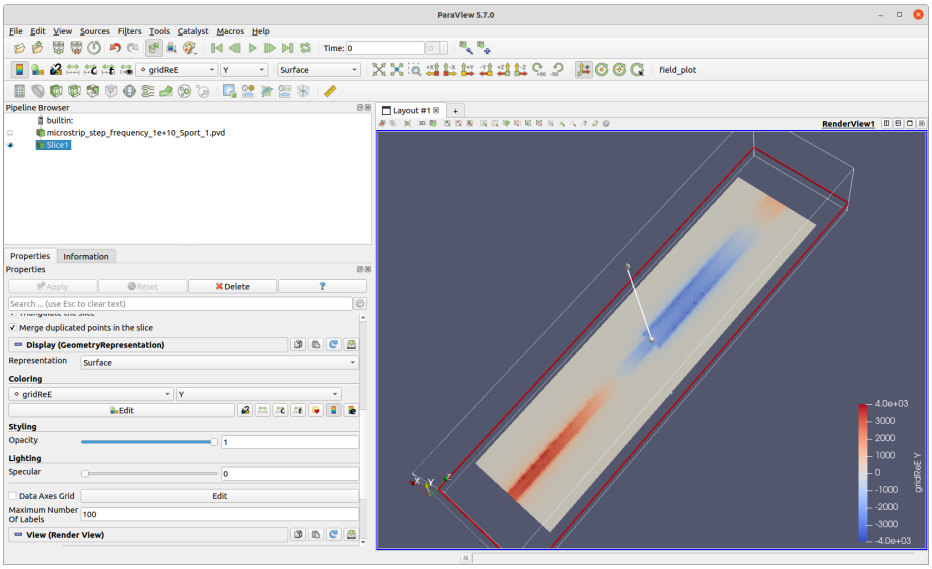


Figure 15: Screenshot of Re(E_y) just under the microstrip lines.

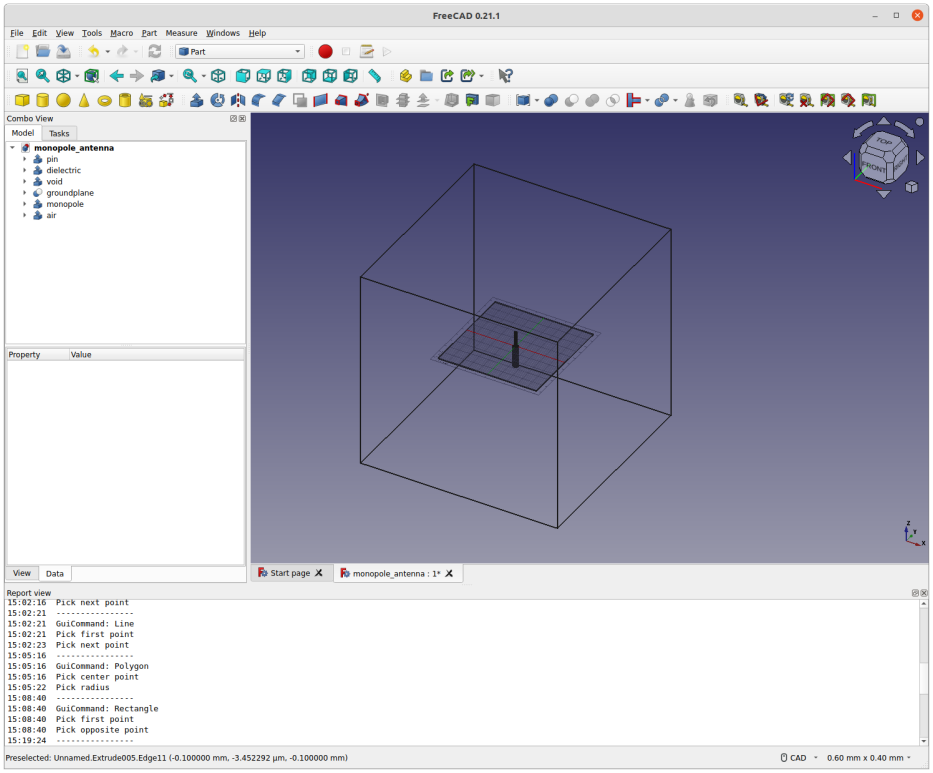
7.3 Monopole Antenna

A monopole antenna with coax feed is constructed and annotated for 1-port S-parameter simulation, meshed, and solved. The tutorial demonstrates the modeling of round conductors, ports internal to the 3D drawing space, and manually applied boundary conditions [radiation in this tutorial].

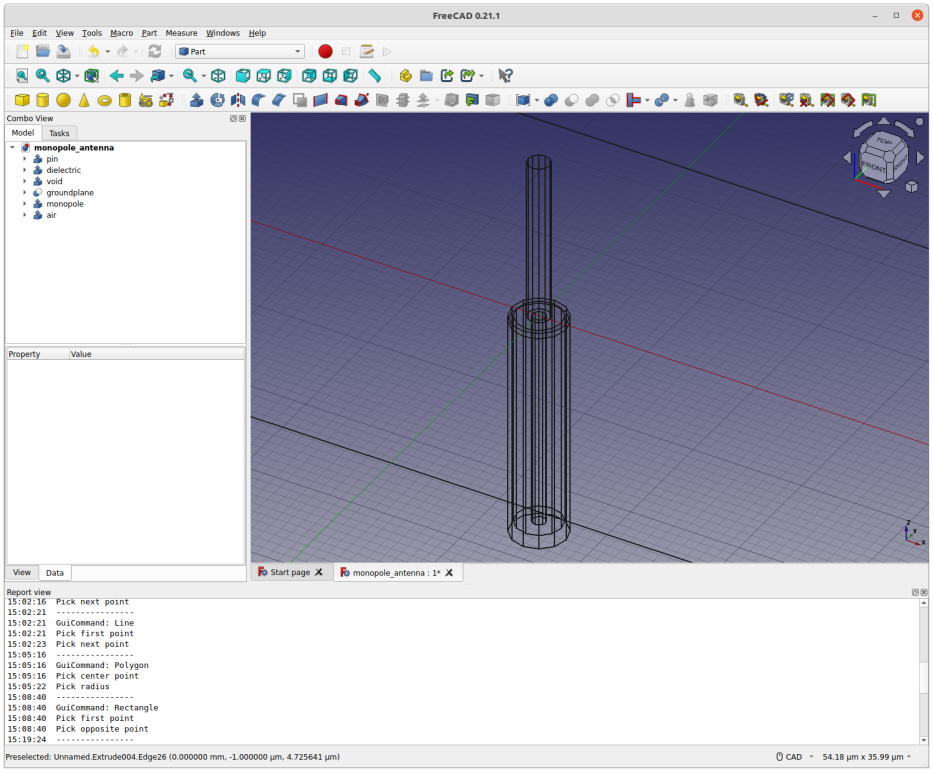
7.3.1 Drawing and Port Annotation

- Create a directory for the project called `monopole_antenna`.
- 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. 5.1.
- Assuming that the prior tutorials have been completed, references to the workspace and specific property boxes are omitted.
- Draw a multi-sided polygon to represent a circle by selecting `Drafting`→`Polygon`. Click on the drawing space to start the polygon then another place to complete it.
- Select the polygon and edit its properties to center it on the origin, set the number of faces to 12, and set the radius to 0.6um to end up with a 6 mm radius in the mesh.
- Extrude this polygon -20um to represent the pin of the feed coax. Change its label to `pin`.
- Draw another 12-sided polygon with a radius of 2.07um at the origin and extrude it -20um to represent the Teflon of the coax feed. Change its label to `dielectric`.
- Draw another 12-sided polygon with a radius of 2.5um at the origin and extrude it -21um to represent the outer conductor of the coax feed. Change its label to `void`. This will eventually be a void to represent metal that also clears a space to place the port at the end of the coax, which must be on the outside facing a void. The thickness of the outer conductor is somewhat arbitrary.
- Draw a rectangle that is 100um per side, center it on the origin, then extrude it -0.5um to represent the ground plane for the antenna. Change its label to `gp`.
- Save the drawing as `monopole_antenna`.

- The groundplane needs a hole in it to clear the monopole. Clone `void`, then select `gp`, add the clone to the selection, then Boolean cut the clone from `gp`. Change the label of the new object to `groundplane`.
- Yes, there is a lot of back-and-forth between the `Draft` and `Part` workspaces.
- Draw a 12-sided polygon with a radius of 1um at the origin and extrude it +15um to represent the dipole with an approximate resonant frequency of 5 GHz. Change its label to `monopole`.
- Draw a rectangle that is 200um per side, center it on the origin, extrude it 200um, then shift the resulting box in the z-direction -100um to represent air. Change its label to `air`.
- Right click in the drawing area and select `Draw style`→`Wireframe`. The view should look like that in Fig. 16.
- Right click in the drawing area and select `Draw style`→`As is`.
- Hide `air` and `void`.
- Set the view to `Bottom`, select one of the visible surfaces, and set the drawing plane to that surface.
- Draw a 12-side polygon to match the size of `dielectric`. Verify that the radius is 2.07 μm. Change its label to `_Pport`. This is the port on the end of the coax line.
- Draw a line from the lowest vertex of `dielectric` to the lowest vertex of `pin`. Verify that the length is 1.47 μm. Change its label to `_Pv`. This is the voltage integration line on the port.
- Set `air` to visible.
- In the `Draft` workspace, select each of the six sides of the air box in turn, make that side the drawing plane, and draw a rectangle covering the complete side of the air box by snapping to the corners. Do this six times and change the labels to `_Pfront`, `_Pback`, `_Pleft`, `_Pright`, `_Ptop`, and `_Pbottom`. These six rectangles completely cover the air box. Note that in a slightly more advanced usage, one side can be set as the drawing plane, and both rectangles parallel to that drawing plane can be drawn, so only three changes of the drawing plane are strictly necessary.
- Select `dielectric` and cut `pin` from it. Change its label to `teflon`. Because of the void, the teflon piece has a metal pin embedded within it.
- Select `air` and cut `void` from it.
- Select the resulting object and cut `groundplane` from it.
- Select the resulting object and cut `monopole` from it.
- Select the resulting object and add `teflon` to the selection, then create a Boolean fragment.
- Select the Boolean fragment and export a BREP file called `monopole_antenna.brep`.
- For the port, add text items with the labels `_Sin(PV)port` `_Min(1,voltage)v`.
- For the radiation boundary conditions, add text items with the labels `_Bfront(radiation)front`, `_Bback(radiation)back`, `_Bleft(radiation)left`, `_Bright(radiation)right`, `_Btop(radiation)top`, and `_Bbottom(radiation)bottom`.
- Run the macro "OpenParEM3D_save.py" and save the results in `monopole_antenna_ports.txt`. Verify that there are no errors.
- Save the file and exit FreeCAD.



(a) View of all



(b) Zoom towards center

Figure 16: Monopole building blocks.

7.3.2 Meshing

- Start gmsh and open the saved BREP file.
- Assign the two volumes with their respective materials of `air` and `teflon`.
- Create the 3D mesh and save it.
- Exit gmsh.

7.3.3 Solving

- Create the materials file `local_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

Material
   name=teflon
   Temperature
      temperature=any
      Frequency
         frequency=any
         er=2.2
         mur=1
         tand=0.0
         Rz=0
      EndFrequency
   EndTemperature
   Source
      generic teflon
   EndSource
EndMaterial
```

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

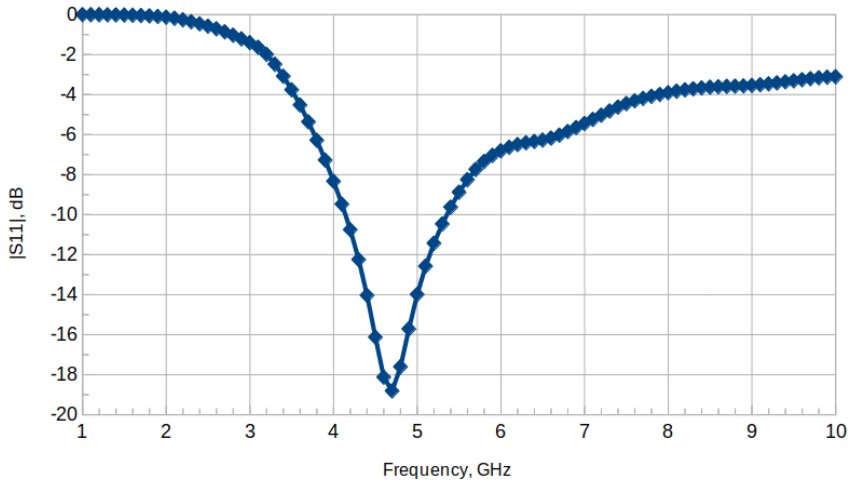
```
#OpenParEM3Dproject 1.0
project.save.fields           true
mesh.file                     monopole_antenna.msh
mesh.quality.limit            1e4
mesh.order                    4
port.definition.file          monopole_antenna_ports.txt
materials.global.path         ./
materials.global.name         //global_materials.txt
materials.local.path          ./
```

```
materials.local.name            local_materials.txt
refinement.frequency            plan
refinement.iteration.min        1
refinement.iteration.max        40
refinement.required.passes      1
refinement.relative.tolerance   0.01
refinement.absolute.tolerance   5e-06
refinement.variable             SandH
frequency.plan.point.refine     7e9
frequency.plan.linear           1e9,10e9,0.1e9
reference.impedance             0
```

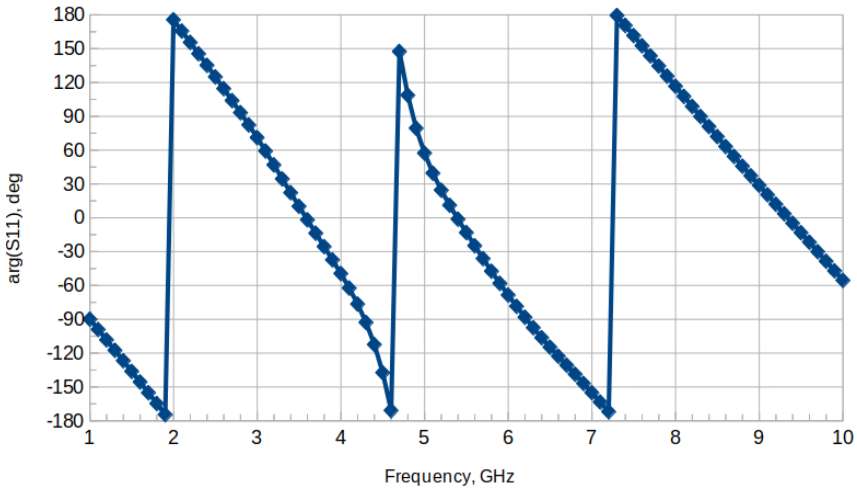
- Run OpenParEM3D at the command line with

```
$ mpirun -q --oversubscribe -np 12 OpenParEM3D monopole_antenna.proj
```

- The computed S-parameters are shown in Fig. 17. The real part of the electric field computed at 4.7 GHz is shown in Fig. 18, while Fig. 19 shows the real part of the magnetic field.

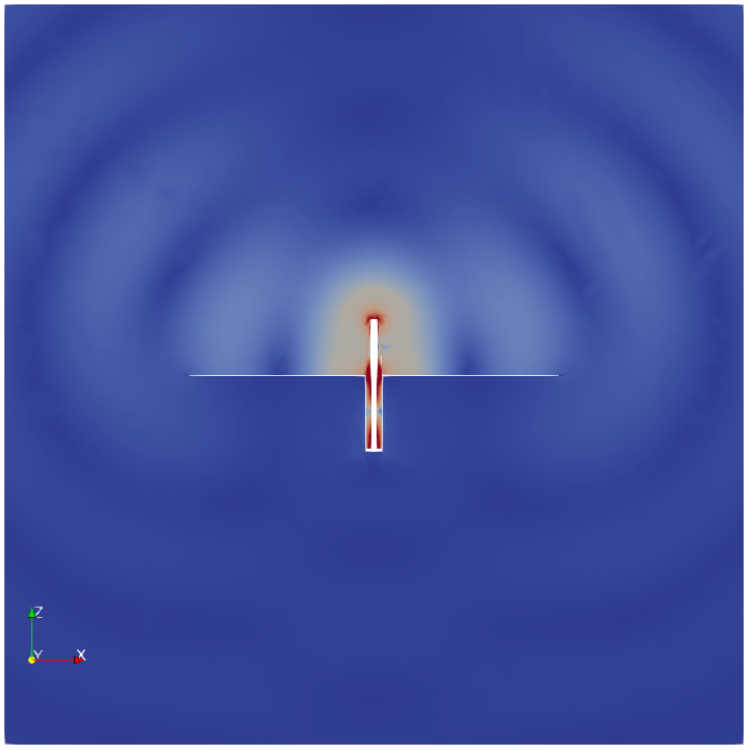


(a) Magnitude.

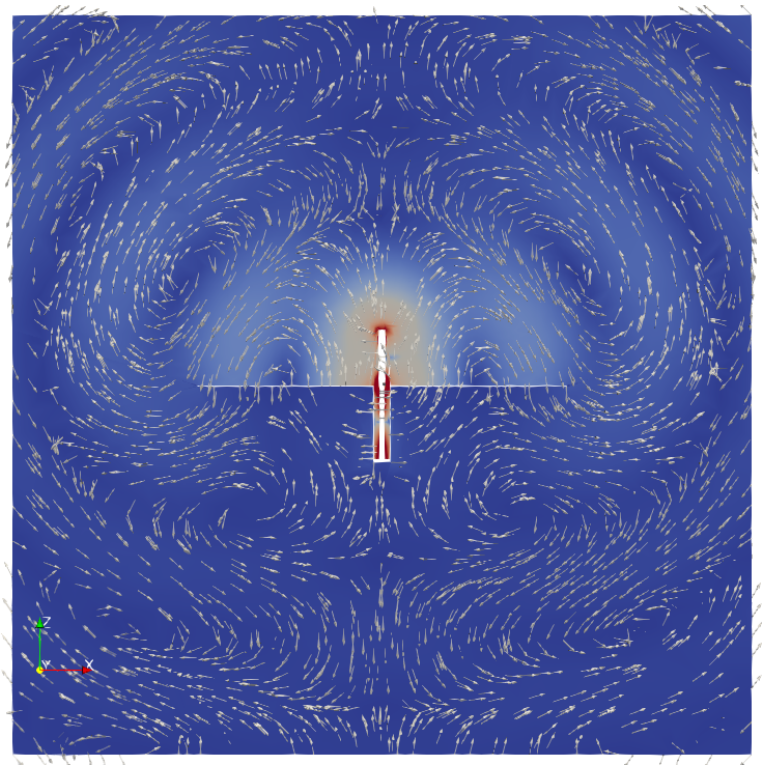


(b) Phase.

Figure 17: Computed S-parameters for a monopole antenna.

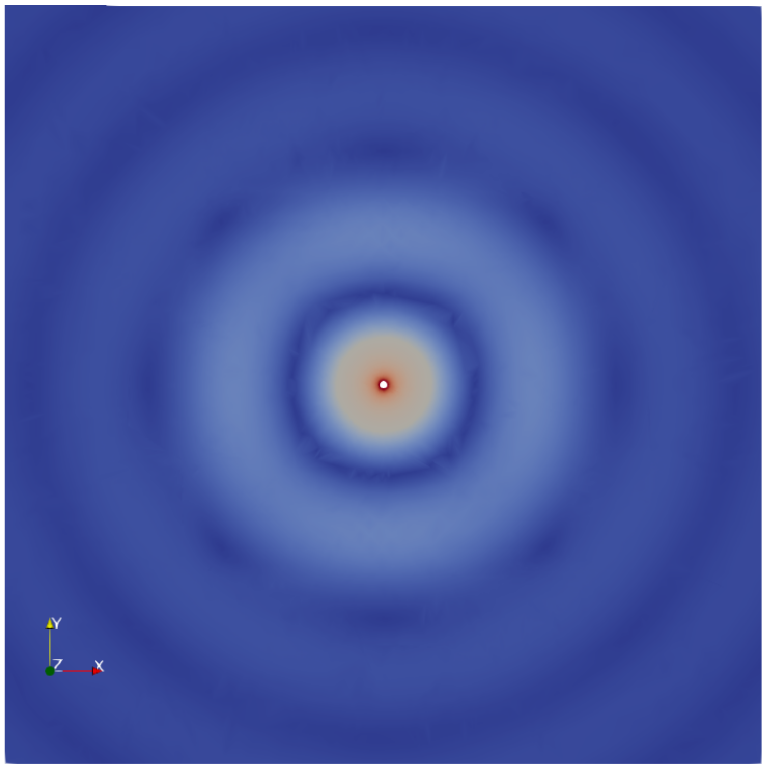


(a) $|\mathrm{Re}(\overline{E})|$

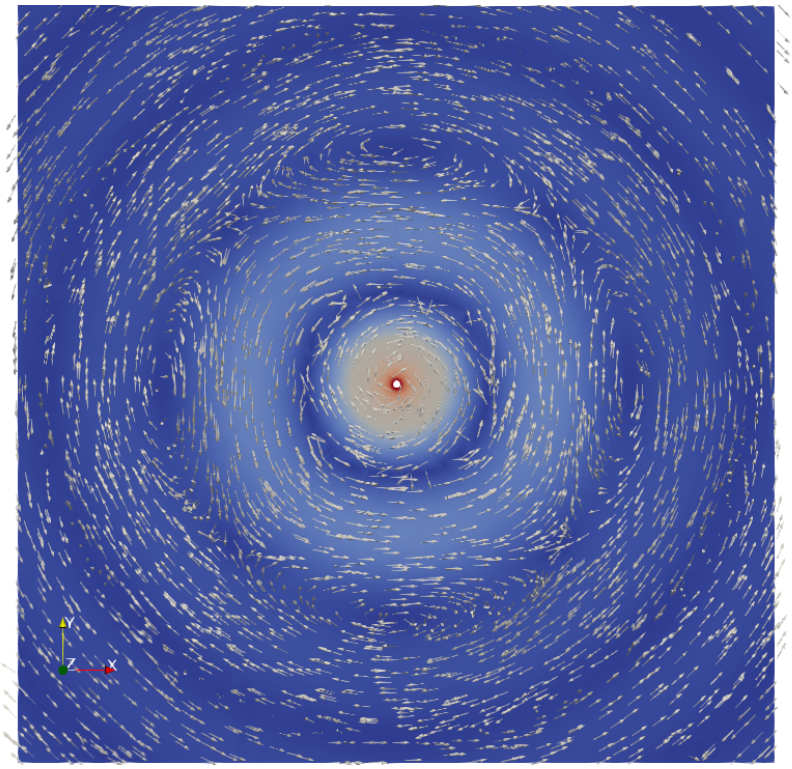


(b) $\mathrm{Re}(\overline{E})$

Figure 18: Real part of the electric field.



(a) $|\mathrm{Re}(\overline{H})|$



(b) $\mathrm{Re}(\overline{H})$

Figure 19: Real part of the magnetic field.

8 Techniques

8.1 Wave Ports

The math behind OpenParEM3D assumes that the 3D electromagnetic fields at the ports represent transmission lines or waveguides, meaning that the field dependence along the direction orthogonal to the port ($\hat{n}$) has the dependence $e^{-\gamma n}$. Ports must be placed on the ends of transmission lines or waveguides that are long enough to ensure that the electromagnetic fields have settled into their modal configurations. How long is long enough? A reasonable rule-of-thumb is to make the transmission lines or waveguides twice as long as the lateral extent of the fields, which may or may not align with the dimensions of the port. For critical work, it could be reasonable to run the problem a second time with a longer ports to see if the S-parameters change. Visual inspection of the fields from a completed solution can also help determine if the ports are long enough. *Failure to ensure that the ports are long enough will result in incorrect answers with no warning from OpenParEM3D.*

8.2 Multimode Ports With Unequal γ

For non-driven ports, an absorbing boundary condition is applied to absorb energy propagating towards the port from within the 3D space. The absorbing boundary condition (ABC) is dependent on the complex propagation constant, (γ), selected for the 2D ABC, and only one γ can be applied at a given port. If more than one mode is present, such as for differential pairs, the largest γ is used. If the γ for the other modes is significantly different, then they will suffer some reflection that puts energy back into the 3D space that affects the solution. The larger the reflection, the worse the effect on accuracy. OpenParEM3D detects when the span of γ is too large, prints an error, and exits. However, there may be cases where the impact on accuracy is not significant, so the option to override the error detection is available. To allow OpenParEM3D to simulate cases with multimode ports with non-equal γ, set the keyword `solution.check.homogeneous` to `false`. This limitation is expected to be addressed with a PML absorbing boundary, when implemented.

8.3 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 in 2D simulations, although order-ramping in OpenParEM2D may offer a fix, when implemented.

The disadvantages ramp quickly with increasing `N`. OpenParEM3D 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 the 2D port solutions take more than a few seconds or require 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 for the 2D simulations can be caused by a poor mesh at the ports, so it is important to ensure that the mesh has good triangulation at the ports 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 the S-parameters.
- Dense meshes with element edges that are small compared to the wavelength should use smaller N to avoid numerical issues due to over-meshing as discussed in Sec. 8.4.
- Meshes with large areas where there is little field variation should use smaller N to avoid numerical issues due to over-meshing as discussed in Sec . 8.4.
- ParaView supports `N` up to 6. If ParaView will be used to review field plots, then keep `N` to 6 or fewer.

8.4 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 will degrade. For finite element programs such as OpenParEM3D, 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 make it worse.

The same effect can happen by choosing finite element orders that provide too much variability than that required by the problem. For example, a perfectly linear field structure would be accurately described by 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.

8.4.1 mesh edge length/wavelength in element

OpenParEM3D provides an output during execution to help with determining settings to avoid over-meshing, and this is the `mesh edge length/wavelength in element` line. At each frequency, the length of each edge of every element is divided by the wavelength in the material in which the element resides, then the minimum and maximum fractions are reported. So for example, the output will show something like

```
mesh edge length/wavelength in element: shortest=0.000965796 longest=0.00846018
```

In this example, the longest element is short compared to the wavelength at just 0.85%. The field cannot change very much in such a small element, so low-order finite elements are appropriate, and care must be taken to not force too many iterations.

8.4.2 concentrated fields

Geometries that have concentrated fields in small areas that are located within large spaces result in areas with limited field variations. These should generally use lower-order finite elements to prevent high-order elements from causing numerical problems modeling fields with low spatial variation away from the area of concentrated fields. An example of such a case is a step in width of microstrip.

8.4.3 diffuse fields

Geometries with fields varying throughout the problem space strongly benefit from using higher-order elements.

8.4.4 small mesh elements

Initial meshes with very small maximum length elements reported by `mesh edge length/wavelength in element` should generally use lower-order finite elements.

8.4.5 large mesh elements

Initial meshes with large maximum length elements reported by `mesh edge length/wavelength in element` can take advantage of higher-order finite elements to accelerate convergence with adaptive mesh refinement.

8.4.6 refinement.required.passes

Sometimes, the S-parameter relative tolerance can be met before the field accuracy is sufficient when a mesh refinement does not happen to significantly impact the S-parameters. A protection against this is to set `refinement.required.passes` to 2 or more. However, this can lead to over-meshing, especially for larger problems.

8.4.7 refinement.absolute.tolerance

Use of `refinement.absolute.tolerance` can drive adaptive mesh refinement to excessive numbers of iterations if it is set too low. It is most useful in helping to avoid early termination when a mesh adaptation does not happen to produce a large change in the S-parameters, so the relative convergence criteria is met too early. Instead of setting `refinement.required.passes` to 2 or more, `refinement.absolute.tolerance` can be set to a moderate error and `refinement.variable` to `SandH` to avoid an early termination.

8.4.8 1st-order elements

1st-order elements are not the most efficient way to solve a problem, but they are probably the safest in avoiding over-meshing. New users may choose to use 1st-order elements to gain experience before moving on to higher-order elements. Experienced users may find it useful to re-run a questionable case with 1st-order elements to double-check results.

8.5 FreeCAD Port Alignment to Mesh

The port must align exactly to the mesh. In cases where the port does not fall naturally on geometric features, simply include the port in the selection while creating the Boolean fragments object for exporting a BREP file. During meshing, gmsh will force mesh elements to follow the port outline.

Including voltage integration paths in the Boolean fragments object will also force mesh edges to follow the path, and this may slightly improve the simulation accuracy.

8.6 Mesh Re-Use

When using adaptive mesh refinement, the refined mesh can be saved by setting `mesh.save.refined true` in the project control file. This can be used to advantage when multiple frequency sweeps are expected. A first simulation pass can be done to just adaptively refine the mesh. Then, the project control file can be copied, the mesh changed to the refined mesh, refinement disabled, and the frequency sweep added. Multiple frequency sweeps can be make without having to re-refine the mesh. For example, a relatively coarse sweep can be made, then finer sweeps can be done to fill in sharp resonance peaks and nulls.

The refined mesh is in the MFEM native format rather than the gmsh format. To view the MFEM mesh directly, glvis can be used.

8.7 Adaptive Mesh Refinement Convergence

There are two controls for deciding when to terminate adaptive mesh refinement: convergence on S with a relative error and convergence on the H field with an absolute error. One or the other or both can be used as the convergence criteria.

8.7.1 S with relative error

In the project control file, set `refinement.variable S` and `refinement.relative.tolerance reltol`, where `reltol` is set to some value such as 0.01.

Converging exclusively on S with a relative error can result in early termination of adaptive refinement. Sometimes a round of mesh refinement improves the fields but does not significantly change the S-parameters, leading to termination while the field accuracy is still low. To protect against this, the project control file can require more than one consecutive iterations using `refinement.required.passes N` to meet the relative convergence criteria. However, this can turn into a bit of a guessing game: what relative convergence to set and how many consecutive iterations are needed.

8.7.2 H with absolute error

In the project control file, set `refinement.variable SandH`, `refinement.relative.tolerance reltol`, and `refinement.absolute.tolerance abstol`. Set `reltol` to a large value such as 0.1 such that it very easy to meet the convergence criteria on S. Set `abstol` to a value such as 1e-06. Iterations will proceed

until both convergence criteria are met, and since S is easy, then the criteria on the field error becomes the ultimate gating limit.

The adaptive mesh refinement algorithm calculates an absolute error metric on the H field. Setting the convergence criteria so that the field error becomes the gating limit ensures that the fields reach a good level of refinement. The downside is that if `abstol` is set too tight, extra iterations may be made even though the S-parameters may be converged.

8.7.3 S with relative error *and* H with absolute error

Use the `SandH` refinement option as above, but set `reltol` to a value such as 0.01. Use an `abstol` such as 2.5e-06 or somewhat higher. Set `refinement.required.passes 1`.

This setup is perhaps the safest. Effectively, adaptive refinement continues while the field error is high, preventing early termination when the relative error on S does not change much. After the fields are well refined, then iteration continues, if needed, until the S-parameters converge on the relative error.

8.7.4 S with relative error *or* H with absolute error

This option is included for completeness, but it may not be useful in general.

8.8 Conductor Loss

Conductor losses are added with polygons defined on each surface to convert from the default perfect electric conductor (PEC) boundary to some other conductor. Until additional scripting is available in FreeCAD to automate this, it can be very tedious to draw these except for the simplest of cases.

If all metals are the same, then there is an option to change the default boundary condition from PEC to a defined material, such as copper. To make this change, add the keyword/value pair
`materials.default.boundary some_metal` to the project control file, where `some_metal` is a conductor defined in either the local or global material databases. Other boundary conditions can still be applied to override the default boundary condition.

8.9 OpenParEM2D Run Time with MPI

OpenParEM3D runs OpenParEM2D at the wave ports, and with MPI parallel processing, an efficiency issue occurs with OpenParEM2D. With MPI, the 3D problem is subdivided and spread across many processors as described in Sec. 3. Generally, the surface areas representing wave ports are a small fraction of the overall 3D problem, so when a large number of processors are specified to `mpirun`, only a small percentage of them are involved with solving the 2D problem with OpenParEM2D. However, all the processors must still communicate to solve the problem, so for OpenParEM2D, the MPI overhead of communication increases but the computational power does not.

When running OpenParEM2D standalone, increasing the number of processors reduces run time until the computer saturates its bandwidth. When running OpenParEM3D, increasing the number of processors reduces the 3D run time [until the computer saturates], but the OpenParEM2D run times while being run from OpenParEM3D will generally `increase` even for small numbers of processors.

The overall run time of a 3D problem is dominated by the 3D solve time, not the 2D port solve times. So while OpenParEM2D suffers a reduction in efficiency, the impact on overall run times is negligible. However, it can be disconcerting to see the port solves slowing down as the number of processors applied for MPI increases.

8.10 OpenParEM2D Convergence

The first step in solving a 3D problem is to solve the 2D eigenvalue problem for the fields and propagation constants at the wave ports. Since these calculations are full-wave electromagnetic simulations, they can become problematic if they are failing to converge since that stalls the 3D simulation. Scenarios leading to issues with 2D wave port convergence are described in the following sections.

8.10.1 Poor Mesh Quality

Since the 2D simulations are eigenvalue problems, they are more susceptible to poor mesh quality that the 3D simulation. The output from OpenParEM2D shows the mesh quality at the port, and if the mesh quality is too low (i.e. a large number), then convergence can become very slow. There are two remedies for this issue: re-mesh to improve the mesh quality at the ports, or decrease the finite element order since high-order elements generally require better mesh quality.

8.10.2 Large Frequency Step

Once the 2D simulations successfully complete the first time, the complex propagation constants are stored and used as initial guesses for the next 2D solve. If the frequencies are closely spaced, then the initial guess is good and the next 2D solve is very quick without convergence issues. However, if the step is too large, then the initial guess is not as good and convergence issues may occur. The remedy in this case 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 are applied and good accuracy will still be obtained.]

8.11 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. OpenParEM3D uses iterative solvers for the linear solves of $\overline{E}$ and $\overline{H}$, 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 the timing of MPI data packets. The net result is that the answers produced by OpenParEM3D change slightly from run-to-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.

8.12 Companion Tool builder

A companion tool called *builder* helps to quickly build projects for straight sections of some common transmission line and waveguide types. Builder is primarily a tool for optimization of dimensions and materials for characteristic impedance using OpenParEM2D. However, it includes the option for providing a length to extrude the 2D cross-section into a 3D object. The target uses are calculations of the effect of the conductor losses on delay and generation of S-parameter files for use in circuit simulations with transmission lines or waveguides. 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 OpenParEM3D 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

OpenParEM3D 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 `#OpenParEM3Dproject 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.calculate.poynting	bool	false	Calculate the Poynting vector (power flow)
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.save.refined	bool	false	Save the mesh for re-use in a later run
mesh.refinement.fraction	double	0.005	Maximum fraction of the mesh to refine at each iteration
mesh.quality.limit	double	20	Limit above which a fatal error is triggered. Typically has to be set higher
port.definition.file	string		File name for the boundary condition/port definition 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
materials.default.boundary	string	PEC	Default material to use for unspecified boundaries. PMC and metals, such as copper, from the materials library are valid entries
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
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.relative.tolerance	double	0.02	Relative tolerance for convergence during adaptive refinement
refinement.absolute.tolerance	double	1e-06	Absolute tolerance for convergence during adaptive refinement.

Keyword	Value	Default	Description
refinement.variable	string	SandH	Variable to test for convergence Options: • S - Converged if the S-parameter error metric meets refinement.relative.tolerance • SorH - Converged if the S-parameter error metric meets refinement.relative.tolerance *or* the electric field error meets refinement.absolute.tolerance • SandH - Converged if the S-parameter error metric meets refinement.relative.tolerance *and* the electric field error meets refinement.absolute.tolerance
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
reference.impedance	double	50	Renormalize the S-parameters to the given impedance. Use 0 to skip renormalization
touchstone.frequency.unit	string	GHz	Valid inputs are Hz, kHz, MHz, and GHz
touchstone.format	string	DB	Valid inputs are DB, MA, and RI. Angles are in degrees
solution.temperature	double	25	Temperature used for materials selection
solution.2D.tolerance	double	1e-13	Tolerance for the 2D eigenvalue solution and H field calculation
solution.3D.tolerance	double	1e-13	Tolerance for the 3D field solutions
solution.iteration.limit	int	5000	Iteration limit for the iterative eigenvalue and Hfield solvers
solution.check.homogeneous	bool	true	Check if the complex propagation constant is uniform across the modes at each port
solution.modes.buffer	int	0	Additional number of modes over solution.modes to solve in 2D port simulations. Increase this value if not all of the solution.modes solutions are found
solution.check.closed.loop	bool	true	Check if current loops are closed. Set to false when use of symmetry causes valid open-loop current paths
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
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
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 when running OpenParEM2D
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.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

Keyword	Value	Default	Description
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.port.definitions	bool	false	Show the setups used for port definitions
debug.show.impedance.details	bool	false	Show all voltages, currents, powers, and impedance calculations
debug.save.port.fields	bool	false	Show 2D port fields at various points in the 3D calculation
debug.skip.mixed.conversion	bool	false	Skip the mixed mode conversion if the setup calls for one using DifferentialPair/EndDifferentialPair blocks
debug.skip.forced.reciprocity	bool	false	Skip enforcement of reciprocity on S-parameter matrices
debug.tempfiles.keep	bool	false	Keep temporary files

B Materials File Specification

Materials are specified in text files with the format given below. Two material files can be specified in the project control file: global and local. The global file can be a general materials library, and the local file can contain a specialized set of materials just for the project. When both global and local material files are specified, the local materials file takes precedence over the global file when the same material name is used in both.

Dielectrics can be specified with a Debye model or by properties at a single frequency or a list of frequencies. Conductors are specified by properties at a single frequency or a list of frequencies.

Note that material properties files and information contained in copyrighted simulators are covered by the copyright of the simulator. It is a violation of the copyright to copy material parameters from these sources. Valid sources for material parameters are datasheets, books, and papers with appropriate citations in the Source/EndSource block.

```
// a comment

#OpenParEMmaterials 1.0   // required on first real line

// All units are MKS
// meter, Ohms, Hz, S/m, Celsius
// "any" means that the model applies at all temperatures

// Any number of Material blocks can be specified.

// Debye model
// dielectric only
Material
   name=text
   Temperature
      {temperature||temp||t}={double||any}
      {epsr_infinity||er_infinity}=double
      {delta_epsr||delta_er}=double
      m1=double
      m2=double
      {relative_permeability||mur}=double
      {loss_tangent||tand||tandel||conductivity||sigma)=double
   EndTemperature
   ... more Temperature blocks, optional
   Source
      any text
      any text
      ...
      any text
   EndSource
   ... more Source blocks, optional
EndMaterial

// frequency list
// dielectrics or metals
// linear interpolation is supported
// extrapolation is not supported
Material
   name=text
   Temperature
      {temperature||temp||t}={double||any}
      Frequency
         {frequency||freq||f}={double||any}
         {relative_permittivity||er||epsr}=double
         {relative_permeability||mur}=double
         {loss_tangent||tand|||tandel||conductivity||sigma)=double
         Rz=double   // for surface roughness if a conductor
```

```
      EndFrequency
      ... more Frequency blocks, optional
   EndTemperature
   ... more Temperature blocks, optional
   Source
      any text
      any text
      ...
      any text
   EndSource
   ... more Source blocks, optional
EndMaterial
```

C Port File Specification

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

```
// a comment

#OpenParEMports 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,
// locations for boundary conditions, and ports.
// 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,double)      // (x,y,z)
   point=(double,double,double)      // any number of points
   ...
   point=(double,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.
// Definition is unchanged from OpenParEM2D - see OpenParEM2D specification for details.
Boundary
   name=string
   type=surface_impedance|perfect_electric_conductor|perfect_magnetic_conductor|radiation
   material=string            // required for surface impedance
   wave_impedance=double      // optional; default wave impedance of free space ~ 376.73
   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 ports.
// S parameter ports must be numbered sequentially across all ports starting with 1.  0 is not
// a valid S-port number.
//
// impedance_calculation=modal is for expert use only.  The user must set up the integration
// paths correctly for for the modes as found in OpenParEM2D.
//
// The Mode blocks are applied in order, so the first mode found in OpenParEM2D is
// paired with the first Mode/EndMode block, etc.  An example of misalignment is for coupled
// microstrip where the first mode in OpenParEM2D is the even mode, so if the the first
// Mode/EndMode block gives a differential integration path, then the computed voltage and
```

```
// currents will be zero.

Port
   name=text  // text name for a port; can be a number, but these are not used as S-port numbers
   path=name  // define the outline of the port on the outside of the 3D problem
   path-=name
   ...
   path+=name
   impedance_definition=VI | PV | PI       // see OpenParEM2D specification for description
   impedance_calculation=modal | line      // see OpenParEM2D specification for description

   // Specify any number of modes per port.  Each mode can have two IntegrationPaths, one for
   // voltage and one for current.
   //
   // Either Mode or Line blocks can be used, but pairing Mode with impedance_calculation=modal
   // and Line with impedance_calculation=line makes for a cleaner file.
   Mode
      Sport=integer           // S-parameter port number
      [net=text]
      IntegrationPath
         type=voltage|current
         [scale=double]          // default=1
         path=+name
         path-=name
         ...
         path=name
      EndIntegrationPath
         IntegrationPath
         type=voltage|current
         [scale=double]          // default=1
         path=+name
         path-=name
         ...
         path=name
      EndIntegrationPath
   EndMode
   Line
      Sport=integer          // S-parameter port number
      [net=text]
      IntegrationPath
         type=voltage|current
         [scale=double]          // default=1
         path=+name
         path-=name
         ...
         path=name
      EndIntegrationPath
         IntegrationPath
         type=voltage|current
         [scale=double]          // default=1
         path=+name
         path-=name
         ...
         path=name
      EndIntegrationPath
   EndLine
   // Specify any number of differential pairs for impedance_calculation=line only
   // and the number of pairs must align with the number of Sports
   DifferentialPair
```

```
      Sport_P=integer
      Sport_N=integer
   EndDifferentialPair
EndPort
```

D Regression Suite

The projects are located in the installation area in the "regression" directory. For cases comparing against literature results, see the document "OpenParEM3D_Theory_Methology_Accuracy.pdf" for references.

Project	Description
WR90/opened/opened.proj	Open-circuited 1-port waveguide to test PMC boundary condition and S11 extraction with strong reflection
WR90/shorted/shorted.proj	Short-circuited 1-port waveguide to test PEC boundary condition and S11 extraction with strong reflection
WR90/shorted-adaptive/shorted.proj	Re-run the project WR90/shorted using adaptive refinement to test convergence
WR90/shorted-adaptive/shorted_field.proj	Re-run the project WR90/shorted-adaptive to test convergence using the absolute error in the electric field
WR90/straight/straight.proj	Straight section of waveguide to test S-parameter magnitude and phase accuracies
WR90/straight-adaptive/straight.proj	Re-run of the project WR90/straight using adaptive refinement to test convergence
WR90/straight-adaptive-reuse/straight.proj	Re-run of the project WR90/straight to test the use of a previously refined mesh
WR90/straight-asymmetric/straight-asymmetric.proj	Similar the project WR90/straight to test with an asymmetric mesh that mesh rotations are properly applied
WR90/straight-flipped/straight-flipped.proj	Flipped section of waveguide to test port orientation
WR90/straight-rotated-x/straight-rotated.proj	Waveguide with rotation about x-axis to test port orientation
WR90/straight-rotated-y/straight-rotated.proj	Waveguide with rotation about y-axis to test port orientation
WR90/straight-rotated-xyz/straight-rotated.proj	Waveguide with arbitrary rotation to test port orientation
WR90/straight-lossy/straight.proj	Straight waveguide with lossy walls to test the impedance boundary condition
WR90/straight-loaded/straight-loaded.proj	Loaded straight waveguide to test S-parameter and phase accuracy for a mixed dielectric system
WR90/straight-loaded-adaptive/straight-loaded.proj	Re-run of the project WR90/straight-loaded to test adaptive mesh refinement for a mixed dielectric system
WR90/straight-loaded-lossy/straight-loaded.proj	Re-run of the project WR90/straight-loaded the impedance boundary condition for a mixed dielectric system
WR90/straight-steps/small/straight-steps.proj	Cascade of waveguides with different impedance and shunt capacitors to test S-parameter calculation against the direct calculation computed by tline3
WR90/straight-steps/small-renorm/straight-steps.proj	Re-run of the project WR90/straight-steps/small to test renomalization against the direct calculation computed by tline3
WR90/tee/tee.proj	Junction of waveguides to test 3-port S-parameter calculations
permeability/rectangular_waveguide/straight.proj	Straight section of waveguide to test S-parameter magnitude and phase accuracies for $\mu_r = 2$
WR75/dielectric-loading/dielectric-loading.proj	Waveguide with dielectric puck obstacle to test against literature result
WR75/T-Junction/T-Junction.proj	Junction of waveguides to test against literature result
partially_filled_rect_waveguide/partially_filled_rect_waveguide_lossless/part.proj	Partially-filled waveguide with TM dominant mode to test the absorbing boundary condition and S-parameter magnitude and phase accuracies
partially_filled_rect_waveguide/partially_filled_rect_waveguide_lossy/part.proj	Re-run of partially_filled_rect_waveguide/partially_filled_rect_waveguide_lossless to test the impedance boundary condition
permeability/partially_filled_rect_waveguide/part.proj	Partially-filled waveguide with TM dominant mode to test the absorbing boundary condition and S-parameter magnitude and phase accuracies for $\mu_r = 2$
slotline/step/step2.proj	Step in width of slotline to test accuracy of edge-coupled transmission line against literature result
slotline/step/step2_field.proj	Re-run the project /slotline/step to test convergence using the absolute error in the electric field
microstrip/bridge/bridge_3p78.proj	Microstrip bridge to test accuracy against literature result
microstrip/filter/filter.proj	Microstrip filter to test accuracy against literature result

Project	Description
microstrip/single_line_short_PI/single.proj	Short lossy microstrip to test S-parameter calculation using the PI definition of characteristic impedance and to test S-parameter magnitude and phase accuracy
microstrip/single_line_short_PV/single.proj	Short lossy microstrip to test S-parameter calculation using the PV definition of characteristic impedance and to test S-parameter magnitude and phase accuracy
stripline/differential_pair_modal/diffPair.proj	Short lossy homogeneous differential pair stripline to test S-parameter magnitude and phase accuracy
stripline/differential_pair_line_mixed/diffPair.proj	Short lossy homogeneous differential pair stripline to test S-parameter magnitude and phase accuracy with mixed mode conversion and mixed-mode Touchstone output
stripline/differential_pair_line/diffPair.proj	Short lossy homogeneous differential pair stripline with single-ended S-parameter Touchstone file output
stripline/differential_pair_via_modal/diffPair.proj	Short lossy homogeneous differential pair stripline with one trace shorted to ground using manual p-convergence
stripline/differential_pair_via_line_mixed/diffPair.proj	Short lossy homogeneous differential pair stripline with one trace shorted to ground with mixed mode conversion and mixed-mode Touchstone output
stripline/differential_pair_via_line/diffPair.proj	Short lossy homogeneous differential pair stripline with one trace shorted to ground with single-ended S-parameter Touchstone file output
stripline/differential_pair_via_rot1_line_mixed/diffPair.proj	Rotated short lossy homogeneous differential pair stripline with one trace shorted to ground with single-ended S-parameter Touchstone file output
stripline/Simonovich_stripline/Simonovich_stripline.proj	Straight lossy stripline to test against literature result
antenna/square_monopole/square-planar.proj	Square planar monopole antenna to test accuracy against literature result

E Accuracy Studies

The projects are located in the installation area in the "regression" directory. The cases are discussed in detail in "OpenParEM3D_Theory_Methology_Accuracy.pdf".

Project	**Description**
microstrip/filter_study/filter.proj	Microstrip bandpass filter with comparison to measurement from the literature
antenna/square_monopole_study/square-planar.proj	Rectangular monopole antenna with comparison to measurement from the literature
microstrip/bridge_study/bridge_*.proj	Microstrip bridge with comparison to simulation from the literature
slotline/step_study/step2.proj	Step in width for slotline with comparison to simulation from the literature
WR75/T-Junction_study/T-Junction.proj	Waveguide T-Junction with comparison to simulation from the literature
WR75/dielectric-loading_study/dielectric-loading.proj	Waveguide loaded with dielectric puck with comparison to simulations from the literature
stripline/Simonovich_stripline_study/Simonovich_stripline.proj	Lossy stripline with comparison to measurement and simulation from the literature
WR90/straight_study/straight_order_*.proj	Straight waveguide with comparisons vs. analytical results for varying orders and mesh densities

References

[1] `https://gmsh.info/`

[2] `https://www.freecad.org/`

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

[4] 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.

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

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

[7] `https://ibis.org`