OpenParEM Installation Manual

Version 2.1

April 2025

Brian Young

Contents

1 Introduction

OpenParEM consists of the 3D simulator OpenParEM3D and the 2D simulator OpenParEM2D plus some helper tools. All can be compiled from source or installed as pre-compiled bindaries for x86 architectures.

OpenParEM uses command-line tools for solving electromagnetic problems once provided with the necessary input files. The primary required input files are a mesh describing the geometry, text files describing modes for OpenParEM2D and ports for OpenParEM3D, and a text project control file. The user is responsible for pulling together the necessary tools to create the needed files.

Assembling a tool flow is a very significant task. For the development of OpenParEM, a tool flow is developed around FreeCAD [1], gmsh [2][3], and ParaView [4]. The user's manuals describe in detail how to use these tools to complete projects. For FreeCAD and ParaView, provided Python scripts greatly simplify the effort to set up and review solutions.

2 Installing Pre-compiled Binaries

Pre-compiled binaries are available for x86 Linux and do not require administrator priviledges. The basic steps for installing binaries are listed below. See the INSTALL file for more details and installation options.

- Download the installation package OpenParEM-*version*-bin.tar.gz
- Unpack the package:

```
$ tar -xf OpenParEM-version-bin.tar.gz
$ ln -s OpenParEM-version OpenParEM
```

- Change into the base directory and view the installation instructions:

```
$ cd OpenParEM
$ more INSTALL
```

- Set the PATH environment variable. See the INSTALL file for details.
- Install Python scripts for FreeCAD and ParaView:

```
$ cd scripts
$ OpenParEM_build --without-all
```

  The `--without-all` option installs the scripts while skipping all compilation.
- Test:

```
$ OpenParEM_test
```

- Installation is complete if both tests report `pass`.

3 Compiling from Source

OpenParEM can be compiled for Linux-based operating systems and does not require administrator priviledges if the needed build tools make, g++, gfortran, and perl are already installed. The basic steps for compilation are listed below. See the INSTALL file for more details and options.

- Download the installation package OpenParEM-*version*.tar.gz
- Unpack the package:

```
$ tar -xf OpenParEM-version.tar.gz
$ ln -s OpenParEM-version OpenParEM
```

- Change into the base directory and view the installation instructions:

```
$ cd OpenParEM
$ more INSTALL
```

- Install make, g++, gfortran, and/or perl as needed.
- Change into the scripts directory and compile:

```
$ cd scripts
$ ./OpenParEM_build --jobs=4 --with-all
```

- On successful compilation, OpenParEM2D and OpenParEM3D are tested. If both tests report `pass`, then compilation is complete.
- Set the PATH environment variable. See the INSTALL file for details.
- Verify that PATH has been properly set by re-running the tests:

  ```
  $ OpenParEM_test
  ```

- If the two tests report `pass`, then installation is complete.

4 Flow Tools

OpenParEM is a command-line tool for running electromagnetic simulations only. Pre- and post-processing must be handled by other tools.

4.1 FreeCAD

The geometry can be built with any number of different CAD tools, with FreeCAD [1] being a free and open-source option that works well. FreeCAD is very powerful, but it does have a significant learning curve that pays off as it is mastered. FreeCAD is scriptable with Python, enabling the development of helper scripts to simplify the setup of OpenParEM projects.

The easiest installation process is to install FreeCAD from the software management tool provided by the Linux distribution. Open the tool and search for FreeCAD, and if found, then install. If an option exists, select the FreeCAD version "An open source parametric 3D CAD modeler". However, functionality issues with macros may occur, in which case installing from packages is required.

Alternatively, to install FreeCAD from packages, execute

```
$ sudo add-apt-repository --enable-source \
  ppa:freecad-maintainers/freecad-stable && \
  sudo apt-get update
$ sudo apt-get build-dep freecad
$ sudo apt-get install freecad
```

Installing from packages is fairly slow and also requires administrator priviledges..

The `OpenParEM_build` installation script installs Python scripts to greatly simplify the use of FreeCAD in building OpenParEM projects. The installed location is `$HOME/.FreeCAD/Macro`. Depending on the version of FreeCAD being used, it may be necessary to set the location for the python scripts using the "User macros location:" box in the "Execute macro window".

To run FreeCAD, execute

```
$ freecad &
```

On the first use of FreeCAD, set `View`→`Panels`→`Report View` to see messages and errors from the OpenParEM macros.

4.2 gmsh

Since OpenParEM uses the finite element method, a mesh of the geometry to be solved is required. An effective free and open-source mesher is gmsh. Output from FreeCAD can be imported into gmsh and meshed, then the mesh can be saved for use by OpenParEM. Gmsh is scriptable and callable from source, but to date, no automation with gmsh for OpenParEM2D or OpenParEM3D has been implemented.

To install gmsh, download the latest version of gmsh from `https://gmsh.info/)` to a suitable installation directory *gmsh-install-dir*, then execute

```
$ cd gmsh-install-dir
$ gunzip gmsh-4.13.1-Linux64.tgz
$ tar -xf gmsh-4.13.1-Linux64.tar
$ cd $HOME/bin
$ ln -s gmsh-install-dir/gmsh-4.13.1-Linux64/bin/gmsh gmsh
```

using the actual downloaded version. If $HOME/bin does not exist, then create it and add it to the PATH environment variable.

To run gmsh, execute

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

Note that OpenParEM only works with the msh22 format of gmsh due to library limitations.

4.3 ParaView

Both OpenParEM2D and OpenParEM3D can optionally produce outputs to enable viewing of 3D vector fields using ParaView [4], for which there is a free version. It is highly recommended to review the field plots at least initially on any new project. While OpenParEM can use arbitrarily high order finite elements, ParaView is currently limited to finite element orders of 6 and lower. As discussed in Sec. 5.12, a ParaView script is available to automate the building of plots for 2D solutions.

To install ParaView, execute

```
$ sudo apt-get install paraview
$ sudo apt-get install paraview-doc
$ sudo apt-get install python3-paraview
```

This installation process requires administrator priviledges.

The `OpenParEM_build` installation script installs a Python script to greatly simplify the viewing of fields from OpenParEM2D simulations. The installed location is `$HOME/.config/ParaView/Macros`.

To run ParaView, execute

```
$ paraview &
```

5 Helper Tools and Scripts

A number of tools and scripts have been developed to help with setting up projects, project management, code management, and validation.

5.1 checkError

This code development script scans the source code and checks error message numbers for missing or duplicate numbers. It is separately run in the OpenParEMCommon, OpenParEM2D, and OpenParEM3D src directories. Note that the error number convention is 1XXX for OpenParEMCommon, 2XXX for OpenParEM2D, and 3XXX for OpenParEM3D. There is no convention for the error number orderings, and they may change from revision to revision.

5.2 proj_search

This is a useful script for finding keywords used in setup files. The format is

```
$ proj_search has|hasnot search_term
```

When executed, the script recursively searches all files ending with ".proj" that either has the search term or does not. For example, to list the finite element order used in all projects in the directory tree, execute

```
$ proj_search has mesh.order
```

The output can be further processed using grep. To continue the example, to find all projects that use a mesh order of 6, then execute

```
$ proj_search has mesh.order | grep 6
```

5.3 builder

Builder automates the construction of some common transmission line and waveguide types to help with impedance and material studies. Outputs for use with OpenParEM2D or OpenParEM3D are selectable. For more information, see the document "builder_User_Manual.pdf".

5.4 waveguide

Waveguide is an unpolished program for calculating propagation constants and characteristic impedances for rectangular waveguide, rectangular waveguide with symmetry, partially-filled rectangular waveguide, coaxial cable, and coaxial waveguide with symmetry. The outputs form the basis for the "exact" answers for many of the cases in the OpenParEM2D regression suite. The only documentation for waveguide is the source code itself. The OpenParEM2D makefile compiles waveguide but does not install it to any location, but it can be run from the source directory or moved to a convenient location.

5.5 simplify2D

simplify2D reduces an OpenParEM2D setup file (*.proj file in the regression suites) to the bare minimum by commenting out inputs that use the default value. The goal is to help identify what is unique about a particular setup.

5.6 simplify3D

simplify3D is the 3D counterpart for OpenParEM3D to simplify2D for OpenParEM2D.

5.7 process2D and process2D.sh

Process2D post-processes OpenParEM2D results and compares against test case values to produce pass/fail evaluations. Test cases are stored in the project directory with the name *project_name*_test_cases.csv. The script process2D.sh automates the use of process2D. To run a project called "my_project" set up with the setup file "my_project.proj" against its test cases, the general command format is

```
$ process2D.sh my_project.proj N
```

where N is the number of cores (or processors) to use for parallel processing. The pass/fail evaluations are saved in the file "my_project_test_results.csv".

5.8 process3D and process3D.sh

Process3D and process3D.sh are the 3D counterparts for OpenParEM3D to process2D and process2D.sh for OpenParEM2D.

5.9 regression

regression automates the execution of the regression suite by running process2D.sh or process3D.sh on a list of projects in the file "regression_case_list.txt" in the directories `install-dir/regression/OpenParEM2D` or `install-dir/regression/OpenParEM3D` . See the "README.txt" files in these directories for more information on running regression.

5.10 OpenParEM2D_save.py

OpenParEM2D_save.py is a Python script that runs in FreeCAD to create the required modes/lines setup file. Building this file by hand is possible but error-prone. The script enables the necessary setup information to be included directly in the FreeCAD drawing, then running OpenParEM2D_save.py as a macro produces the needed setup file quickly and accurately. This script is a "must have" that cannot be overlooked when using FreeCAD. See the OpenParEM2D user's manual for specifications on its use in FreeCAD.

5.11 OpenParEM3D_save.py

OpenParEM3D_save.py is a Python script that runs in FreeCAD to create the required ports setup file. Building this file by hand is theoretically possible but is unmanageable even for near-trivial geometries. The script enables the necessary setup information to be included directly in the FreeCAD drawing, then running OpenParEM3D_save.py as a macro produces the needed setup file quickly and accurately. This script is a "must have" that cannot be overlooked when using FreeCAD. See the OpenParEM3D user's manual for specifications on its use in FreeCAD, which similar but not identical that for OpenParEM2D_save.py.

5.12 field_plot.py

Field_plot.py is a Python script that runs in ParaView to automate the construction of plots for examining the fields produced from OpenParEM2D simulations. Highly recommended.

5.13 tline3

The program tline3 is a custom solver for the cascade of three transmission lines or waveguides with two shunt capacitors at the transitions, and it is used to check matching 3D setups for accuracy. The documentation is in the source code. For examples of use, see the projects
`install-dir/regression/OpenParEM3D/WR90/straight-steps/small`
and
`install-dir/regression/OpenParEM3D/WR90/straight-steps/small-renorm`.

6 Execution

OpenParEM2D and OpenParEM3D are command-line tools executed with the following commands when parallel processing

```
$ mpirun -q --oversubscribe -np N OpenParEM2D my_project.proj
$ mpirun -q --oversubscribe -np N OpenParEM3D my_project.proj
```

and

```
$ OpenParEM2D my_project.proj
$ OpenParEM3D my_project.proj
```

when running in serial on a single core. Running on a single core is sometimes convenient when getting a new project up-and-running.

When parallel processing, the `-np` option specifies the number of cores to use for the simulation, given by `N`. The `-q` option suppresses messages from the MPI infrastructure, which are rarely needed. The `--oversubscribe` is required with OpenParEM3D when `N` is more than half the number of available cores.

The number of cores to specify, `N`, requires some consideration. If the simulations are being spread across multiple computers, it is assumed that the user knows about MPI processing considerations and no further comments are provided. For simulations on a single computer, `N` should always be set to equal to or less than the number of physical cores. For high-performance computing, there is not sufficient left-over capacity for threads to be utilized. Depending on the computer, setting `N` to the number of physical cores can (and probably will) run slower than if `N` is set to a lower number. The reason is that the available bandwidth on the CPU itself and to/from memory becomes saturated, so adding more cores causes MPI messages to be delayed due to limited bandwidth, slowing overall performance. Experimenting with the value of `N` can identify the value that produces the fastest run times for a given computer. Different computers have different optimal values of `N`. See Sec. 7 for additional discussion.

When running OpenParEM2D with `N` cores, `N` copies of OpenParEM2D are run in parallel with one instance per core. When running OpenParEM3D with `N` cores, `N` copies of OpenParEM3D are run with one instance per core, *plus* `N` copies of OpenParEM2D are run with one instance per core while ports are being solved. For some times during the simulation, 2*`N` cores are being utilized, and if `2N` > the number of physical cores, then MPI issues an error and exits. Specifying `--oversubscribe` allows processing to continue without errors. OpenParEM3D sleeps while OpenParEM2D runs, so the actual load on the computer is only slightly more than `N`.

7 Computer Considerations for MPI

Ultimately, bandwidth saturation limits the performance gains available from increasing the number of cores applied to a simulation. At some number of cores, the performance gains level out then begin to fall as the number of cores increase further. Maximizing bandwidth is the key to increasing parallel performance with MPI.

For an existing computer, the primary upgrade to improving MPI parallel performance is to fully populate all memory channels with memory sticks. The memory controller will take advantage of the additional memory channels, improving memory bandwidth and reducing run times. This works even if the additional memory capacity is not actually needed. For example, if a computer has two memory channels but only one memory stick, then 1/2 of the memory bandwidth is not being utilized, and adding a second memory stick on the other memory channel can gain significant performance benefits.

If specifying a computer for high-performance computing with MPI, target computers with the largest number of memory channels. For desktops, the number may max out at 4, while for workstations, the number of channels tend to max out at 8. A dual-processor workstation will double the number of memory channels since each processor has its own set, and MPI is smart enough to split the processing across the processors to utilize the extra memory channels. As of this writing, a high-end desktop PC will have a single processor with 4 memory channels, while a high-end workstation will have two processors with 16 memory channels. For parallel processing with MPI, such a workstation handily beats the high-end PC.

Maximum clock rate does play a role, and higher clock speeds do lead to faster run times for a given memory bandwidth. High-end desktop PCs tend to have higher clock frequencies than workstations, and for simulations using low numbers of cores, the high-end PC may produce faster run times than a high-end workstation. However, the PC saturates its bandwidth more quickly with increasing core count than the workstation due to memory bandwidth saturation, and at some number of cores, the workstation begins to produce faster run times, then it can continue driving run times down with increasing core counts until its memory bandwidth saturates. For a workstation, more memory channels trumps higher clock rates.

To recap, populate all memory channels with memory sticks to maximize bandwidth. If specifying a computer, prioritize memory channel counts over clock rates. If budget allows, go for maximum memory channels *and* maximum clock rate.

8 Change Log

Version	**Date**	Description
1.0.1	Sept. 18, 2024	Initial release
1.0.2	Sept. 25, 2024	- Fixed small memory leak in OpenParEM3D
		- Updated installation documentation for Ubuntu 22.04.5 LTS
		- Added change log to the installation documentation
1.0.3	Oct. 5, 2024	- Fixed a memory cleanup error
		- Ubuntu 24 now supported
		- Updated installation documentation for Ubuntu 24.04.1 LTS
2.0.0	Mar. 4, 2025	- Added detail for clarification
		- Rev'ed the release to 2.0.0 to align with OpenParEM3D
2.1.0	April 29, 2025	- Revamped for the merging of the OpenParEMCommon, OpenParEM2D, and OpenParEM3D into a single project
		- Updated for new build script

References

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

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

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

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

[5] `https://github.com`

[6] `https://www.virtualbox.org`