OpenParEM Installation Manual

Version 1.0

September 2024

Brian Young

Contents

1 Introduction

OpenParEM2D and OpenParEM3D can be installed as pre-compiled binaries for Ubuntu Linux or compiled from source for Linux. All development and testing has been performed for Ubuntu 20.04.2.0 LTS.

2 Tool Flow

OpenParEM2D and OpenParEM3D are 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], and ParaView [3]. The user's manuals describe in detail how to use these tools to complete projects. For FreeCAD and ParaView, Python scripts greatly simplify the effort to set up and review solutions.

The makefiles for OpenParEM2D and OpenParEM3D assume the existence of certain directories that are created during the installation of FreeCAD and ParaView. The first step to installing OpenParEM is installing the tools necessary to create the entire simulation flow and verifying the existence of the required directories.

2.1 bin and lib

Executables are installed in the user's home bin and lib directories. Check to see if the directories exist by executing

```
$ ls $HOME/bin
$ ls $HOME/lib
```

If an error message including "No such file or directory" results, then create the directory with

```
$ mkdir $HOME/bin
```

or

```
$ mkdir $HOME/lib
```

as needed.

2.2 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.

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 very slow.

OpenParEM copies macros for use in FreeCAD to the directory `$HOME/.FreeCAD/Macro`. If this directory does not exist, then create it. To run FreeCAD, execute

```
$ freecad &
```

2.3 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. To run gmsh, execute

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

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

2.4 ParaView

Both OpenParEM2D and OpenParEM3D can optionally produce outputs to enable viewing of 3D vector fields using ParaView [3], 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. 6.2.7, 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
```

OpenParEM copies macros for use in ParaView to the directory `$HOME/.config/ParaView/Macros`. If this directory does not exist, then create it. To run ParaView, execute

```
$ paraview &
```

3 OpenParEM Installation Package

After installing the flow tools using the installation steps above, then the next step to installing OpenParEM2D and OpenParEM3D is to download the installation packages from GitHub [4] in the OpenParEM project for the three repositories OpenParEMCommon, OpenParEM2D, and OpenParEM3D. These packages include pre-compiled binaries, source code, documentation, tutorials, test/example cases in automated regression suites, and accuracy demonstrations.

Create or select a location to unpack the installation packages, and this location is referred to in this document as *install-dir*. Once the installation package is placed in install-dir, then execute

```
$ cd install-dir
$ gunzip OpenParEMCommon_1.0.0.tar.gz
$ tar -xf OpenParEMCommon_1.0.0.tar
$ ln -s OpenParEMCommon_1.0.0 OpenParEMCommon
$ gunzip OpenParEM2D_1.0.0.tar.gz
$ tar -xf OpenParEM2D_1.0.0.tar
$ ln -s OpenParEM2D_1.0.0 OpenParEM2D
$ gunzip OpenParEM3D_1.0.0.tar.gz
$ tar -xf OpenParEM3D_1.0.0.tar
$ ln -s OpenParEM3D_1.0.0 OpenParEM3D
```

using the actual downloaded versions of the installation package.

After unpacking, there are three directories: OpenParEMCommon, OpenParEM2D, and OpenParEM3D. As the name suggests, OpenParEMCommon has source code common to both OpenParEM2D and OpenParEM3D along with installation documentation. The other two directories contain OpenParEM2D and OpenParEM3D. The two simulators are each standalone and have their own executables, source files, documentation, tutorials, regression suites, accuracy demonstrations, and scripts. OpenParEM3D executes OpenParEM2D during its operation.

4 Pre-compiled Binaries

If installing pre-compiled binaries, then execute

```
$ sudo apt-get install make
$ cd install-dir/OpenParEM2D/src
$ make install_pre-compiled
$ cd install-dir/OpenParEM3D/src
$ make install_pre-compiled
```

Note the underscores in the make commands. To complete installation, skip to Sec. 5.8.

5 Compiling from Source Code

In addition to the download package, OpenParEM uses several additional packages that must also be downloaded and installed. Some can be installed as pre-compiled binaries, while some require compilation. For the complete installation, it is recommended to execute the following operations in the order presented. The process take about an hour if no problems are encountered, with most time spent waiting for compiles.

5.1 Pre-compiled Packages

Execute the following commands at the command line. The apt package installer is assumed, but use the appropriate package installer for the system's flavor of Linux.

```
$ sudo apt-get install g++
$ sudo apt-get install make
$ sudo apt-get install cmake
$ sudo apt-get install libx11-dev
$ sudo apt-get install openmpi-bin
$ sudo apt-get install bison
$ sudo apt-get install flex
$ sudo apt-get install liblapacke64
$ sudo apt-get install liblapacke64-dev
$ sudo apt-get install libeigen3-dev
$ export EIGEN_DIR=/usr/include/eigen3/Eigen/src/misc
```

`EIGEN_DIR` is needed for compiling OpenParEM2D, and the given path above may need adjusting. The linked directory should include `blas.h`, `Image.h`, `Kernel.h`, `lapacke.h`, `lapacke_mangling.h`, `lapack.h`, and `RealSvd2x2.h`.

5.2 Hypre

The Hypre package must be compiled from source, which is located at `https://github.com/hypre-space/hypre`. Download `hypre-2.30.0.zip`, move the zip file to install-dir, then execute

```
$ unzip hypre-2.30.0.zip
$ rm hypre-2.30.0.zip
$ ln -s hypre-2.30.0 hypre
$ cd hypre/src
$ ./configure
$ make install
$ export HYPRE_DIR=install-dir/hypre
```

5.3 PETSc

Not all installations of PETSc are compiled with the same options. For example, a particular PETSc installation may be compiled only for real numbers, but OpenParEM requires PETSc to be compiled for complex numbers. It is recommended to compile a fresh installation of PETSc to avoid compatibility issues.

The PETSc download package is available at `https://ftp.mcs.anl.gov/pub/petsc/release-snapshots/`. Download `petsc-3.19.6.tar.gz` and place it in install-dir, then execute

```
$ gunzip petsc-3.19.6.tar.gz
$ tar -xf petsc-3.19.6.tar
$ rm petsc-3.19.6.tar
$ ln -s petsc-3.19.6 petsc
$ cd petsc
$ ./configure PETSC_ARCH=linux_complex --with-clean --with-shared-libraries=0 \
  --with-debugging=0 --with-single-library=0 --with-scalar-type=complex \
  --with-precision=double --with-64-bit-indices --download-mumps --download-scalapack \
  --download-ptscotch --download-fblaslapack=yes --with-fortran-bindings=1 \
  --COPTFLAGS="-O3" --CXXOPTFLAGS="-O3" --FOPTFLAGS="-O3"
```

The configure script outputs two make commands that must be executed to complete the compilation and verification. Execute the make commands provided by PETSc. Complete the PETSc operations by exporting two variables as follows

```
$ export PETSC_DIR=install-dir/petsc
$ export PETSC_ARCH=linux_complex
```

5.4 SLEPc

The compilation of SLEPc closely follows that of PETSc and uses the compilation options set there. The SLEPc download package is available at `https://slepc.upv.es/download/`. Download the version with the same version number as that downloaded for PETSc, or if not available, the closest lower version. Place the downloaded version in install-dir, then execute

```
$ gunzip slepc-3.19.2.tar.gz
$ tar -xf slepc-3.19.2.tar
$ rm slepc-3.19.2.tar
$ ln -s slepc-3.19.2 slepc
$ export SLEPC_DIR=install-dir/slepc
$ cd slepc
$ ./configure
```

Similar to PETSc, execute the two make commands printed to the screen.

5.5 METIS

The METIS package must be compiled from source, which is located at `http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`. If the website is not available, then download from `https://openparem.org`.

MFEM requires version 4.0 even though the latest is 5.1.0. [Note: There is a compile option in MFEM for version 5, but a trial of that did not work. If that can be made to work, then METIS can be installed as a package.] Download `metis-4.0.3.tar.gz` and place it in install-dir, then execute

```
$ gunzip metis-4.0.3.tar.gz
$ tar -xf metis-4.0.3.tar
$ ln -s metis-4.0.3 metis
$ cd metis
$ make clean
$ make   [Ignore the many warnings.]
$ export METIS_DIR=install-dir/metis
```

5.6 MFEM

The MFEM package must be compiled from source, which is located on GitHub in the MFEM project. Download version 4.7 and store the tar.gz file in install-dir, then execute

```
$ gunzip mfem-4.7.tar.gz
$ tar -xf mfem-4.7.tar
$ ln -s mfem-4.7 mfem
$ cd mfem
$ make parallel -j 4
$ export MFEM_DIR=install-dir/mfem
```

5.7 OpenParEM

OpenParEM2D and OpenParEM3D link a library created for the common code. Before either OpenParEM2D or OpenParEM3D are compiled, the library must be compiled. To compile all source for for OpenParEM, execute

```
$ cd install-dir/OpenParEMCommon/src
$ make all
$ make install
$ cd install-dir/OpenParEM2D/src
$ make all
$ make install
$ cd install-dir/OpenParEM3D/src
$ make all
$ make install
```

5.8 PATH

Edit the rc file for the system shell to add the executable location and OMP_NUM_THREADS environment variable. For example, for bash edit `$HOME/.bashrc` and add two lines as

```
PATH="$HOME/bin:$PATH"
export OMP_NUM_THREADS=1
```

Start a new terminal to pick up these changes and close the old terminal.

OMP_NUM_THREADS is for MPI processing with BLAS to prevent multi-threading conflicting with MPI operation and dramatic reduction in performance. For details on over-subscribing resources, see `https://petsc.org/release/docs/manual/blas-lapack/`. When running OpenParEM2D or OpenParEM3D, if the load percentage per instance is >100%, then OMP_NUM_THREADS needs to be set.

The environment variables set with the export statements are set only for the current terminal. For future re-compiles of OpenParEM in new terminals, it is very convenient to put the export statements into the bashrc file.

5.9 Testing

One or more cases from the regression suites can be run to verify operation. Using a straight section of WR90 waveguide as a test, execute

```
$ cd install-dir/OpenParEM3D/regression/WR90/straight
$ process3D.sh straight.proj 2
```

If there are no issues, OpenParEM3D will run while executing OpenParEM2D as needed. Verify in the output that there are no messages including "ERROR" or "ASSERT". Assuming the job runs to completion, the "Job Complete" message and elapsed time are displayed. Verify that the file "straight_test_results.csv" has an appropriate time stamp showing that it was just generated by OpenParEM3D. If all looks good, then execute

```
$ more straight_test_results.csv
```

and verify that all test cases report "pass". It is acceptable if a test case just barely fails the cutoff criteria since these are intentionally set tight. If the test results file looks good, then installation is complete.

As discussed in Sec. 5.8, it is critical that the OMP_NUM_THREADS environment variable is set. This can be checked by running top in a second terminal while OpenParEM3D is running. In a first terminal, execute

```
$ top
```

and in a second terminal, execute

```
$ cd install-dir/OpenParEM3D/regression/WR90/straight
$ process3D.sh straight.proj 2
```

While the process3D.sh script runs OpenParEM3D, line entries in the first terminal will show two instances of OpenParEM3D. Check that the %CPU is 100% or less for both instances during execution. If the percentage is 200% for either instance of OpenParEM3D, then the OMP_NUM_THREADS environment variable needs to be set.

6 Helper Tools

A number of tools and scripts have been developed to help with setting up projects, project management, code management, and validation. These tools are listed grouped by the directory where they reside, so OpenParEMCommon, OpenParEM2D, or OpenParEM3D.

6.1 OpenParEMCommon

6.1.1 checkErrors.sh

This code development script scans the source code and lists sorted error message numbers along with any 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. The script "listErrors" is called by checkErrors.sh.

6.1.2 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
```

6.2 OpenParEM2D

6.2.1 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".

6.2.2 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.

6.2.3 simplify

Simplify 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.

6.2.4 process and process.sh

Process 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 process.sh automates the use of process. 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

```
$ process.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".

6.2.5 regression.sh

Regression.sh automates the execution of the regression suite by running process.sh on a list of projects in the file "regression_case_list.txt" in the directory `install-dir/OpenParEM2D/regression/`. See "README.txt" for more information on running regression.sh.

The very simple checkBuilder.sh script compares output from builder to saved "golden" outputs to verify operation of builder. It is run as part of the regression suite from regression.sh.

6.2.6 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.

6.2.7 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.

6.3 OpenParEM3D

6.3.1 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/OpenParEM3D/regression/WR90/straight-steps/small`
and
`install-dir/OpenParEM3D/regression/WR90/straight-steps/small-renorm`.

6.3.2 simplify3D

Simplify reduces an OpenParEM3D 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.

6.3.3 process3D and process3D.sh

Process3D and process3D.sh are the 3D counterparts for OpenParEM3D to process and process.sh for OpenParEM2D. See Sec. 6.2.4 for use.

6.3.4 regression3D.sh

Like regression.sh for OpenParEM2D, regression3D.sh automates the execution of the regression suite for OpenParEM3D by running process3D.sh on a list of projects in the file ”regression_case_list.txt” in the directory `install-dir/OpenParEM3D/regression/`. See ”README.txt” for more information on running regression3D.sh.

6.3.5 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.

7 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. 9 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 2`N` > 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`.

8 Virtual Execution

OpenParEM is developed on Linux, specifically Ubuntu 20.04.2.0 LTS. For those without access to Linux, an alternative to installing Linux in a dual-boot configuration or on a second computer is to use VirtualBox [5] to install and run Ubuntu 20.04.2.0 LTS virtually on an existing computer running another operating system. VirtualBox keeps the host and virtual operating systems isolated from each other. The regression case from Sec. 5.9 run using 12 cores in VirtualBox shows about a $5\times$ increase in run time over running native in Linux. Based on limited testing, virtual execution of OpenParEM is not recommended.

To install OpenParEM using VirtualBox, download and install VirtualBox. Next, create a virtual computer using VirtualBox. Then install Ubuntu 20.04.2.0 LTS onto the virtual computer using the installation instructions from Ubuntu. Finally, install OpenParEM using the installation instructions here, either installing the pre-compiled binaries or compiling from source.

9 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.

References

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

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

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

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

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