OpenParEM2D Installation

Overview

- There are two options for installing OpenParEM2D
  - Use the pre-compiled binaries.
  - Compile from source.
    - Compiling OpenParEM2D also requires compiling required packages.
      - Compilation of PETSc and SLEPc are required to support complex numbers and large integers.
    - With a suitable OS in place, expect the installation process to take about one hour.
- Operating system considerations
  - OpenParEM2D is developed using Ubuntu 20.04.2.0 LTS.
    - It should compile in other Linux flavors, but this has not been tested.
    - A port for Windows has not yet been developed.
  - A flexible option is to use a virtual machine (VM) to install Ubuntu 20.04.2.0 LTS. The installation is then confined to the VM so that the existing operating system (OS) installation is not disturbed in any way.
    - Installation has been tested on a VM installed with Ubuntu 20.04.2.0 LTS using Oracle VM Virtual Box Version 6.1.34 r150636 (Qt5.12.8).
  - The installation is tested on a fresh, minimally configured VM with the steps executed in the order given.

Installation - Pre-compiled Binaries

- https://github.com/OpenParEM/OpenParEM2D
- Click the latest release.
  - Currently only Linux is supported, and the binaries are tested on Ubuntu 20.04.2.0 LTS.
- Click the desired compression format for the download.
- Move the downloaded file to a suitable project directory.
- Unpack the file.
- Two options at this point
  - Set your PATH to include the script and bin directories.
  - Copy the contents of the script and bin directories to your local bin directory.
- This completes installation. Skip to “Testing the Installation” slide.

Installation - Compilation

- https://github.com/OpenParEM/OpenParEM2D
- Click the latest release
  - Currently only Linux is supported, and the binaries are tested on Ubuntu 20.04.2.0 LTS.
- Click the desired compression format for the download
- Move the downloaded file to a suitable project directory
- Unpack the file

Packages

- For apt-get operations, answer “Y” when asked.
- The exports are critical. It is worthwhile to make a note of all exports so that they can be reset later [in a new terminal] if OpenParEM2D is to be re-compiled.
- bash is assumed as the shell

```
$ sudo apt-get install g++
$ sudo apt-get install make
$ sudo apt-get install libopenmpi-dev
$ 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 for compiling OpenParEM2D. This may need adjusting. The linked directory should include blas.h, Image.h, Kernel.h, lapacke.h, lapacke_mangling.h ,lapack.h ,RealSvd2x2.h.

Hypre

https://github.com/hypre-space/hypre

click "master->"tags"->hypre-2.22.0->Code->Download ZIP
Move the zip file to a folder that will be the main installation folder for everything.

```
$ unzip hypre-2.22.0.zip
$ rm hypre-2.22.0.zip
$ ln -s hypre-2.22.0 hypre
$ cd hypre/src
$ ./configure
$ make install
$ export HYPRE_DIR=set to the directory linked to hypre [used for compiling OpenParEM2D]
```

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. It is recommended to compile a fresh installation of PETSc to avoid compatibility issues.

https://ftp.mcs.anl.gov/pub/petsc/release-snapshots/

Download petsc-3.16.0.tar.gz. Move the gz file to the main installation folder set up for the Hypre install.

```
$ gunzip petsc-3.16.0.tar.gz
$ tar -xf petsc-3.16.0.tar
$ rm petsc-3.16.0.tar
$ ln -s petsc-3.16.0 petsc
$ cd petsc
```

Execute one of the three following configure statements.

1. With debugging on and with shared libraries [for general development with debugging including PETSc code]:

```
$ ./configure --with-clean --with-shared-libraries=1 --with-debugging=1 --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="-g -O"  --CXXOPTFLAGS="-g -O" --FOPTFLAGS="-g -O"
```

2. With debugging off and with shared libraries [for general development with debugging excluding PETSc code]:

```
$ ./configure --with-clean --with-shared-libraries=1 --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"
```

3. With debugging off and with static libraries [for distributing a pre-compiled binary and for 2X speed improvement]:

```
$ ./configure --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"
```

After some configuration, execute the make command printed to the screen, then execute a second make command after compiling.

`$ export PETSC_DIR`=set to the directory linked to petsc [used for compiling SLEPc and OpenParEM2D]

`$ export PETSC_ARCH`=set to the sub-directory name of PETSc that names the architecture [looks something like "arch-linux-c-debug"]

SLEPc

The compilation of SLEPc tracks that of PETSc, so the options selected there apply here.

https://slepc.upv.es/download/

Download slepc-3.16.3.tar.gz. Move the gz file to the main installation folder set up for the Hypre install.

```
$ gunzip slepc-3.16.3.tar.gz
$ tar -xf slepc-3.16.3.tar
$ rm slepc-3.16.3.tar
$ ln -s slepc-3.16.3 slepc
$ export SLEPC_DIR=set to the directory linked to slepc
$ cd slepc
$ ./configure
```

Execute the make commands printed to the screen [there are two, similar to PETSc].

METIS

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. Move the gz file to the main installation folder set up for the Hypre install.

```
$ 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=set to the directory linked to metis [used for compiling OpenParEM2D]
```

MFEM

https://mfem.org

Download mfem-4.3.tgz. Move the gz file to the main installation folder set up for the Hypre install.

```
$ gunzip mfem-4.3.tgz
$ tar -xf mfem-4.3.tar
$ rm mfem-4.3.tar
$ ln -s mfem-4.3 mfem
$ cd mfem
$ make config
```

To build with debug symbols:

```
$ make debug
$ make pdebug
```

To build without debug symbols:

```
$ make all
$ make parallel
```

`$ export MFEM_DIR=`set to the directory linked to mfem [used for compiling OpenParEM2D]

OpenParEM2D

If $HOME/bin does not exist, then

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

```
https://openparem.org/
```

Click the link under the “Source” block, https://github.com/OpenParEM/OpenParEM2D. Under Releases, click the latest release, then click the compressed source code file of your choice. Move the gz file to the main installation folder set up for the Hypre install.

```
$ gunzip OpenParEM2D-1.1.1.tar.gz
```
[or the specific version being compiled]

```
$ tar -xf OpenParEM2D-1.1.1.tar
$ rm OpenParEM2D-1.1.1.tar
$ ln -s OpenParEM2D-1.1.1 openparem2D
$ cd openparem2D/src
$ make all DEBUG=x
```
[x=0 (default) for standalone binary build, and x=1 for a build using shared libraries]

If this linker error occurs:

```
$ /usr/bin/ld: cannot find -lX11
```

then a link is missing in the Ubuntu installation pointing at the library. To fix this, reinstall the X11 library:

```
$ sudo apt-get --reinstall install libx11-6
```

then re-execute $ make all DEBUG=x

```
$ make install
```

Edit $HOME/.bashrc and add two lines at the bottom:

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

Start a new terminal to pick up the changes.

`OMP_NUM_THREADS` is for MPI processing with BLAS to prevent multi-theading conflicting with MPI operation. See https://petsc.org/release/docs/manual/blas-lapack/ on over-subscribing resources. If the load percentage per instance is > 100%, then this needs to be set.

This completes the installation of OpenParEM2D.

Testing the Installation

One or more or even all of the test cases in the regression suite can be run to verify proper operation of OpenParEM2D.

```
$ cd openparem2D/regression
```

View README.txt.

Example

```
$ cd WR90_rectangular_waveguide/WR90/WR90_order_3_no_refinement_no_mesh_reuse

$ process.sh WR90.proj 5
```

After the job completes, check the log and results files to verify errors. Waive small errors.

If an error is generated about an insufficient number of processors, either increase the number of processors in the VM [if using a VM], or decrease the number 5 down to something that works or even down to 1.

To run the job without doing the accuracy check, the command is

```
$ mpirun -np 5 OpenParEM2D WR90.proj
```

The regression suite also forms an extensive set of examples. Each case is set up to run using the above command substituting the name of the project file in the directory. To view the fields with ParaView, edit the *.proj file and set “project.save.fields” to “true”.

builder

The front-end tool “builder” is also compiled and installed along with OpenParEM2D.

Many of the projects in the regression directory include a subdirectory named “builder” or something similar. These are ready to run. All of the directories name the input file to builder as “builder.txt”.

Example

```
$ cd regression/differential_pair/diff_pair_modal_symmetry_odd/builder

$ rm diffPair.geo diffPair_modes.txt diffPair.msh   [To avoid stale files. Not really required.]

$ mv diffPair.proj diffPair.proj.orig  [To prevent over-writing project-specific setup
information.]

$ builder builder.txt

$ cp diffPair.proj.orig diffPair.proj

$ gmsh -format msh22 &
```

Read the diffPair.geo file into gmsh, create a 2D mesh, then save the mesh.

```
$ mpirun -np 5 OpenParEM2D diffPair.proj
```

Other Tools

The flow described in the user manual for OpenParEM2D uses several open-source tools to prepare data for simulation and to visualize fields. Three additional tools need to be installed: FreeCAD, gmsh, and ParaView.

FreeCAD

Used to draw the geometry of the waveguide or transmission line and to setup boundary conditions and modes. The outputs from FreeCAD are inputs to gmsh and OpenParEM2D. FreeCAD is not required if geometry construction is exclusively done using builder.

https://www.freecadweb.org/

```
$ 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

$ cd $HOME/.FreeCAD
$ mkdir Macro
$ cd Macro
$ cp openparem2D/scripts/OpenParEM2D_save.py $HOME/.FreeCAD/Macro
```

To execute:

```
$ freecad &
```

gmsh

Used to generate the mesh for finite element analysis. The output from gmsh is an input to OpenParEM2D.

https://gmsh.info/

Download gmsh-4.8.4-Linux64.tgz.

```
$ gunzip gmsh-4.8.4-Linux64.tgz
$ tar -xf gmsh-4.8.4-Linux64.tar
$ ln -s gmsh-4.8.4-Linux64 gmsh
$ cd /usr/bin
$ sudo ln -s /home/briany/Desktop/gmsh/bin/gmsh gmsh
```

To execute:

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

ParaView

Used to view the computed electric and magnetic fields along with the Poynting vector. The output from OpenParEM2D is an input to ParaView.

https://www.paraview.org/

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

If the directory “Macros” is not present in $HOME/.config/ParaView/, then create it.

```
$ cp openparem/scripts/field_plot.py $HOME/.config/ParaView/Macros
```

To execute:

```
$ paraview &
```