OpenParEM2D

Volunteering

v.1.1

Overview

- There are many opportunities to help with the improvement and strengthening of OpenParEM2D.
- For the sake of organization, suggested areas of work are organized in level of difficulty, starting at Level 1.
- The suggested tasks are not intended to be comprehensive or to imply that other areas of work would not be beneficial or appreciated.

Level 1 Tasks

- Improve the user documentation.
- Use OpenParEM2D, document bugs or areas for improvement, and file good bug reports.
- Develop test cases where analytical results are available to add to the regression suite.
- Find strong published results and develop matching test cases to add to the regression suite.
- Test on different computers and microprocessors to see if there are models that do not saturate as quickly for improved parallel performance.
- Improve the algorithm for automatic mesh refinement in Mode::ZZrefineMesh in fem2D.cpp.
- Improve and/or add scripts for automation with FreeCAD and gmsh.
- Improve and/or take ownership of the project web page.
- Improve and/or take ownership of a project GitHub site.

Level 2 Tasks

- Write scripts to automate installation to greatly simplify the process of getting new users up-and-running.
  - GNU Autotools or Cmake, for example
  - DEB package
- Port to Windows
  - It would be best if Autotools or CMake was in place first.
- Write a GUI to simplify operation.
  - Many options: standalone GUI and integration into FreeCAD are two ideas
- Expand the front-end tool “builder” to add more transmission line or waveguide types.
- Update the parsers to support units.
  - For example, allow 10 GHz vs. requiring 10e9 as a frequency input.
  - Robust parsing takes a lot of work.
    - Continuing the frequency example, all of these inputs have to work: 10000000000, 10e9, 10e9 Hz, 10e9Hz, 10GHz, 10000 MHz, etc.

Level 3 Tasks

- Write a custom integrator to enable adaptive mesh refinement on the total electric field. See Mode::ZZrefineMesh in fem2D.cpp.
- Modify the methodology to solve impedance boundary conditions as part of the eigenvalue problem.
  - This converts the eigenvalue problem from a generalized one to a polynomial one.
  - SLEPc does support polynomial eigenvalue problems, so the eigenvalue solver package probably does not need to change.
- Modify the methodology to eliminate the null space solutions.
  - This would avoid the issue of having to solve for more modes than what is actually needed.
  - Lee has a later paper that covers this, so needed changes may or may not be extensive.