OpenParEM2D
Specifications

Specifications

OpenParEM2D is a command line tool that uses text files for all inputs. There are four required input files, and each has a specification.

All numbers are entered as MKS units without qualifiers. For example, 1 GHz is entered as 1e9.

The four input files with specifications are:

- project
- boundary conditions and modes
- materials
- mesh

Project File Specification

- The project file is a text file that uses keyword/value pairs to control the simulation.
- Use one keyword/value pair per line.
- Each keyword has a default value, so every keyword/value pair is optional.
- Keywords can appear in the file more than once, and excepting frequency.plan.* and field.point, the last entry is the one that is used.
  - Note: No error message or warning is issued when keyword values are overwritten.
- The first line of the file must be

  #OpenParEM2Dproject 1.0

Project File Keyword/Value Pairs

```
#OpenParEM2Dproject 1.0                 // required on first line
project.save.fields             bool    // Saves the files needed for ParaView to plot the fields.
mesh.file                       string  // File name for the mesh file.
mesh.order                      int     // Order of the finite elements used in the mesh.
                                        // 1 → linear, 2 → quadratic, etc.
mesh.refinement.cutoff          double  // mesh elements less than mesh.refinement.cutoff x max mesh error
                                        // are not refined.
mesh.refinement.fraction        double  // maximum fraction of the mesh to refine at each iteration
mesh.uniform_refinement.count   int     // The mesh is uniformly subdivided across all mesh elements
                                        // by the provided number.
mode.definition.file            string  // File name for the boundary condition/mode file.  Applies to both modal and
                                        // line style of impedance calculation.  See the specification for this file.
materials.global.path           string  // Path to a global materials file serving as a library.
                                        // May also leave blank or use ./ for the local directory.
materials.global.name           string  // File name for the global materials file.
                                        // See the specification for this file.
materials.local.path            string  // Path to a local materials file serving as a library.
                                        // May also leave blank or use ./ for the local directory.
materials.local.name            string  // File name for the local materials file.
                                        // See the specification for this file.
materials.check.limits          bool    // Check the material values against range limits.
                                        // Set to false if the range limits are not appropriate.
refinement.frequency            string  // Sets 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.variable             string   // Sets the variable used to determine refinement convergence
                                         // Options: alpha, beta, |gamma|, |Zo|, Re(Zo), Im(Zo)
                                         // A comma-separated list can be provided with the refinement variable
                                         // set per mode in order, with the final entry setting the refinement
                                         // variable for following modes.  Examples:
                                         //    refinement.variable |Zo|             - refine all modes on |Zo|
                                         //    refinement.variable |Zo|,|gamma|     - refine mode 1 on |Zo| and
                                         //                                          all other modes on |gamma|
                                         //    refinement.variable beta,alpha,|Zo|  - refine mode 1 on beta, mode 2
                                         //                                          on alpha, and all others on |Zo|
refinement.iteration.min        int      // Minimum number of iterations to perform.
refinement.iteration.max        int      // Maximum number of iterations to perform.
refinement.required.passes      int      // The number of consecutive iterations that must meet the refinement
                                         // tolerance.
refinement.tolerance            double   // Tolerance for convergence during adaptive refinement.
frequency.plan.log              string   // Adds solution frequencies to the frequency plan using a log scale with
                                         // the comma-separated list start,stop,pointsPerDecade.
frequency.plan.log.refine       string   // Same as above plus mesh refinement is applied at these frequencies.
frequency.plan.linear           string   // Adds solution frequencies to the frequency plan using a linear scale with
                                         // the comma-separated list start,stop,step.
frequency.plan.linear.refine    string   // Same as above plus mesh refinement is applied at these frequencies.
frequency.plan.point            double   // Adds the given solution frequency to the frequency plan.
frequency.plan.point.refine     double   // Same as above plus mesh refinement is applied at the given frequency.

// Any number of frequency.plan.* lines can be specified.
// Overlapping frequencies are ok because duplicates are omitted.
// Mesh refinement is done in the order that the refinement frequencies are specified.

solution.modes                  int      // number of modes to solve
solution.temperature            double   // Solution temperature, which is used for materials selection.
solution.tolerance              double   // Tolerance for the eigenvalue solution and H field calculation.
                                         // Can be set as a comma-separated list to provide values per mode.
                                         // See refinement.variable on how the values are applied to the modes.
solution.iteration.limit        int      // Iteration limit for the iterative eigenvalue and Hfield solvers.
solution.modes.buffer           int      // Additional number of modes over solution.modes to solve.
                                         // Increase this value if not all of the solution.modes solutions are found.
```

```
solution.impedance.definition   string   // Sets the definition to use for the characteristic impedance calculation.
                                         // Options: VI, PV, PI, or none, where none omits the calculation.
solution.impedance.calculation  string   // Sets the style of the setup for the characteristic impedance calculation.
                                         // Options: modal,line
solution.check.closed.loop      bool     // Check if current loops are closed.
solution.accurate.residual      bool     // 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     // Sets the eigenvalue solver to use the shift-and-invert method.
solution.use.initial.guess      bool     // Use the current solution as the initial guess in the next iteration.
solution.shift.factor           double   // A multiplier of the initial guess eigenvalue.
output.show.refining.mesh       bool     // Show details about the mesh during adaptive refinement.
output.show.postprocessing      bool     // Show details about the post-processing steps after the eigenvalue solution
                                         // is found.
output.show.iterations          bool     // Show the iterations during the eigenvalue solution.
output.show.license             bool     // Show the license governing use of the software.
test.create.cases               bool     // Create test cases useful for setting up regression testing.
test.show.audit                 bool     // Show audit results when regression testing with the program called
                                         // ªprocessº.  This keyword is not used by OpenParEM2D.
test.show.detailed.cases        bool     // Show detailed information about the test cases when regression testing with
                                         // the program called ªprocessº.  This keyword is not used by OpenParEM2D.
debug.show.memory               bool     // Show memory usage at strategic times to look for memory leaks.
debug.show.project              bool     // Show the full set of project keywords and their values.
debug.show.frequency.plan       bool     // Show the full set of simulation and refinement frequencies.
debug.show.materials            bool     // Show the full set of materials from the material databases.
debug.show.mode.definitions     bool     // Show the setups used for mode definitions.
debug.show.impedance.details    bool     // Show all voltages, currents, powers, and impedance calculations.
debug.skip.solve                bool     // Perform all operations but skip the solve.
debug.tempfiles.keep            bool     // Keeps the temporary files.
field.point            double,double     // x,y coordinate at which to calculate the E and H field values.
                                         // Any number of field.point keyword/value pairs can be specified.
```

Boundary/Mode File Specification

```
// This file specifies the physical locations of boundaries and modes along with information regarding type.
// Path definitions define physical placement, then boundaries and modes use the paths to complete the setup.
// Paths can be reused across boundaries and modes for simpler setup.

// a comment

#OpenParEMmodes 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,
// and locations for boundary conditions.
// 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)      // (x,y)
   point=(double,double)      // any number of points
   ...
   point=(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.
// The physical placement of the boundary is defined by the paths.
// One path can be used if it is complete, and paths can be chained together to form more complex physical setups.
// Names must be unique within all boundary definitions.
Boundary
   name=string
   type=surface_impedance|perfect_electric_conductor|perfect_magnetic_conductor
   material=string            // required for surface impedance
   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 modes.
// Use with solution.impedance.calculation set to ªmodalº.
// One Mode/EndMode block is applied to one solved waveguide/transmission line mode.  N modes requires N blocks.
// For N modes, all mode numbers are required from 1 to N, so mode definitions are required for modes 1, 2, ..., N.
// One path can be used if it is complete, and paths can be chained together to form more complex physical setups.
// One voltage definition and/or one current definition can be provided per mode number.
// 0 is not an allowed mode number
Mode
   mode=integer
   type=voltage|current
   path=+name
   path-=name
   ...
   path=name
EndMode

// Specify any number of lines.
// Use with solution.impedance.calculate set to ªlineº.
// One Line/EndLine block is applied to each conductor.  N conductors requires N blocks.
// For N lines, all line numbers are required from 1 to N, so line definitions are required for lines 1, 2, ..., N.
// One path can be used if it is complete, and paths can be chained together to form more complex physical setups.
// One voltage definition and/or one current definition can be provided per line number.
// 0 is not an allowed line number
Line
   line=integer
   type=voltage|current
   path=+name
   path-=name
   ...
   path=name
EndLine
```

```
// The Mode/EndMode and Line/EndLine blocks are interchangeable.  Two block types are supported
// so that the block naming is coordinated with the impedance calculation type.
```

Materials File Specification

```
// This file specifies how materials are defined for use in OpenParEM2D.  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.

// OpenParEM2D allows two material files to be specified: global and local.  The global file can be a general
// materials library, and the local file can represent a specialized set of materials 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.

// a comment

#OpenParEMmaterials 1.0   // required on first real line

// All units are MKS
// meter, Ohms, Hz, S/m, Celcius

// Any number of Material blocks can be specified.

// Debye model
// dielectric only
Material
   name=text
   Temperature
      {temperature||temp||t}={double||any}       // ªanyº means that the model applies at all temperatures
      {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 supported
// extrapolation not supported
Material
   name=text
   Temperature
      {temperature||temp||t}={double||any}       // ªanyº means that the model applies at all temperatures
      Frequency
         {frequency||freq||f}={double||any}      // ªanyº means that the model applies at all frequencies
         {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
```

Mesh File Specification

- OpenParEM2D supports the gmsh mesh file version 2.2.
- See https://gmsh.info/
- The gmsh manual includes specifications of all versions of gmsh mesh formats including legacy versions like 2.2.