Omega3P Sample Inputs CW09-CW10

Table of Contents

Input Examples

A complete example for a lossless cavity

```
  ModelInfo: {
    File: dds3.ncdf                                  //mesh file. It is the file converted using acdtool
    BoundaryCondition: {                        //specify boundary conditions. The numbers here are sideset in cubit
      Magnetic: 1, 2                                //reference surfaces 1 and 2 are symmetric planes
      Electric: 3 4                                    //set reference surfaces 3 and 4 to be electric boundary
condition
      Exterior: 6                                      //surface group 6 (maybe many surfaces) is metal
   }
   SurfaceMaterial: {                               //for each metal (exterior) surface group, list the sigma values
      ReferenceNumber: 6
      Sigma: 5.8e7
    }
  }

  FiniteElement: {
     Order: 2                                          //set the finite element basis function order to be used.
     CurvedSurfaces: on
  }

  EigenSolver: {
      NumEigenvalues:     1                 //want to compute 1 mode
      FrequencyShift:                 10.e9     //the eigenfrequency of the mode should be above 10GHz
  }

  CheckPoint: {
    Action: save
    Directory: eigens                          //eigenvectors are saved out into this directory
  }

  PostProcess: {
    Toggle: off                                        //postprocess switch
    ModeFile: dds                                  //The prefix of the mode filename.
  }

  Log: thisrun.log                                 //If you want more printout logged into the file
```

Once Omega3P run is successfully completed, eignvectors are stored in subdirectory <tt>eigens</tt>. User can convert them to mode files to be visualized using paraview. The following is the command to do that:

```
  acdtool postprocess eigentomode eigens
```

A complete example about a cavity with lossy materials

```
 ModelInfo: {
  File: ./pillbox.ncdf
  BoundaryCondition: {
    Electric: 1,2,3,4
    Exterior: 6
  }
   Material : {
     Attribute: 1
     Epsilon:   1.0
     Mu:        1.0
   }
   Material : {
      Attribute: 2
      Epsilon:   1.0
      Mu:        1.0
      EpsilonImag: -0.2        //lossy material
   }
 }

  FiniteElement: {
   Order: 1
   Curved Surfaces: off
  }

  PostProcess: {
    Toggle: off
    ModeFile: mode
    SymmetryFactor: 2
  }

  EigenSolver: {
    NumEigenvalues:     2
    FrequencyShift:                 5e9
  }
```

A complete example with periodic boundary conditions

```
ModelInfo: {
File: c026ds-pbc.ncdf
  BoundaryCondition: {
    Magnetic: 1 2
    Periodic_M: 3   //master surface
    Periodic_S: 4   //slave surface, the mesh should be exactly same as those on the master surface
    Exterior: 6
    Theta: -150     //phase
  }
}

FiniteElement: {
  Order: 2
  CurvedSurfaces: on
  ScalarPotential: 1 //use A-V formulation
}

PostProcess:
{
  Toggle: on
  ModeFile: mode
  SymmetryFactor:  8.
}
EigenSolver: {
    NumEigenvalues:     1
    FrequencyShift:         10e9
}
```

A complete example with waveguide loaded cavity

```
  ModelInfo: {
    File: cell1fourth.ncdf
    BoundaryCondition: {
      Magnetic: 1,2,3,4
      Exterior: 6
      Waveguide: 7    //for each number appeared here, it should have at least one Port container later.
Absorbing and Waveguide have the same effects. Omega3P internally will figure out which BC to use.
    }
  }

  FiniteElement: {
    Order:           1
    Curved Surfaces: on
  }

  PostProcess: {
    Toggle: on
    ModeFile: test
  }

  EigenSolver: {
    NumEigenvalues:     1
    FrequencyShift:                 9.e9
  }

  CheckPoint: {
    Action: save
    Directory: eigens
  }


  Port: {
        ReferenceNumber: 7     //this number should match surface groups in waveguide boundary condition.
        Origin:     0.0, 0.0415, 0.0    //the origin of the 2D port in the 3D coordinate system
        XDirection: 1.0, 0.0,    0.0    //the x axis of the 2D port in the 3D coordinate system
        YDirection: 0.0, 0.0,   -1.0   //the y axis of the 2D port in the 3D coordinate system
        ESolver: {
                Type: Analytic              //analytic expression is used
                Mode: {
                        WaveguideType: Rectangular     //it is a rectangular waveguide
                        ModeType: TE 1 0                    //load the TE10 mode
                        A: 0.028499                            //dimension of the waveguide in x
                        B: 0.0134053                           //dimension of the waveguide in y
                 }
         }
   }
```

Load TEM mode in a coax waveguide

```
Port: {
     ReferenceNumber: 2
     Origin:     0.0, 0.0, 0.011
     ESolver: {
             Type: Analytic
             Mode: {
                     WaveguideType: Coax
                     ModeType: TEM
                     A: 0.0011  //smaller radius
                     B: 0.0033  //larger radius
             }
     }
}
```

Load TE11 mode in a circular waveguide

```
Port: {
     ReferenceNumber: 2
     Origin:     0.0, 0.0, 0.1
     XDirection: 1.0,  0.0,    0.0
     YDirection: 0.0,  1.0,    0.0
     ESolver: {
             Type: Analytic
             Mode:  {
                     Waveguide type: Circular
                     Mode type: TE 1 1
                     A: 0.03
             }
     }
}
```

Load two TE modes in the same rectangular waveguide

```
Port: {
    Reference number: 9  // FPC
    Origin: 0.0, 0.198907, -0.4479152585
    XDirection:  -1.0, 0.0, 0.0
    YDirection: 0.0, 0.0, 1.0
    ESolver:  {
           Type: Analytic
           Mode: {
                   WaveguideType: Rectangular
                   ModeType: TE 1 1
                   A: 0.1348935946
                   B: 0.024973714999999970
          }
   }
}

Port: {
    Reference number: 9  // FPC
    Origin: 0.0, 0.198907, -0.4479152585
    XDirection:  -1.0, 0.0, 0.0
    YDirection: 0.0, 0.0, 1.0
    ESolver: {
           Type: Analytic
           Mode:  {
                   WaveguideType: Rectangular
                   ModeType: TE 2  0
                   A: 0.1348935946
                   B: 0.024973714999999970
          }
     }
 }
```

Make a non-planar surface absorbing boundary

```
Port: {
  ReferenceNumber: 5              //reference surface ID
  Origin:     0.0, 0.0, 0.0       //not used
  XDirection: 1.0, 0.0,    0.0    //not used
  YDirection: 0.0, 1.0,    0.0    //not used
  ESolver: {
    Type: Analytic
    Mode:{
                     Mode number: 1
                     Waveguide type: ABC
                     Mode type: ABC
    }
  }
}
```

LinearSolver options in EigenSolver container

- The first option is that user does not provide anything. The EigenSolver container in the input file looks like:

  ```
  EigenSolver: {
    NumEigenvalues:     1
    FrequencyShift:             10.e9
    Tolerance: 1.e-8
  }
  ```

  In this case, Omega3P will use the default option for linear solver for solving shifted linear systems

- The second option is to use float version of the sparse direct solver.

```
EigenSolver: {
  NumEigenvalues:     1
  FrequencyShift:             10.e9
  Preconditioner: MUMPSFLOAT //use the float version. memory usage reduced into half.
}
```

- The third option is to use Krylov subspace method with different preconditioner.

  ```
  EigenSolver: {
    NumEigenvalues:     1
    FrequencyShift:             10.e9
    Preconditioner: MP      //this use p-version of multilevel preconditioner.
  }
  ```

  The code will choose either CG (real matrices) or GMRES (complex matrices) and the p-version
  of multilevel precondtioner as the solver for shifted linear systems.

- The fourth option is to use out-of-core sparse direct solver (an experimental feature).

  ```
  EigenSolver: {
    NumEigenvalues:     1
    FrequencyShift:             10.e9
    Memory: 1000  //if the memory usage of the matrix factor in any process is larger than 1000MBytes,
                          //switch to use out-of-core solver.
  }
  ```

FAQ

How to calculate Wallloss Quality Factor?

There are two ways to do so. Each way has its advantage and disadvantage.

1. Inside ModelInfo.BoundaryCondition define a set of boundary surfaces as Exterior.
   For each of the boundary surfaces, have a corresponding SurfaceMaterial container inside ModelInfo.
   For example:

   ```
   ModelInfo: {
    File: .dds3.ncdf

    BoundaryCondition: {
      Magnetic: 1, 2, 3, 4
      Exterior: 6   // sideset 6 is defined as Exterior BC.
    }

    SurfaceMaterial: {        // have a separate for each number in Exterior BC
      ReferenceNumber: 6  //the corresponding sideset in Exterior BC
      Sigma: 5.8e7            //electrical conductivity of the material
    }
   }
   ```

   After that, make sure you toggle the PostProcess on.

   ```
   PostProcess: {
    Toggle: on   // this should be on for computing wallloss Q
    ModeFile: ./dds
   }
   ```

   After you run omega3p with the input file, you will get a file called "output" under the same directory. Inside the file, it has a summary of results such as:

```
            Mode : {
                TotalEnergy : 4.4270939088102e-12
                QualityFactor : 6478.5096350252
                File : ./dds.l0.1.144469E+10.m0
                PowerLoss : 4.9139118623939e-05
                Frequency : 11444685657.626
            }
```

The number after QualityFactor is the one you are looking for. This method uses perturbation theory and has advantage that it is very simple. The computation associated with it is minimal.

2. Inside ModelInfo.BoundaryCondition, define the set of surfaces as Impedance (instead of Exterior in method 1).
   Set the HFormulation to be 1 (this is very important). Also, have a set of corresponding SurfaceMaterials inside ModelInfo as those in method 1.
   For example:

```
ModelInfo: {

 File: dds3.ncdf

 BoundaryCondition: {
   HFormulation: 1
   Magnetic: 1, 2, 3, 4
   Impedance: 6
 }

 SurfaceMaterial: {
   ReferenceNumber: 6
   Sigma: 5.8e7
 }
}
```

   After you run omega3p with the input, in the output file, you will see

```
        Mode = {
            TotalEnergy = { 6.2827077634198e-07, 0 },
            ExternalQ = 6579.1486638005,
            QualityFactor = inf,
            File = './dds.l0.R1.144619E+10I8.698837E+05.m0',
            PowerLoss = 0,
            Frequency = { 11446188331.641, 869883.69746227 }
        }
```

   The number after ExternalQ is the wall loss Q you are looking for. During the omega3p run, it should also print out the Q information such as

```
COMMIT MODE: 0 FREQ = (11446188331.64141,869883.6974622669)         k =
(239.8943683519209,0.01823141417003215)         Q = 6579.148663800495
```

   Note that this method set an impedance boundary condition on those surfaces and make the eigenvalue problem complex and nonlinear. It takes more time and memory to solve the problem. But the field will be in the right phase (even close to the boundary surfaces).

Both methods should give you converged Q results if mesh is dense enough.