T3P Top-Level Commands

- **ModelInfo**
- **FiniteElement**
- **PRegion**
- **LoadingInfo**
- **Loading**
- **TimeStepping**
- **Monitor**
- **LinearSolver**
- **CheckPoint**

Note: Refer to **acdtool** command syntax for postprocessing capabilities.

ModelInfo

```
ModelInfo:
{
 File: ./pillbox.ncdf

 BoundaryCondition:
 {
  Exterior: 6
  Magnetic: 1, 2
  Absorbing: 3, 4
  Impedance: 5
  Waveguide: 7
 }

 Material : {
  Attribute: 1
  Epsilon:   1.0
  Mu:        1.0
 }

 Material : {
  Attribute: 1
  Epsilon:   35.83
  Mu:        1.0
  Sigma:     0.316
}

Material : {
  Attribute: 2
  PML: {
    Type: Face
    Start: -0.01143 -0.00508 0.08
    End:    0.01143  0.00508 0.10
    Direction: 0. 0. +1.
    Reflection Coefficient:  1.e-5
    Polynomial Order: 3
    Maximum Sigma: 50.e-2 50.e-2 50.e-2
  }
}

 SurfaceMaterial: {
  ReferenceNumber: 5
  Sigma: 1.e6
  Frequency: 1.5e9
```

```
 }     // For surface impedance boundary & surface power monitor

SurfaceMaterial: {
   ReferenceNumber: 3
   Sigma: 5.8e7
   Frequency: 5e9
   Coating: {
    Type: ThinLossyDielectric
    Epsilon:   56
    Sigma: 12
    Thickness:1.e-3
   }
 }     // For surface impedance boundary with thin lossy dielectric on conductor
```

ModelInfo specifies the model used for t3p simulation.

File: The name of the mesh file in netcdf format. It can be located in a directory specified by the path. If the path is not specified, the default is the directory where the job is submitted.

BoundaryCondition specifies the boundary conditions on all the surfaces of the mesh. Every single surface of the mesh has a reference number which is set in Cubit and each reference number is associated with a boundary condition. The types of boundary condition used in t3p are

- **Electric**: The tangential component of the electric field is zero. It can be used to define a symmetry plane in the model.
- **Magnetic**: The tangential component of the magnetic field is zero. It can be used to define a symmetry plane in the model.
- **Exterior**: The tangential component of the electric field is zero. Computationally it is equivalent to **Electric** boundary condition, but used normally for perfectly conducting cavity surface. For normal conducting surface, it can be used for calculating power loss on the surface using perturbation theory.
- **Absorbing**: Absorbing boundary condition (ABC) allows electromagnetic waves propagating at the speed of light to pass through the boundary without reflection. It is used for the entry and exit planes for a beam excitation, and for coaxial waveguides.
- **Waveguide**: Broadband waveguide boundary condition to account for the dispersive propagation of electromagnetic wave in waveguides.
- **Impedance**: The electric and magnetic fields are related by the conductivity of the boundary surface for imperfect conductor; useful for self-consistent field calculation for finite loss of a conductor when the skin depth is small compared with the finite element sizes in the mesh. The conductivity is specified in **SurfaceMaterial**. The conductivity is assumed to be constant, specified at a certain frequency.

The surface impedance boundary condition can also be used to model thin lossy dielectric layer deposited on a conductor surface. One needs to specify the dielectric properties and thickness of the dielectric material.

Material specifies the material properties of a region in the model. If not specified, the region is set to vacuum.

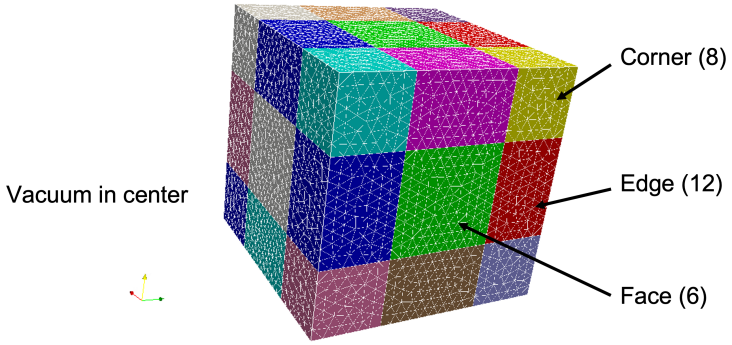
- **Attribute**: The ID of the region
- **Epsilon**: Relative permittivity of the region
- **Mu**: Relative permeability of the region
- **Sigma**: Conductivity of lossy material [unit in S/m]
- **PML**: Container to specify perfectly matched layer (see setup below)

SurfaceMaterial specifies the external wall material properties, which are used to model the power loss on the surface.

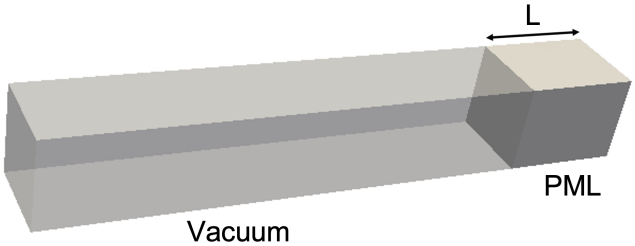
- **ReferenceNumber**: The reference number of the surface set in Cubit
- **Sigma**: Conductivity of the surface [unit in S/m]
- **Frequency**: Frequency at which conductivity is specified [unit in Hz]
- **Coating** specifies the material properties of a thin dielectric deposited on a conductor surface.
  - **Type:** Type of coating layer
    - **ThinLossyDielectric:** Thin lossy dielectric layer
  - **Epsilon:** Relative permittivity of layer
  - **Sigma:** Conductivity of layer [unit in S/m]
  - **Thickness:** Thickness of layer [unit in m]

PML terminates wave propagation by absorption in artificially lossy layers.

- Free space termination set by **Face**, **Edge** and **Corner** layers



- Waveguide termination set by **Face** layer



- PML parameters specified by the following model

$$\sigma_w(l) = \sigma_{w,\max}\left(\frac{l}{d}\right)^m, \quad \sigma_{w,\max} = -\frac{(m+1)\ln R}{2\eta_0 d}$$

where σ_w is the conductivity set by $\sigma_{w,max}$, thickness *d*, order *m* and location in layer. *R* is the reflection coefficient from the PML.

PML container specifications

- **Type**: Type of PML layer
  - **Face**: 6 orientations specified by **Direction**
  - **Edge**: 12 orientations specified by **Direction**
  - **Corner**: 8 orientations specified by **Direction**
- **Start**: x, y, z
  Lower bound coordinates (x, y, z) of layer's bounding box [units in m]
- **End**: x, y, z
  Upper bound coordinates (x, y, z) of layer's bounding box [units in m]
- **Direction:** ix, iy, iz
  Orientation of layer:
  (+/- 1, 0, 0), (0, +/- 1, 0), (0, 0, +/- 1) for **Face** layer
  (+/- 1, +/- 1, 0), (0, +/- 1, +/- 1), (+/- 1, 0, +/- 1) for **Edge** layer
  (+/- 1, +/- 1, +/- 1) for **Corner** layer
- **Reflection Coefficient**: Reflected from PML; if specified, **Maximum Sigma** will be calculated.
- **Maximum Sigma**: Conductivity in x, y, z directions [units in S/m]

FiniteElement

```
FiniteElement:
{
 Order: 1
 Curved Surfaces: on
}
```

FiniteElement specifies parameters for the finite element method.

Order: The order of the finite elements used in the simulation. Order 1-6 have been implemented. The higher the order is, the more accurate the calculation is and the more computational resources are or a fixed mesh. For most applications, using 2nd order is good enough to obtain the solution accuracy.

Curved Surfaces: **on** or **off**
on: The surfaces of the finite elements on the model surface are represented by curved surfaces to better approximate the geometry.
off: The surfaces of the finite elements on the model surface are represented by flat surfaces.

PRegion

```
FiniteElement:
{
 Order: 0              // finite element order p outside the window
 CurvedSurfaces: on
}

PRegion:
{
 Type: AutomaticMovingWindow
 Order: 2
 Back:  0.001
 Front: 0.002
 StructureEnd: 0.0689
}
```

PRegion enables the use of the moving window technique in calculating short-range wakefield. The size of the window is governed by the distances in front of and at the back of the moving bunch. Within the window, higher-order finite elements are used for the computation, and outside the window, the order of the finite elements are set to zero (as shown above) because the fields in these regions will not affect the wakefield calculation.

Type: The type of calculation in the **PRegion** is **AutomaticMovingWindow** for wakefield computation from the transit of a beam.

Order: The order of finite elements

Back: Distance from the back of the bunch to the back end of the window [unit in m]

Front: Distance from the front of the bunch to the front end of the window [unit in m]

StructureEnd: z coordinate of the end point of the model in the z direction [unit in m]

Note that the window length in the beam propagation direction is **Back** + Total length of bunch + **Front**. The beam will traverse this window until its front reaches the front end of the window, then a new window moving in the forward direction will be created. The smaller the window is, the less the computational time for each time step is. However, whenever a new window is created, t3p will assemble new matrices for solving the linear system, which will increase the computational time. Therefore, the choice of a reasonable size window can optimize the runtime performance.

LoadingInfo

```
LoadingInfo:
{
 Bunch:
 {
  Type: Gaussian
  Sigma: 0.01
  Number of sigmas: 5
  Charge: 1e-12
 }
 SymmetryFactor: 4.0
 StartPoint: 0 0 -0.10
 Direction: 0 0 1
 BoundaryID: 3
 NumBunches: 2.              // optional
 BunchSeparation: 0.1        // optional
 StartTime: 1.e-9            // optional
}

LoadingInfo:
{
 // BiGaussian bunch
 Bunch:
 {
  Type: BiGaussian
  Sigma1: 59e-6
  Sigma2: 8e-6
  Number of sigmas: 5
  Charge: 1.e-12
 }

 // Flattop bunch with half Gaussian rise and fall
Bunch:
{
 Type: Flattop
 Rise sigma: 0.002
 Number of rise sigma: 5
 Drop sigma: 0.002
 Number of drop sigma: 5
 Flattop width: 0.01
 Charge: 1.e-12
 }

 SymmetryFactor: 4.0
 StartPoint: 0 0 -0.10
```

Direction: 0 0 1
BoundaryID: 3
}

LoadingInfo specifies the information for beam excitation in t3p. For beam excitation, the following parameters for the bunch shape and its path are described. One can define multiple instances of **LoadingInfo** representing the use of multiple bunches.

Bunch: A rigid beam traverses the computational domain at a certain direction. Currently only transit in the z direction is allowed. The bunch shape is defined by the following parameters.

- **Type**: Type of bunch shape
  - **Gaussian**: symmetric Gaussian shape
  - **BiGaussian**: asymmetric Gaussian shape
  - **Flattop**: flattop with Gaussian rise and fall

For **Gaussian**, specify the following:

- **Sigma**: RMS value of bunch length [unit in m]
- **Number of sigmas**: The total length of the symmetric Gaussian bunch is defined by 2***Sigma*******Number of sigmas**.

For **BiGaussian**, specify the following:

- **Sigma1**: RMS value of bunch length [unit in m] for the front side of the asymmetric Gaussian bunch
- **Sigma2**: RMS value of bunch length [unit in m] for the back side of the asymmetric Gaussian bunch
- **Number of sigmas**: The total length of the asymmetric Gaussian bunch is **Number of sigmas** * (**Sigma1**+**Sigma2**).
- **Charge:** Total charge of the bunch [unit in C]

For **BiGaussian**, specify the following:

- **Rise sigma**: RMS value of bunch length [unit in m] for the rise Gaussian
- **Drop sigma**: RMS value of bunch length [unit in m] for the fall Gaussian
- **Number of rise sigma**: The total length of the rise Gaussian is **Number of rise sigma*Rise sigma**.
- **Number of drop sigma**: The total length of the fall Gaussian is **Number of drop sigma*Rise sigma**.
- **Flattop width**: [unit in m]
- **Charge:** Total charge of the bunch [unit in C]

SymmetryFactor: If the bunch trajectory is located on a symmetry plane, this parameter sets the correct value of **Charge** in simulation. For example, if the bunch is located on a symmetric plane with **Magnetic** boundary condition, it is set to 2. If the bunch is located at the corner of two symmetry planes with **Magnetic** boundary condition (in a quarter model), it is set to 4.

StartPoint: x, y, z

The bunch enters the computational domain at the coordinates (x, y, z) [units in m].

Direction: x, y, z
The direction of the bunch, only z direction is allowed

BoundaryID: The reference number of the surface where the bunch enters

Note that multiple bunches can be specified by using multiple instances of **Bunch**. The positions of individual bunches can be set by **StartPoint**. The time of injection of each bunch can be specified using

StartTime: The time when the bunch is injected [unit in s]. The default is zero, and it is not required to specify this parameter for a single bunch.

NumBunches: The number of bunches in a bunch train. it is not required to specify this parameter for a single bunch.

BunchSeparation: The distance between bunches in a train [units in m]

Loading

```
Loading:
{
 Type: DipoleLoading
 Coordinate: 0.0001, 0.00001, 0.
 Direction: 0.0, 0.0, 1.0
 Tolerance: 0.001
 Excitation:
 {
  Power: 1.
  Pulse:
  {
   Type: Gaussian
   Frequency: 11.0e9
   Bandwidth: 2.0e9
   DB: 20.
   T0: 5.
   Phase: 90.0
  }
 }
}

Loading:
{
 Type: DipoleLoading
 Coordinate: 0.0001, 0.00001, 0.
 Direction: 0.0, 0.0, 1.0
 Tolerance: 0.001
 Excitation:
 {
  Power: 1.
  Pulse:
  {
   Type: Monochromatic
   Frequency: 10.915e9
   Rise periods: 10
   T0: 0.
  }
 }
}

Loading:
{
 Type: PortModeLoading
```

```
// Example for loading external port mode fields from files
Port:  {
  ReferenceNumber: 3
  Origin: 0.0 0.0 -1.928
  XDirection: 1.0 0.0 0.0
  YDirection: 0.0 1.0 0.0
  ESolver: {
    Type: Interpolative
    Mode: {
      ExFile: Ex.prn
      EyFile: Ey.prn
      BxFile: Bx.prn
      ByFile: By.prn
    }
  }

// Example for calculating port modes numerically
Port:  {
  ReferenceNumber: 3
  SymmetricFactor: 4
  NumberOfModes: 1
 }

Excitation:  {
   Power: 1.
   Pulse: {
     Type: Monochromatic
     Frequency: 1.e9
     Rise periods: 10
     T0: 0.
   }
}

Loading:
   {
    Type: SurfaceLoading
    ReferenceNumber: 4

    InputFile: TM01.prn

   Excitation:  {
      Mode number: 1
      Power: 1.
      Pulse: {
        Type: Monochromatic
        Frequency: 390.2035e6
```

```
    Rise periods: 10
    T0: 0.
   }
  }
 }
```

Loading specifies the information for dipole excitation and port mode excitation in t3p. The type of loading is specified by the following.

Type: Loading type

- **DipoleLoading**: Excitation of a dipole at a point in the computational volume
- **PortModeLoading**: Field excitation at a boundary surface, specified by its location through **Port** and excitation properties through **Excitation**
- **SurfaceLoading**: Field excitation at a curved boundary surface, specified by user-defined fields and excitation properties through **Excitation**

(1) For **DipoleLoading**, specify the following parameters.

Coordinate: x, y, z
The location of the dipole at the coordinates (x, y, z) [units in m].

Direction: x, y, z
The direction of the dipole.

Tolerance: Set the tolerance that the dipole **Direction** aligns with the edge of the element closest to the dipole **Coordinate**.

Excitation: Specifies properties of excitation.

- **Power**: Electric dipole moment [unit in A-m]
- **Pulse**: Type of excitation
    - **Gaussian**: Gaussian pulse
    - **Monochromatic**: Single-frequency pulse

For **Gaussian**, specify the following:

- **Frequency**: Mean frequency [unit in Hz]
- **Bandwidth**: Pulse bandwidth [unit in Hz]
- **DB**: Cutoff in dB of the Gaussian from the peak value
- **T0**: Number of sigma's to define the Gaussian pulse length
- **Phase**: Phase for the sinusoidal oscillation at the mean frequency [unit in degree]

For **Monochromatic**, specify the following:

- **Frequency**: Drive frequency [unit in Hz]
- **Rise periods**: Number of cycles to reach steady state from initial zero value
- **Fall periods**: Number of cycles to zero value from flattop (optional)
- **T0**: Start time of pulse [unit in s]

- **TMax**: Total time of pulse [unit in s] (optional)

(2) For **PortModeLoading**, specify the following parameters.

Port: Specifies the properties of the port and loaded external fields. Note that the boundary condition is set to **Absorbing** to minimize reflection of outgoing fields at the port. Port mode fields can be imported from files for arbitrary ports or solved numerically for homogeneous waveguides. For importing fields from files, set

- **ReferenceNumber**: The reference number of the boundary port, specified when generating the mesh using Cubit
- **Origin**: x, y, z
  Coordinates of the origin of the port boundary [units in m]
- **XDirection:** x, y, z
  x direction of the port
- **YDirection:** x, y, z
  y direction of the port
- **ESolver**: Solution of the port mode
  - **Type: Interpolative**
    The fields are determined by interpolation from input data, which are formatted in 2D Cartesian coordinates
  - **Mode**: Field data of the drive fields
    - **ExFile**: File name containing the x component of the electric field
    - **EyFile**: File name containing the y component of the electric field
    - **BxFile**: File name containing the x component of the magnetic field
    - **ByFile**: File name containing the y component of the magnetic field

      The data format in these field files is (ix, iy, x, y, F). For a rectangular grid of MxN, ix is the grid ID number in the x direction (starting from 1 to M), iy the grid ID in the y direction (starting from 1 to N), x the corresponding x coordinate, y the corresponding y coordinate, and F the field value at the grid point [electric field unit in V/m, magnetic field unit in T], respectively. The data file is filled each row by advancing ix from 1 to M for a fixed iy (starting from 1), and then repeat for the next iy until iy = N .

For numerical solution of waveguide port modes, set

- **ReferenceNumber**: The reference number of the boundary port, specified when generating the mesh using Cubit
- **SymmetricFactor**: 1 for no symmetry plane; 2 for symmetry with 180^o plane; 4 for symmetry with 90^o planes
- **NumberOfModes**: Number of modes loaded at port

Excitation: See **DipoleLoading** above.

(2) For **SurfaceLoading**, specify the following parameters. User supplies a field map in a uniform (theta, phi) grid on a curved surface; Boundary condition of curved surface set to **Absorbing**.

ReferenceNumber: The reference number of the boundary curved surface, specified when generating the mesh using Cubit

InputFile: Name of input file
Specifies field map on curved surface in (theta, phi) spherical coordinates. The data format in is (ix, iy, theta, phi, Ex, Ey, Ez, Bx, By, Bz). For a (theta, phi) grid of MxN, ix is the grid ID number in the theta direction (starting from 1 to M), iy the grid ID in the phi direction (starting from 1 to N), theta the corresponding theta coordinate [unit in radian], phi the corresponding phi coordinate [unit in radian], and Ex, Ey, Ez, Bx, By, Bz the field components at the grid point [electric field unit in V/m, magnetic field unit in T], respectively. The data file is filled each row by advancing ix from 1 to M for a fixed iy (starting from 1), and then repeat for the next iy until iy = N.

Excitation: See **DipoleLoading** above.

TimeStepping

```
TimeStepping:
{
  MaximumTime: 3.0e-9
  DT: 1e-12
}
```

TimeStepping sets the total time and the time step of t3p simulation.

MaximumTime: Total simulation time [unit in s]

DT: Time step for advancing the Maxwell equation numerically in the time domain [unit in s]

Monitor

```
Monitor:
{
 Type: Point
 Name: mon1
 Coordinate: 0.01, 0.00, 0.001
}

Monitor:
{
 Type: Volume
 Name: mymon
 TimeStart: 0e-9
 TimeEnd:   3e-9
 TimeStep:  2.5e-11
}

Monitor:
{
 Type: WakeField
 Name: wakefield
 StartContour: -0.03
 EndContour:  0.03
 Smax: 0.5

// Use the following only for direct integration for collimator-type structures
 Grid:
 {
   Method: Circle
   Radius: 0.0149
   NPoints: 10
   StartDense: 0
   EndDense:   90
   Fraction:   1
 }
}

Monitor:
{
 Type: Power
 ReferenceNumber: 5
 Name: port
}

Monitor:
```

```
{
 Type: SurfacePowerLoss
  ReferenceNumber: 3
  Name: wallLoss
 }

Monitor:
{
 Type: ModeVoltage
 Name: modecoeff
 Port: {
  ReferenceNumber: 5

  ESolver: {
   Type: Numerical2D
   NumberOfModes: 1
   Frequency: 1.e9
   Tolerance: 1.e-15
   MaxIterations: 1000
   }
 }
}
```

Monitor provides diagnostics for the electromagnetic field in a time domain simulation. Several types of monitors have been implemented in t3p, namely, **Point**, **Volume**, **WakeField**, **Power** and **ModeVoltage**.

Type: Type of monitor

- **Point**: Monitors the fields at a location as a function of time. The output is stored in an ascii file with the format (t Hx Hy Hz Ex Ey Ez) with units in SI.
- **Volume**: Dumps snapshots of volumetric fields. The output is stored in netcdf format which can be visualized using ParaView.
- **WakeField**: Calculates the wakefield due to beam excitation.
- **Power**: Calculates the power at a boundary port.
- SurfacePowerLoss: Calculates the power loss on a lossy surface defined by a reference number,.
- **ModeVoltage**: Calculates the voltage of waveguide modes at a boundary port.

(1) For **Point** monitor, specify the following.

- **Name**: The name of the file storing the fields as a function of time. The output is stored in an ascii file with the format (t Hx Hy Hz Ex Ey Ez), where the units are SI.
- **Coordinate**: x, y, z
  Location where the fields are monitored. The coordinates (x, y, z) are in untis of m.

(2) For **Volume** monitor, specify the following.

- **Name**: The name of the files storing the snapshots of fields at a regular interval.
- **TimeStart**: Start time for dumping fields [unit in s]
- **TimeEnd**: End time for dumping fields [unit in s]
- **TimeStep**: Time interval for dumping fields [unit in s]

(3) For **Wakefield** monitor, specify the following.

- **Name**: The name of the file storing the wakefield as a function of s, which is the distance from the front of the bunch and greater than 0. The output is stored in an ascii file with the format (s W(s), I(s)), where s is the distance in m, W(s) wakefield in V/C, and I(s) the bunch density in A/m, respectively.
- **StartContour**: The position [unit in m] where wakefield integration starts. If not specified, it is set to the beginning of the model in the z direction.
- **EndContour**: The position [unit in m] where wakefield integration ends. If not specifies, it is set to the end of the model in the z direction.
- **Smax**: The maximum distance for which the wakefield is calculated. It cannot be larger than that set by **MaximumTime**, the total bunch length and the model geometry to ensure the wakefield catch up with the beam. It is determined as follows.
  Let L = total model length and l = length of downstream beampipe, then **Smax** = c***MaximumTime** – (L-l).

Note that the wakefield integration used in this monitor is the Weiland wakefiled integration. It only applies to structures where on can see only vacuum from the cross section of the upstream and downstream beampipe. The integration can be reduced along the vacuum gap (specified by **StartContour** and **EndContour**) along the beampipe boundary. If **StartContour** and **EndContour** are not specified, they will be set to the start and end of the model in the z direction.

The wakefield will be automatically calculated at the transverse coordinates specified by (x, y) in **Bunch**: **StartPoint**. For other transverse coordinates, the wakefield can be calculated using acdtool as follows.

acdtool postprocess wake_new t3p_results <x1, y1>

where *<x1, y1>* are the transverse coordinates in units of m. Because the Laplace solver is mesh-based, it returns the wakefield at a mesh point closed to *<x1, y1>* indicated in the wakefield output file. Therefore, it is advantageous to use *<x1, y1>* on the grid. The output file is stored as t3p_results/OUTPUT/wakefield.out, where the file name "wakefield" has been specified in **Monitor**.

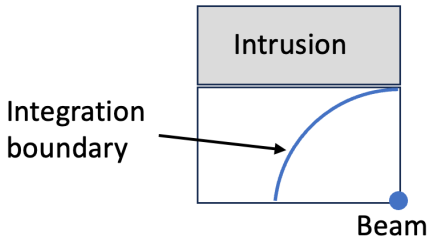
The transverse wakefield is obtained through the Panofsky-Wenzel theorem by taking the derivative of the integration of the longitudinal wakefield. One needs to calculate the longitudinal wakefield at two different transverse coordinates by

exciting the beam at an offset position. For example, in calculating the transverse wakefield in the x-direction, the longitudinal wakefields at two sets of transverse coordinates (x1, 0) and (x2, 0) need to be calculated. The wakefield can be calculated using acdtool as follows.

acdtool postprocess transwake t3p_results <x1, y1> <x2, y2>

where *<x1, y1>* and *<x2, y2>* are the transverse coordinates in units of m. Because the Laplace solver is mesh-based, it returns the wakefield at a mesh point closed to *<x1, y1>* and *<x2, y2>* indicated in the wakefield output file. Therefore, it is advantageous to use *<x1, y1>* and *<x2, y2>* on the grid. The output file is stored as t3p_results/OUTPUT/wakefield.out.

For collimator-type structure, direct wakefield integration can be used along a circular boundary within the vacuum chamber (see figure below). A long enough downstream catchup distance is required to obtain convergence.



Specify the following to define direct integration.

- **Grid**: The container to invoke direct integration. Do not use it for indirect integration using Weiland method
  - **Method: Circle** – Use circle to define the integration boundary.
  - **Radius**: Radius of circle [unit in m]
  - **NPoints**: Number of integration intervals on the circle boundary
  - **StartDense**: Start angle of circle [unit in deg]
  - **EndDense**: End angle of circle [unit in deg]
  - **Fraction: 1** – Use this default.

acdtool postprocess wake_direct t3p_results <x1, y1> can be used to calculate the longitudinal wakfield inside the circle.

(4) For **Power** monitor, specify the following.

- **Name**: The name of the file storing the power as a function of time. The output is stored in an ascii file with the format of time [unit in s] and power [unit in W].
- **ReferenceNumber:** The reference number of the surface where power is monitored
- **TimeStart**: Start time for writing power to file [unit in s]
- **TimeEnd**: End time for writing power to file [unit in s]

- **TimeStep**: Time interval for writing power to file [unit in s]

(5) For **SurfacePowerloss** monitor, specify surface material properties in **ModelInfo** and the following.

- **Name**: The name of the file storing the power as a function of time. The output is stored in an ascii file with the format of time [unit in s] and power [unit in W].
- **ReferenceNumber:** The reference number of the surface where power loss is monitored

(6) For **ModeVoltage** monitor, specify the following.

- **Name**: The name of the file storing the generalized voltage of a waveguide mode as a function of time. The output is stored in an ascii file with the format of time [unit in s] and voltage [unit in V].

Port: Specifies the properties of the port and solution of the waveguide modes.

- **ReferenceNumber:** The reference number of the boundary port, specified when generating the mesh using Cubit
- **ESolver**: Solution of the port mode
    - **Type: Numerical**: The fields are solved by a 2-dimenional eigensolver.
    - **NumberOfModes**: Number of waveguide modes to be solved
    - **Frequency**: Specify the frequency [unit in Hz] where the mode solutions are computed.
    - **Tolerance**: Tolerance for solution convergence
    - **MaxIterations**: Maximum number of iterations allowed for solution convergence

LinearSolver

```
LinearSolver: {
  Solver:            MUMPS
}

LinearSolver: {
 Solver:           CG
 Preconditioner:     CHOLESKY
}
```

LinearSolver specifies the solver for solving the linear system used in t3p simulation.

MUMPS is a direct solver and used for small problems.

CG is an iterative solver and used for large problems. **Preconditioner** for the iterative solver includes **CHOLESKY** and **DIAGONAL**.

CheckPoint

```
CheckPoint:
{
 Action: restart
 Ntimesteps: 700
}
```

CheckPoint enables the writing of all relevant information so restart of time domain run can be restarted. This is useful when not enough CPU time is specified for a run or more simulation time is required. In the former case, one can just re-submit the job. In the latter case, one needs to change the **MaximumTime** in **TimeStepping** to a larger value.

Action: Option for checkpoint

- **none**: no checkpoint
- **restart**: default, with checkpoint

Ntimesteps: The interval in number of time steps when the information is dumped for the subsequent restart. The default is 1000.