Acdtool

acdtool is a set of utility tools for preprocessing and postprocessing. It includes the following tasks

- **meshconvert**: mesh convertor from Cubit format to netcdf format, which is used in ACE3P; for preprocessing
- mesh: functions for quantifying and modifying a mesh; for preprocessing
- **postprocess**: functions for postprocessing of results from different ACE3P modules

acdtool submodules run serially with the exception of **acdtool postprocess volmontomode** and **acdtool postprocess rf**

acdtool meshconvert has no subtasks.

acdtool meshconvertdirect has no subtasks.

acdtool resource has no subtasks

acdtool mesh includes the following subtasks

- **stats**
- **check**
- **fix**
- **warpsurface**

acdtool postprocess includes the following subtasks

- **rf**
- **eigentomode**
- **volmontomode**
- **wake_new**
- **wake_direct**
- **transwake**
- **coaxsignal**
- **pic3pstats**
- **pic3pconvert**
- **track3p**
- **project**

acdtool meshconvert

Usage:

acdtool meshconvert <inputmeshfile.gen> [optional output file name]

- **inputmeshfile.gen**: The input mesh file in Genesis format generated by Cubit
- **[optional output file name]**: The output file in netcdf format after converted from Cubit Genesis format. This parameter is optional. If not specified, the default output file will be <inputmeshfile.ncdf>. This file is used as the mesh for ACE3P simulation.

Function: Convert the mesh format output by Cubit into the mesh format read by ACE3P. It will also list the mesh statistics and check the mesh quality as done by **acdtool mesh stats** and **acdtool mesh check**. The conversion fixes the inverted tetrahedral elements.

acdtool meshconvertdirect

Usage:

acdtool meshconvertdirect <inputmeshfile.gen> [optional output file name]

- **inputmeshfile.gen**: The input mesh file in Genesis format generated by Cubit
- **[optional output file name]**: The output file in netcdf format after converted from Cubit Genesis format. This parameter is optional. If not specified, the default output file will be <inputmeshfile.ncdf>. This file is used as the mesh for ACE3P simulation.

Function: Convert the mesh format output by Cubit into the mesh format read by ACE3P. It will also list the mesh statistics and check the mesh quality as done by **acdtool mesh stats** and **acdtool mesh check**. But the conversion does not fix the inverted tetrahedral elements.

acdtool resource

Usage:

acdtool resource <input file>

- **Input file**: omga3p input file
- **Output file**: a job submit script file on Perlmutter

Function: Provide guidance for resource allocation on Perlmutter

acdtool mesh

acdtool mesh stats

Usage:

acdtool mesh stats <inputmeshfile.ncdf>

- **inputmeshfile.ncdf**: The input mesh file in netcdf format

Function: List the statistics of a mesh.

acdtool mesh check

Usage:

acdtool mesh check <inputmeshfile.ncdf>

- **inputmeshfile.ncdf**: The input mesh file in netcdf format

Function: Check whether there are inverted second order tetrahedral elements in the mesh. These elements may lead to inaccurate calculation in ACE3P modules.

acdtool mesh fix

Usage:

acdtool mesh fix <inputmeshfile.ncdf> <outputmeshfile.ncdf>

- **inputmeshfile.ncdf**: The input mesh file in netcdf format containing inverted tetrahedrals
- **outputmeshfile.ncdf**: The output mesh file in netcdf format after fixing the inverted tetrahedrals

Function: Fix the inverted second order tetrahedral elements by removing the troublesome midpoints.

acdtool mesh deform

Usage:

acdtool mesh deform <inputmeshfile.ncdf> <outputmeshfile.ncdf> <scaling factor>

- **inputmeshfile.ncdf**: The input mesh file in netcdf format without deformation
- **outputmeshfile.ncdf**: The output mesh file in netcdf format by scaling the boundary deformation from TEM3P mechanical solver
- **scaling factor**: a number to multiply boundary displacement

Function: Provide artificial but better visualization for deformed mesh from TEM3P mechanical solver when boundary deformation is small.

acdtool mesh warpsurface

Usage:

acdtool mesh warpsurface <warp.in>

- **warp.in**: The input file specifying the parameters to be used. Below is an example.

File: SNS_1Cell_Closed_12K.ncdf
ExteriorBoundary: 6
WarpFile: SNS_1Cell_Closed_12K.txt

File: Name of mesh file in Netcdf format to generate triangle vertices on boundary surface
ExteriorBoundary: Boundary surface ID on which the triangle vertices will be written
WarpFile: Output file name for the triangle vertices in the following format
Index vertex_1 vertex_2 vertex_3
Example:

1 (0.0780002, 0.0135704, -0.0379568) (0.0831544, -0.00447699, -0.0350076) (0.0736616, -0.000687928, -0.0409466)
2 (0.0556233, -0.0825706, -0.00628908) (0.0494423, -0.0807614, -0.0211988) (0.067233, -0.0693693, -0.017149)
3 (0.0410526, -0.0744074, -0.033558) (0.0500267, -0.0593698, -0.0388919) (0.0558752, -0.0693768, -0.0293868)
4 (0.0441762, -0.0327638, -0.045) (0.0314193, -0.0349546, -0.045) (0.0375182, -0.0283088, -0.045)

Function: Write 2D triangular elements on boundary surface read by the plasma code Warp to construct metal surface.

acdtool postprocess rf

Usage:

acdtool postprocess rf <inputfile.rfpost>

- **inputmeshfile.rfpost**: The input file specifying the rf parameters to be calculated

Function: Provides a wide range of functionalities to obtain various rf parameters of a cavity mode. The parameters for performing these functionalities are specified in an input file. A sample input file named **sample.rfpost** can be generated by executing **acdtool postprocess rf**, which one can modify for specific calculations. The functionalities include

- **RFField**
- **RoverQ**
- **RoverQT**
- **kickFactor**
- **pointRoverQ**
- **dFSlater**
- **VFFT**
- **GBZFFT**
- **Multipole**
- **FieldMap**
- **IMPACTMap**
- **OpenPMD_IMPACT**
- **FieldAtPoint**
- **ALLFieldAtPoint**
- **FieldOnLine**
- **ALLFieldOnLine**
- **fieldOn2DBoundary**
- **fieldOnSurface**
- **maxFieldsOnSurface**
- **powerThroughSurface**

The following describes the parameters for each of the functionalities in the input file <inputfile.rfpost>.

➢ RFField

RFField is required to be the first item in the input file. It specifies the field information obtained from the field solvers of ACE3P, e.g. omega3p, s3p, etc.

```
RFField
{
  ResultDir   = omega3p_results  // Jobname
```

```
  FreqScanID  =     0  // For S3P only: frequency scan index
  ModeID     =     0
  xsymmetry  =     none   // [none, electric, magnetic]
  ysymmetry  =     none   // [none, electric, magnetic]
  gradient   =  2.00000e+07
  cavityBeta  =     1.00000     //for R/Q, V integral
  reversePowerFlow=     0   // [1=charging 0=decaying]
  x0         =     0.00000  // x for gradient calc
  y0         =     0.00000  // y for gradient calc
  gz1        =    -0.03500  // starting z for gradient calc
  gz2        =     0.03500  // starting z for gradient calc
  npoint     =    300
  fmnx       =    10
  fmny       =    10
  fmnz       =    50
}
```

ResultDir: The name of the directory where the field solver results are stored. This name is the "Jobname" specified in the batch job submission script. If no "Jobname" specified in the job submission script, the field solvers will write the results to the default result directories: "omega3p_results" for omega3p, "s3p_results" for s3p, etc.

FreqScanID: This input parameter is used only for s3s. s3p can perform a frequency scan to obtain S-parameters over a frequency range. The field results are saved for each frequency point of the scan. **FreqScanID** is the index of these frequency points, starting from "0".

ModeID: The mode index obtained by a field solver.

- For omega3p, **ModeID** is the index of the eigenmodes. **ModeID** starts from "0" (normally the lowest frequency mode in a run).
- For s3p, **ModeID** is the index of the port modes (excitation). The order for the port modes is the order they are specified in the s3p input file. **ModeID** starts from "0", counting the modes in the order at the first port, the next port, etc.

xsymmetry/ysymmetry: Indicated the symmetry type of the model

- **none**: No symmetry
- **magnetic**: Magnetic symmetry at x=0 or y=0
- **electric**: Electric symmetry at x=0 or y=0

gradient: The electric field gradient; the E and B fields in the mode files will be scaled such that the field gradient along a specified path (see below) is equal to the value **gradient**

- **positive real number, e.g., 1e6** [unit in V/m]
- **-1**: no scaling

cavityBeta: v/c of particles being accelerated, used for modeling of low beta cavities

reversePowerFlow: For omega3p calculation with open ports

- **0**: Field decaying through the open port
- **1**: Power flowing toward the cavity from the open port

x0/y0/gz1/gz2: Integration path for gradient calculation [units in m]

- **x0, y0**: Offset of the integratiom path in x and y
- **gz1, gz2**: Start and end points of the integration path in the z direction. The axial orientation of the cavity is required to be in the z direction. If **gz1** is given by a value larger than 1e6, it will take the minimum z position of the calculation domain. If **gz2** is given by a value larger than 1e6, it will take the maximum z position of the calculation domain.

npoint: The number of points along the integration path for the gradient calculation.

fmnx/fmny/fmnz: The grid for particle localization in tracking calculation

- **RoverQ**

Calculate R/Q. The unit of R/Q is ohm/cavity.

```
RoverQ
{
  ionoff    =    0
  modeID1   =   -1
  modeID2   =   -1
  x1        =    0.00000
  x2        =    0.00000
  y1        =    0.00100
  y2        =    0.00100
  z1        = 100000000.00000
  z2        = 100000000.00000
}
```

ionoff: Switch to perform or skip **RoverQ** calculation

- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

modeID1/modeID2: The range of modes for the R/Q calculation. If **modeID1**=-1, it will take the default value of "0". If **modeID2**=-1, it will take the default value of the maximum number of modes calculated by ACE3P modules.

x1/y1/z1/x2/y2/z2: Start and end points of the integration path for R/Q calculation. If **z1** is given a value larger than 1e6, it will take the minimum z of the calculation domain. If **z2** is given a value larger than 1e6, it will take the maximum z of the calculation domain. [units in m]

- **RoverQT**

Calculate the transverse RT/Q. The unit of RT/Q is ohm/cavity.

```
RoverQT
{
  ionoff    =    0
  modeID1   =   -1
  modeID2   =   -1
  x0        =    0.00000
  y0        =    0.00500
  z1        = 100000000.00000
  z2        = 100000000.00000
}
```

The definitions of the input parameters are the same as those for **RoverQ**. Since this is for transverse RT/Q calculation, an offset in the x-y plane (x0, y0) should be given for the integration path.

- **RoverQRoverQT**

Calculate the longitudinal R/Q and transverse RT/Q of both *x* and *y* planes of an eigenmode. The unit is ohm/cavity.

```
RoverQRoverQT
{
  ionoff    =    1
  modeID1   =   -1
  modeID2   =   -1
  r0        =    0.030
  z1        = 1e9
  z2        = 1e9
}
```

r0: is the radial offset for the integration.
The definitions of the rest input parameters are the same as those for **RoverQ** and **RoverQT**.

- **kickFactor**

Calculate the transverse kick factor. The kick factor is used to calculate the long-range transverse wakefield.

```
kickFactor
{
  ionoff    =    0
  modeID1   =   -1
  modeID2   =   -1
  x0        =    0.00500
  y0        =    0.00000
  z1        = 100000000.00000
  z2        = 100000000.00000
}
```

The definitions of the input parameters are the same as those for **RoverQ**. Since this is for the transverse kick factor calculation, an offset in the x-y plane (x0, y0) should be given for the integration path.

- **pointRoverQ**

Similar to R/Q, but only use the field at a point. This quantity is used to calculate the power requirement for producing a specified electric filed at the point (x0,y0,z0).

```
pointRoverQ
{
  // pointRoverQ=E^2/(omega*U)
  ionoff    =    1
  modeID1   =   -1
  modeID2   =   -1
  x0        =    0.00000
  y0        =    0.00000
  z0        =    0.00010
}
```

ionoff: Switch to perform or skip **pointRoverQ** calculation

- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

modeID1/modeID2: The range of modes for the dFSlater calculation. If **modeID1**=-1, it will take the default value of "0". If **modeID2**=-1, it will take the default value of the maximum number of modes calculated by ACE3P modules.

x0,y0,z0: the point where the pointRoverQ will be calculated, [units in m].

- **dFSlater**

Calculate frequency offset due to a geometry error in a region defined by (x1, x2, y1, y2, z1, z2) using the Slater perturbation theory.

```
dFSlater
{
  ionoff    =    1
  modeID1   =   -1
  modeID2   =   -1
  derror    = 100.0e-6
  x1        = 1e9
  x1        = 1e9
  y2        = 1e9
  y2        = 1e9
  z1        = 1e9
  z2        = 1e9
}
```

ionoff: Switch to perform or skip **dFSlater** calculation

- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

modeID1/modeID2: The range of modes for the dFSlater calculation. If **modeID1**=-1, it will take the default value of "0". If **modeID2**=-1, it will take the default value of the maximum number of modes calculated by ACE3P modules.

derror: the error/deviation of the geometry normal to the surface in m. Positive indicates error that reduces the RF volume.

x1/y1/z1/x2/y2/z2: defines the volume which contents the surfaces with geometry error. If **z1** is given a value larger than 1e6, it will take the minimum z of the calculation domain. If **z2** is given a value larger than 1e6, it will take the maximum z of the calculation domain. [units in m]

- **VFFT**

Calculate the multipole components of the longitudinal voltage.

```
VFFT
{
  ionoff    =    0
  printGroup  = nterm   // [nterm, ModeID]
  modeID1   =   -1
  modeID2   =   -1
  nterm     =    4
  ntheta    =   16
  r0        =    0.00500
  z1        = 100000000.00000
```

```
  z2        = 100000000.00000
}
```

ionoff: Switch to perform or skip **VFFT** calculation

- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

printGroup: Specifies how the results will be printed in either of the following ways.

- **nterm**: The results will be grouped by the multipole component **nterm**. The same multipole component of all the modes will be printed as a group, starting from the lowest multipole to the highest multipole.
- **ModeID**: The results will be grouped by **ModeID**. All multipole components of a mode will be printed together, starting from **modeID1** to **modeID2**.

modeID1/modeID2: The range of modes for the calculation. If **modeID1**=-1, it will take the default value of "0". If **modeID2**=-1, it will take the default value of the maximum number of modes calculated by ACE3P modules.

nterm: The number of multipole components to be calculated

ntheta: The number of voltage integrations performed on a circle for multipole calculation, preferred to be a power of 2 (e.g., 4, 8, 16, 32, …)

r0: The radius of the circle on which the voltage integration are performed [unit in m]

z1/z2: Start and end points of the integration path in the longitudinal direction. If **z1** is given a value larger than 1e6, it will take the minimum z of the calculation domain. If **z2** is given a value larger than 1e6, it will take the maximum z of the calculation domain. [units in m]

- **GBZFFT**

Calculate the multipole components of a particle momentum change due to the RF fields. This calculation only applies to the mode with **ModeID** specified in **RFField**.

```
GBZFFT
{
  ionoff     =     0
  filename    = filename1
  nterm      =     4
  ntheta     =    16
  energy     =  5.00000e+07
```

```
  npoint     =    300
  r0         =     0.00500
  z1         = 100000000.00000
  z2         = 100000000.00000
  phase1     =   -180.00000
  phase2     =    180.00000
  nphase     =     19
}
```

ionoff: Switch to perform or skip **GBZFFT** calculation

- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

filename: The output file name

nterm: The number of multipole components to be calculated

ntheta: The number of voltage integrations performed on a circle for multipole calculation, preferred to be a power of 2 (e.g., 4, 8, 16, 32, …)

energy: Initial energy of the particle [unit in eV]

npoint: The number of points for the integration

r0: The radius of the circle on which the voltage integration are performed [unit in m]

z1/z2: Start and end points of the integration path in the longitudinal direction. If **z1** is given a value larger than 1e6, it will take the minimum z of the calculation domain. If **z2** is given a value larger than 1e6, it will take the maximum z of the calculation domain. [units in m]

phase1/phase2: The range of the RF phase [units in degree]

nphase: The number of points for the RF phase scan

➢ Multipole

Calculate the multipole components of the momentum change due to the RF fields, using a simplified method. It only calculates the dipole and quadrupole terms. This calculation only applies to the mode with **ModeID** specified in **RFField**.

```
Multipole
{
  ionoff     =     0
```

```
  filename    = filename1
  energy      =  5.00000e+07
  npoint      =    300
  r0          =     0.00500
  z1          = 100000000.00000
  z2          = 100000000.00000
  phase1      =   -180.00000
  phase2      =    180.00000
  nphase      =     19
}
```

The definitions of the input parameters are the same as those for **GBZFFT**.

- **FieldMap**

Outputs a field map on a regular grid. The field map data will be written to *Efield-map.dat* and *Bfield-map.dat* for the E and B fields, respectively. This calculation only applies to the mode with **ModeID** specified in **RFField**.

```
FieldMap
{
  ionoff     =     0
  nx         =    21
  ny         =    21
  nz         =    50
  x1         =     0.00000
  x2         =     0.02000
  y1         =     0.00000
  y2         =     0.02000
  z1         = 100000000.00000
  z2         = 100000000.00000
}
```

ionoff: Switch to perform or skip **FieldMap** calculation

- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

nx/ny/nz: The numbers of grid lines in x, y, and z.

x1,x2: The range of grid lines in x [units in m]

y1,y2: The range of grid lines in y [units in m]

z1,z2: The range of grid lines in z [units in m]. If **z1** is given a value larger than 1e6, it will take the minimum z of the calculation domain. If **z2** is given a value larger than 1e6, it will take the maximum z of the calculation domain.

- **IMPACTMap**

Similar to the "FieldMap". The field map data will be written to *EBfield-map-"filename".dat* in the format for the beam dynamics code IMPACT.

```
IMPACTMap
{
  ionoff    =    0
  filename  =    py
  rot180    =    0
  nx        =   21
  ny        =   21
  nz        =   50
  x1        =    0.00000
  x2        =    0.02000
  y1        =    0.00000
  y2        =    0.02000
  z1        = 100000000.00000
  z2        = 100000000.00000
}
```

Rot180: flag for rotating the field map about z axis by 180 degrees relative to the calculation model

- **0**: Do not rotate.
- **1**: Rotate about z by 180 degrees.

- **OpenPMD_IMPACT**

Outputs a field map on a regular grid in OpenPMD format. The field map data will be written to E_Real.h5, E_Imag.h5, B_Real.h5, B_Imag.h5 for the E and B fields, respectively. This calculation only applies to the mode with **ModeID** specified in **RFField**.

```
OpenPMD_IMPACT
{
  ionoff    =    0
  nx        =   21
  ny        =   21
  nz        =   50
  x1        =    0.00000
  x2        =    0.02000
  y1        =    0.00000
```

```
  y2        =     0.02000
  z1        = 100000000.00000
  z2        = 100000000.00000
}
```

ionoff: Switch to perform or skip **FieldMap** calculation

- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

nx/ny/nz: The numbers of grid lines in x, y, and z.

x1,x2: The range of grid lines in x [units in m]

y1,y2: The range of grid lines in y [units in m]

z1,z2: The range of grid lines in z [units in m]. If **z1** is given a value larger than 1e6, it will take the minimum z of the calculation domain. If **z2** is given a value larger than 1e6, it will take the maximum z of the calculation domain.

- **FieldAtPoint**

Calculate E and B fields at a point. This calculation only applies to the mode with **ModeID** specified in **RFField**.

```
FieldAtPoint
{
  ionoff    =     0
  x0        =     0.00000
  y0        =     0.00000
  z0        =     0.00000
}
```

ionoff: Switch to perform or skip **FieldAtPoint** calculation

- **0**: Do not perform the calculation
- **1**: Perform the calculation

x0,y0,z0: The coordinates at which the fields will be evaluated [units in m]

- **ALLFieldAtPoint**

Calculate E and B fields at a point for a number of modes.

```
ALLFieldAtPoint
{
  ionoff    =     0
```

```
  modeID1    =    -1
  modeID2    =    -1
  x0         =    0.00000
  y0         =    0.00000
  z0         =    0.00000
}
```

- **ionoff:** Switch to perform or skip **ALLFieldAtPoint** calculation
  - **0**: Do not perform the calculation.
  - **1**: Perform the calculation.

modeID1/modeID2: The range of modes for the calculation. If **modeID1**=-1, it will take the default value of "0". If **modeID2**=-1, it will take the default value of the maximum number of modes calculated by ACE3P modules.

x0,y0,z0: The coordinates at which the fields will be evaluated [units in m]

- **FieldOnLine**

Calculate E and B fields along a line. This calculation only applies to the mode with **ModeID** specified in **RFField**. The E and B fields are scaled to the gradient specified in **RFField**.

```
FieldOnLine
{
  ionoff     =     0
  npoint     =    300
  filename    = filename1
  rfphase     = 0.00000
  x1         =    0.00000
  x2         =    0.00000
  y1         =    0.00100
  y2         =    0.00100
  z1         = 100000000.00000
  z2         = 100000000.00000
}
```

ionoff: Switch to perform or skip **FieldOnLine** calculation
- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

npoint: The number of points along the line to be calculated

rfphase: The rf phase at which the fields will be evaluated

filename: The output file for the fields along the line. The E and B fields at **rfphase** will be written to "filename".e and "filename".b, respectively. The complex E and B fields will be written to "filename".ec and "filename".bc, respectively.

x1/y1/z1/x2/y2/z2: Start and end points of the line for the field evaluation. If **z1** is given a value larger than 1e6, it will take the minimum z of the calculation domain. If **z2** is given a value larger than 1e6, it will take the maximum z of the calculation domain. [units in m]

- **ALLFieldOnLine**

Calculate E and B fields for a number of modes along a line. The E and B fields are directly from omega3P eigenmode, which are normalized to a total stored energy in the system of $\varepsilon_0/2$.

```
ALLFieldOnLine
{
  ionoff    =    0
  rot180    =    0
  modeID1   =   -1
  modeID2   =   -1
  npoint    =  300
  filename  = filename1
  rfphase   =    0.00000
  x1        =    0.00000
  x2        =    0.00000
  y1        =    0.00100
  y2        =    0.00100
  z1        = 100000000.00000
  z2        = 100000000.00000
}
```

The definitions of the parameters are the same as those for **FieldOnLine**, with the following additional parameters to define the range of modes.

modeID1/modeID2: The range of modes for the calculation. If **modeID1**=-1, it will take the default value of "0". If **modeID2**=-1, it will take the default value of the maximum number of modes calculated by ACE3P modules.

Rot180: flag for rotating the field about z axis by 180 degrees relative to the calculation model

- **0**: Do not rotate.
- **1**: Rotate about z by 180 degrees.

- **fieldOn2DBoundary**

Output the E and B fields on the x=0 symmetry plane of the surface "surfaceID". This calculation only applies to the mode with **ModeID** specified in **RFField**.

```
fieldOn2DBoundary
{
  ionoff     =     0
  filename    = filename1
  surfaceID  =     6
}
```

ionoff: Switch to perform or skip **fieldOn2DBoundary** calculation

- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

filename: The filename for the output

surfaceID: The ID of the surface for calculating the E and B fields. It is set in Cubit.

- **fieldOnSurface**

Output E and B fields on a surface. This calculation only applies to the mode with **ModeID** specified in **RFField**.

```
fieldOnSurface
{
  ionoff     =     0
  filename    = filename1
  surfaceID  =     6
  output      = amplitude   // [component, amplitude]
}
```

ionoff: Switch to perform or skip **fieldOnSurface** calculation

- **0**: do not perform the calculation.
- **1**: Perform the calculation.

filename: The output file name

surfaceID: The ID of the surface. It is set in Cubit

output: The format of the output

- **amplitude**: Amplitude of the field is written to the output file.
- **component**: x,y,z components of the field are written to the output file.

- **maxFieldsOnSurface**

Calculate the maximum fields on a surface. This calculation only applies to the mode with **ModeID** specified in **RFField**.

```
maxFieldsOnSurface
{
  ionoff    =    0
  surfaceID  =    6
}
```

ionoff: Switch to perform or skip **maxFieldsOnSurface** calculation

- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

surfaceID: The ID of the surface where the maximum surface fields are evaluated. It is set in Cubit.

- **powerThroughSurface**

Calculate the power flow through a surface. This calculation only applies to the mode with **ModeID** specified in **RFField**.

```
powerThroughSurface
{
  ionoff    =    0
  surfaceID  =    6
}
```

ionoff: Switch to perform or skip **powerThroughSurface** calculation

- **0**: Do not perform the calculation.
- **1**: Perform the calculation.

surfaceID: The ID of the surface through which the power is calculated. It is set in Cubit.

The output power [unit W] is complex, where the real part is the average power flow calculated using the complex Poynting vector.

acdtool postprocess eigentomode

Usage:

acdtool postprocess eigentomode <jobname>

- **jobname**: The directory name specified in the batch job submission script. If no "Jobname" specified in the job submission script, the field solvers will write the results to the default result directories: "omega3p_results" for omega3p and "s3p_results" for s3p.

Function: Convert eigenvector solutions from omega3p and s3p to mode files (*.mod) that can be used for visualization and analysis in ParaView. The default in these solvers is to convert to mode files automatically while running omega3p and s3p, which can be disabled by setting **Toggle: off** in **PostProcess** of the startup file.

acdtool postprocess volmontomode

Usage:

acdtool postprocess volmontomode <jobname>

- **jobname**: The directory name specified in the batch job submission script. If no "Jobname" specified in the job submission script, the field solvers will write the results to the default result directories: "t3p_results" for t3p and "pic3p_results" for pic3p.

Function: Convert eigenvector solutions (*.out) using the volume **Monitor** in t3p and pic3p to mode files (*.mod) that can be used for visualization and analysis in ParaView. The default in these solvers is to convert *.out files to *.mod files automatically while running t3p and pic3p. It is used when they are not converted when there is not enough CPU time to complete the conversion.

acdtool postprocess wake_new

Usage:

acdtool postprocess wake_new <jobname> <x y>

- **jobname**: The directory name specified in the batch job submission script. If no "Jobname" specified in the job submission script, the field solver will write the results to the default result directories: "t3p_results" for t3p.
- **x y**: The transverse coordinates of a point [units in m]

Function: Calculate the longitudinal wakefield at a certain transverse position in t3p. The wakefield at the drive particle beam position is automatically calculated in t3p, so this tool is used if the wakefields at other transverse positions are needed. For example, in order to use Panofsky-Wenzel theorem to calculate the transverse wakefield from the longitudinal, the longitudinal wakefield is required at another transverse position other than that calculate automatically. Because the Laplace solver is mesh-based, it returns the wakefield at a mesh point closed to <x, y> indicated in the wakefield output file. Therefore, it is advantageous to use <x, y> on the grid. The output file is stored as <jobname>/OUTPUT/wakefield.out, where the file name "wakefield" has been specified in the wakefield monitor of t3p**.** The loss factor will be calculated and written to the file as well.

acdtool postprocess wake_direct

Usage:

acdtool postprocess wake_direct <jobname> <x y>

- **jobname**: The directory name specified in the batch job submission script. If no "Jobname" specified in the job submission script, the field solver will write the results to the default result directories: "t3p_results" for t3p.
- **x y**: The transverse coordinates of a point [units in m]

Function: Similare to **wake_new**, calculate the longitudinal wakefield at a certain transverse position in t3p using direct integration (see t3p command syntax for details). The output file is stored as <jobname>/OUTPUT/wakefield.out, where the file name "wakefield" has been specified in the wakefield monitor of t3p**.** The loss factor will be calculated and written to the file as well.

acdtool postprocess transwake

Usage:

acdtool postprocess transwake <jobname> <x1 y1> <x2 y2>

- **jobname**: The directory name specified in the batch job submission script. If no "Jobname" specified in the job submission script, the field solvers will write the results to the default result directories: "t3p_results" for t3p.
- **x1 y1**: The transverse coordinates of a point [units in m]
- **x2 y2**: The transverse coordinates of a point [units in m]

Function: Calculate the transverse wakefield at a certain transverse position in t3p using Panofsky-Wenzel theorem. One specifies the driven beam transverse position as <x0, y0> in the t3p startupfile, and choose two transverse points <x1, y1> and <x2, y2> around <x0, y0> to calculate the derivative of the longitudinal wakefield. 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 <jobname>/OUTPUT/wakefield.out. The kick factor will be calculated as well. Note that it only works for **wake_new** using indirect method to calculate the longitudinal wakefield.

acdtool postprocess coaxsignal

Usage:

acdtool postprocess coaxsignal <jobname>

- **jobname**: The directory name specified in the batch job submission script. If no "Jobname" specified in the job submission script, the field solvers will write the results to the default result directories: "t3p_results" for t3p.

Function: Calculate the signal at a coaxial port from a beam current excitation in t3p. Currently, it works for coaxial cables with 50 Ω impedance. It uses the power and the mode coefficient of the outgoing mode to determine the signal. Thus in the t3p startupfile, one has to specify a **power** monitor and a **modevoltage** monitor. The file with name "signal.out" will be written to <jobname>/OUTPUT/. The format of "signal.out" is <t, V, I>, where t is the time [unit in s], V the voltage [unit in V], and I the beam current [A], respectively.

acdtool postprocess pic3pstats

Usage:

acdtool postprocess pic3pstats <filename> <symmetry factor>

- **filename**: The filename where the particles are stored in netcdf format after a pic3p run.
- **symmetry factor**: The symmetry factor of the model: 1 for no symmetry, 4 for a quarter model with 2 symmetry planes

Function: Compute phase space statistics from writing particle data snapshots in a pic3p run. The particle data are stored in pic3p_ressults/OUTPUT/Particles directory.

acdtool postprocess pic3pconvert

Usage:

acdtool postprocess pic3pconvert <filename>

- **filename**: The filename containing the particle distribution

Function: Convert 6D phase space pic3p particle dumps from netcdf to ascii or vice versa. The netcdf file ends with extension “ncdf” and the ascii file with “txt”.

acdtool postprocess track3p

Usage:

acdtool postprocess track3p <inputfile.acdtool> <jobname>

- **inputfile.acdtool:** The input file specifying the track3p postprocess parameters to be calculated
- **jobname**: The directory name specified in the batch job submission script. If no "Jobname" specified in the job submission script, the field solvers will write the results to the default result directories: "track3p_results" for track3p.

Function: Provides functionalities to extract some parameters of track3p results. The parameters for performing these functionalities are specified in an input file. The functionalities include

- **EnhancementCounter**
- **Trajectory**

➢ EnhancementCounter

EnhancementCounter: Specifies the parameters used to calculate the particle enhancement counter.

```
EnhancementCounter:
{
  Token          :    on
  SEYFileName1   :    copper.dat
  SEYFileName2   :    niobium.dat
  BoundarySurfaceID1 :   6
  BoundarySurfaceID2 :   3
  SolidVolID     :    0
  MinimumEC      :    0.01
  OutputFile     :    en
}
```

EnhancementCounter: Specifies the parameters used to calculate the particle enhancement counter. The enhancement counter is defined as the total number of secondary electrons originated form a single primary electron during the number of rf cycles in a simulation. It is written as

$$ec = \delta_1 * \delta_2 * \ldots * \delta_m$$

where δ_i is the secondary emission yield (SEY) at the *i*th impact and its value is determined by the impact energy and the SEY curve. The parameters for setting the enhancement counter are

- **Token**: **on** or **off**
  on: Turn on the enhancement counter.
  off: Turn off the enhancement counter.
- **BoundarySurfaceID1**: The boundary surface ID, which is set in Cubit, used for calculating the enhancement counter
- **SEYFileName1:** The name of the file containing the SEY for the corresponding material. The first and second columns in the file are the impact energy and the number of emitted particles due to a single particle impact, respectively.
- **BoundarySurfaceID2**: The boundary surface ID, which is set in Cubit, used for calculating the enhancement counter
- **SEYFileName2:** The name of the file containing the SEY for the corresponding material on **BoundarySurfaceID2**. The first and second columns in the file are the impact energy and the number of emitted particles due to a single particle impact, respectively.
- **SolidVolID:** Window volume ID, which is set in Cubit
- **MinimumEC:** minimum enhancement counter, less than it will not be written, default: 10
- **OutputFile:** output file name, it's dumped under ./jobname/

➢ Trajectory

Trajectory: Extract single a particle trajectory

```
Trajectory:
{
  Token          :    on
  ParticleID     :    2300 7011
}
```

- **Token**: **on** or **off**
  on: Turn on the single particle trajectory extraction.
  off: Turn off the single particle trajectory extraction.
- **ParticleID**: particle id for extracting trajectory.

acdtool postprocess project

Usage:

acdtool postprocess project <eigenmodes path> [displacements path]

- **eigenmodes path**: The path to the TEM3P eigen mode simulation results folder.
- **displacements path**: The path to the TEM3P static or harmonic response simulation results folder.

Function: Calculates the L2 projections of the vector displacement fields for the static or harmonic response TEM3P problem onto a set of the mechanical eigenmodes determined by the TEM3P eigenmode solver. The decomposition coefficients are calculated by multiplying the vector displacement field by the eigenmode shapes and integrating the product over the computational mesh, so the mesh used for static/harmonic response and eigenmode TEM3P simulation must be identical in order to use the projection feature. If the **displacements path** is omitted, it projects the set of the mechanical eigenmodes onto themselves to check the orthogonally condition. The results are printed to stdout and also saved into acdtool.log.