GGI_TLM User Documentation

George Green Institute for Electromagnetics Research

University of Nottingham

UK

January 2013

Revisions:

26th February 2013: Include more of the GGI_TLM packets into the documentation, with examples.

8th April: Include post processing and visualisation

4th June: Final update and checking before initial release

10th September 2014: Update to include revisions to the make process and to include additions to the code capability

16th October 2014: Include notes on the automatic documentation of simulations and updates to the automatic running scripts.

18th November 2014: include parameterised geometry notes, revised excitation surface list,

22nd January 2017: Document the changes to the run_automatic_test scripts relating to the viewing of geometry and meshes with Paraview.

3rd September 2018: Document the use of derived parameters in a GGI_TLM input file

9th October 2019: Document PML plus other previously un-documented developments

1. Introduction

The **GGI_TLM** time domain Electromagnetic field solver described in this document allows the creation and solution of 3D electromagnetic problems on a structured cubic mesh. **GGI_TLM** has been created as a consolidation of software developed at the University of Nottingham under the HIRF-SE project funded by the European Community's Seventh Framework Programme FP7/2007-2013, grant agreement no 205294 and also the EPSRC funded Flaviir project (www.flaviir.com).

The projects feeding into this software have been concerned with the development of techniques for modelling electromagnetic interactions with materials and cables, the code capabilities reflect this work.

The software has been applied to a range of simulation tasks over recent years and these applications are reflected in the large number of test cases provided with the software. We cannot give any guarantees regarding the validity or accuracy of the results of this code. We strongly suggest that any new problem analysed using GGI_TLM is validated appropriately and confidence in the results obtained through comparison with independent data.

It is hoped that GGI_TLM, the supporting codes and documentation will be consolidated, improved and developed in the future. We strongly encourage users to send us feedback regarding the code, new test cases etc so as to help in this development.

1.1 Licensing

There are 2 license types used in GGI_TLM.

GPL- GNU General Public License v.3 (see file **COPYING**) in directory **GGI_TLM/**

LGPL- GNU Lesser General Public License v.3 (see file **COPYING.LESSER**) in directory **GGI_TLM/**

GPL applies to all source code with the exception of the **EISPACK** directory: **GGI_TLM/GGI_TLM/SRC/EISPACK.** It is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or(at your option) any later version.

LGPL applies to the **EISPACK** directory: **GGI_TLM/GGI_TLM/SRC/EISPACK**

1.2 Disclaimer

GGI_TLM is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

1.3 Applications

Shielding calculations
Coupling from external fields onto cable systems
Radiation from equipment
Antenna analysis: impedance and far field pattern
Waveguide and cavity analysis
S parameter calculation
Radar Cross section
Specific Absorption Rate
Modelling objects in a resonant environment

1.4 Capabilities

The geometry may contain conducting and dielectric volumes, PEC surfaces and thin layers as well as wire bundles with junctions and terminations. Materials and cables may have frequency dependent properties.

The TLM solution uses the symmetrical condensed node.
Problems may be sourced using either hard or soft sources, plane waves may be launched from a Huygens surface or sources may be placed on wires.

GGI_TLM may incorporate Spice lumped circuit models by linking the software with Ngspice. This is described in a separate document, **GGI_TLM_Ngspice_documentation**.

A perfectly matched layer (PML) boundary condition has been implemented.

Time domain fields at points, over surfaces or volumes as well as cable currents may be output

Far field and radar cross section may also be calculated in the frequency domain.

Some post processing procedures are available for common tasks such as calculating Fourier Transforms, combining data to give S parameters, antenna impedance etc.

Theoretical manuals are provided so that the underlying theory and the implementation of that theory may be understood.

1.5 Parallel implementation

The GGI_TLM TLM code may be compiled to run in parallel. The parallel implementation uses mpi. In order to run a job using n_p processes, the mesh is decomposed into n_p equal partitions in the z direction. Each process performs the update on its own mesh partition and the partitions are linked appropriately in the TLM connection process at the partition boundaries. The parallel implementation is not optimum by any means but is effective in many cases. Cable and material models are included in the parallel implementation, each processor owns the cable and material cells which happen to be in its mesh partition. At the moment, no attempt is made to load balance in order to take into account the extra work involved in the cable and material updates.

If the geometry is long and thin then it will always be advantageous to define the longest dimension to be in the z direction in the mesh.

The parallel implementation does not yet extend to the software when linked to Ngspice.

1.6 Test cases

A variety of test cases have been generated which illustrate the capabilities of the code and demonstrate a wide range of applications. The test set may be run automatically in order to test code installation and developments. This enables the user to check the results quickly, the test cases therefore provide a basis for validation and testing of the code.

The test cases and the supporting scripts may be used as templates for developing user's own models.

Many of the test cases are described in the accompanying documents in the directory **DOCUMENTATION/TEST_CASE_DOCUMENTATION**.

2. Compiling and running GGI_TLM

GGI_TLM may be compiled and run under both UNIX and Windows. If operating under windows then Cygwin must be installed. Cygwin provides a UNIX environment within windows. Once cygwin is installed the compilation process should proceed in exactly the same manner as for UNIX.

2.1 Prerequisites

GGI_TLM has the following requirements

Sequential installation:

Fortran 90 compiler: we have used **gfortran** version 4.4. Other compilers may be used though we have not tested the compilation with any other compilers.

Mpi installation:

Fortran 90 compiler: we have used **mpif90.** Other compilers may be used though we have not tested the compilation with any other compilers.

mpi library

2.2 Support codes

GGI_TLM is designed to make use of the openly availably codes **paraview** and **gnuplot**. Paraview allows the visualisation of geometry, computational meshes and animation of fields in 3D. Gnuplot may be used to plot curves.

2.3 GGI_TLM Directory Structure

At the top level **GGI_TLM** contains text files; a README file and files with license information. The directory **GGI_TLM/ GGI_TLM** contains the main GGI_TLM code and test data.

Inside **GGI_TLM/ GGI_TLM** are eight directories and three files.

Directories:

SRC	containing the source code
DOCUMENTATION	containing the user guide, theoretical documentation and test case documentation in pdf format
TEST_DATA	containing the test case directories
ANALYTIC_TEST_DATA	containing the test case directories for analytic validation codes
SOFTWARE_NOTES	containing text files including a work log, known problems and other notes relating directly to the source code development.
bin	Directory for GGI_TLM executabes
GGI_TLM_LIBS	Directory for GGI_TLM libraries

GGI_TLM_MODULE_FILES	Directory for the GGI_TLM fortran90 module files
TEMPLATE_PROJECT	Directory for use as a template for new GGI_TLM projects. The directory contains an example problem in the same form as the test data directory so that it can be run, post processed and the results viewed in the same manner. Please see the test case documentation for details.

Files:

README	Text file containing instructions for compiling and running GGI_TLM
make_configuration_data	file containing information which must be set by the user in order to Compile GGI_TLM on the user's system
run_configuration_data	file containing information which must be set by the user in order to run GGI_TLM on the user's system

2.4 Making GGI_TLM

GGI_TLM has been designed to run in a UNIX like operating system. For running under Windows, GGI_TLM requires the installation of Cygwin, a Unix type environment for Windows. This is described in the documents **Cygwin_installation_for_GGI_TLM** and **Cygwin_installation_for_GGI_TLM_Windows_10** as appropriate.

Most of the directories used In making GGI_TLM are set automatically relative to the GGI_TLM directory however if you want to make the mpi version then you will need to set the path for mpi:

Edit the file **make_configuration_data** in order to re-set the paths for the following:

```
# fortran compiler name. For example:
FC=mpif90

# path to mpif.h file (if required). For example:
MPI_INCLUDE=/usr/include/mpich2

You will also need to set the fortran compiler name and the
complation flags:

# compilation flags. For example:
FLAGS= -fdefault-real-8 -cpp -fbounds-check -finit-real=nan -finit-
integer=nan -J$(MOD_DIR) -I$(MOD_DIR) -I$(MPI_INCLUDE)
```

GGI_TLM can be made in two configurations, a sequential solver and a parallel solver (mpi). The two versions of the solver are made explicitly with the commands

make PARALLEL_MODE=SEQ

make PARALLEL_MODE=MPI

The command

make clean removes all the object and library files

Please note that you must **make clean** before changing between making the sequential and the mpi versions of GGI_TLM. Please note that if you need to make the mpi version of the code then you should make the sequential code afterwards i.e. **make clean** then **make PARALLEL_MODE=SEQ**. This is required at the moment in order to ensure that the model builder and cable model builder codes are linked to sequential versions of the required subroutines.

The command **make clean_all** does the same as **make clean** but in addition all the executables are deleted.

From version 1.0.3 it is possible to make GGI_TLM with the command

make

The action of this command is as follows:

If there is a previously made version of GGI_TLM (sequential or parallel) then this version is re-made.

If the code has been newly installed or cleaned then the sequential version is made by default.

It is useful to ensure that the current directory and the `EXECUTABLE_DIR` is in your path in order that the test cases run correctly. The paraview and gnuplot executables should also be included in the path if they are not already there. This can be achieved with the command of the form:

PATH="./:/home/chris/SOFTWARE/GGI_TLM/ GGI_TLM/bin:$PATH"

Where the path is set appropriately for the user's system. Other directories can be included in a similar manner (paraview, gnuplot etc)

The process for making GGI_TLM linked to Ngpsice is described in the document **GGI_TLM_Ngpsice_documentation.**

3. Running GGI_TLM

THE GGI_TLM system consists of 9 codes which are used to create models, visualise and check geometry, meshes and cable models, create frequency dependent material models from measured data, run and then post-process results.

GGI_TLM consists of the following codes:

GGI_TLM_model_builder

GGI_TLM_model_checks

GGI_TLM_cable_model_builder

GGI_TLM_cable_model_checks

GGI_TLM_filter_fit

GGI_TLM_filter_format_convert

GGI_TLM_material_model_checks

GGI_TLM_SEQ/ GGI_TLM_MPI

GGI_TLM_post_process

GGI_TLM_document.py

All the Fortran codes with the exception of the main solver **GGI_TLM_MPI** are sequential and are run by simply typing the name of the code. The **GGI_TLM_SEQ** solver code may be run on a single process in the same way however in order to run **GGI_TLM_MPI** i.e. a parallel run, we require a command of the form

mpirun –np x GGI_TLM_MPI

where x is the number of processors to run the solution on. This command may be different depending on the user's system.

Note that this run command must be specified in the file **GGI_TLM/run_configuration_data** in order to run the test cases. The process of setting up and running the test cases is described in an accompanying document.

Running a **GGI_TLM** model requires an input file which specifies the geometry, materials, cables, boundary conditions, excitation and outputs required for a problem. The input file should have the extension **.inp**. The construction of the input file is described in detail in the subsequent sections. The model may also require files which describe material and cable properties.

Each code prompts the user for the name of the input file. The name without the **.inp** extension should be provided i.e. if we have a file **NAME.inp** then enter **NAME** only
All output files have the format **NAME.output_extension**. The file extensions for the different types of output are summarised in a subsequent section.

Running a model proceeds in a number of stages, firstly a 3D meshed model is created from the input file using the code **GGI_TLM_model_builder**. This creates a mesh from the geometry description. The model may be examined and checked by using the code **GGI_TLM_model_checks** which produces files for 3D visualisation of the model and mesh with **paraview**.

If the problem includes cables then the code **GGI_TLM_cable_model_builder** creates a cable model.

The cable model may be checked using the code **GGI_TLM_cable_model_checks.** This allows the following actions:

1. Individual cable cross section geometries may be examined
2. The cable route can be to the written out to a vtk file for visualisation within **paraview**.
3. The geometry of a cable bundle may be examined at any segment in the cable mesh
4. Details of cable junctions may be examined at any cell
5. Details of cable junctions may be examined at any cell face

GGI_TLM_filter_fit generates frequency dependent material models from measured material parameter

GGI_TLM_filter_format_convert allows the conversion of the material filter formats between rational functions, pole-zero format and pole-residue format. This is only used if the material models are to be used in other codes which use a different filter format definition – it is not required to run GGI_TLM.

GGI_TLM_material_model_checks allows the visualisation of material model frequency responses as well as producing files for visualisation of the allocation of materials onto the computational mesh.

Once the mesh and the cable model have been built then **GGI_TLM** is run with a command of the form:

mpirun –np x GGI_TLM_MPI

where x is the number of processors to run the solution on.

A general information file with extension **.info** is always produced. Any warnings are written to a **.warnings** file and if there are cables in the problem specification then a **.cable_info** file is produced which contains details of the cable aspects of the model.

Once the model has been run, the results may be post processed using **GGI_TLM_post_process**. There are a number of post processing options depending on the type of analysis and the particular output produced as detailed in section 15.

The process is illustrated in the process diagram below:

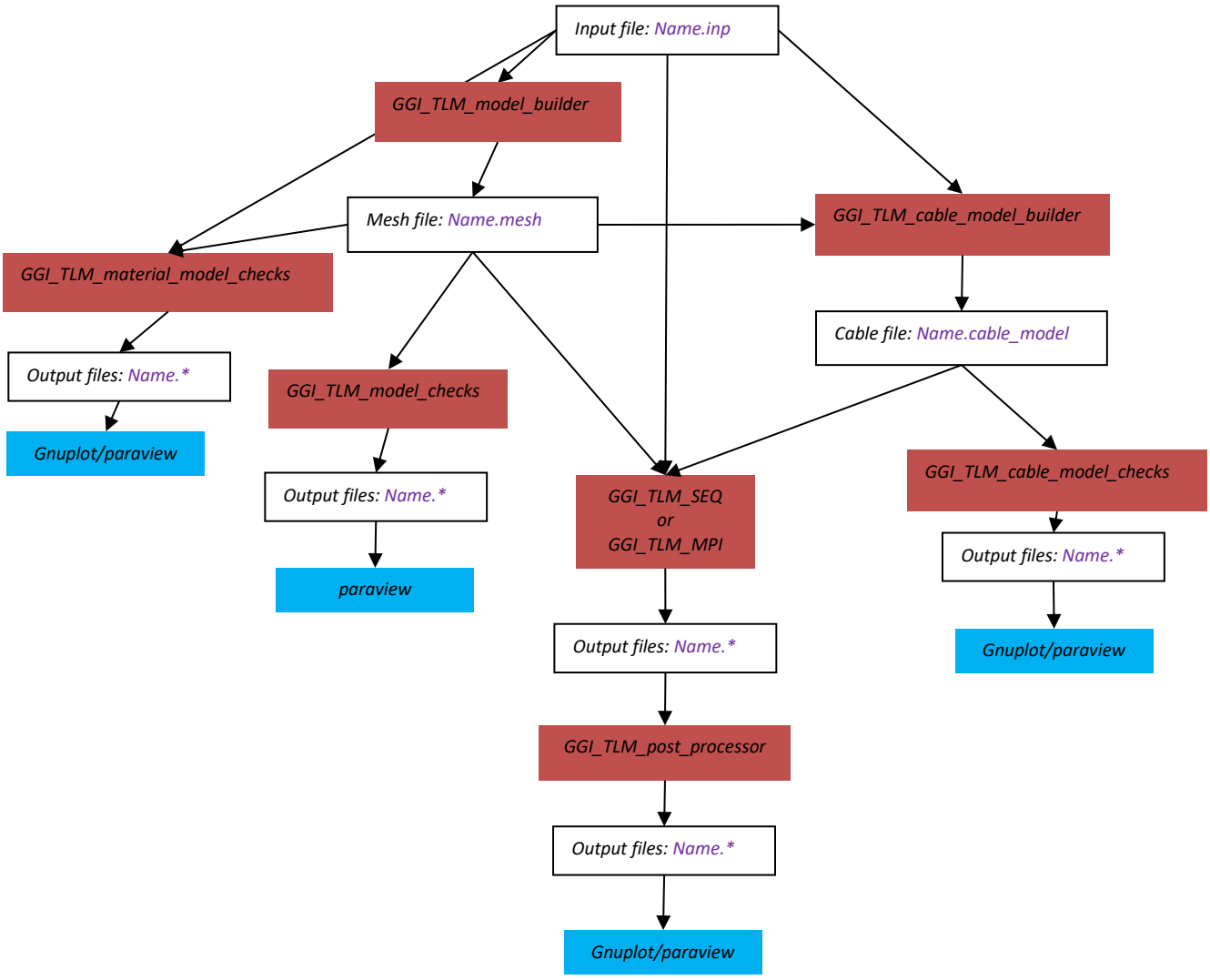


Figure 3.1 GGI_TLM process diagram.

There is a script which runs the model builder, cable model builder and solver with a single command. The script may be directed to use the sequential or the mpi solver. In order to use this script, the executable directory must be in the user's path (see section 2.4.) The script only requires that the input file (**NAME.inp)** be present.
The script is run using the following form for a sequential run:

run_GGI_TLM run_seq NAME

Where NAME is the name of the input file without the .inp extension.

Similarly the mpi solver is run using

run_GGI_TLM run_mpi np NAME

where np is an integer number of processes to run on.

The Python script **GGI_TLM_document.py** creates a document in **.rtf** format which provides a skeleton document describing the simulation and results. The script should be run from the directory in which the simulation has been run. The document contains the following sections:

- a model description (if provided in the input file)
- some views of the geometry including the boundary of the defined problem space
- details of the mesh (mesh cell size, number of cells, outer boundary reflection coefficients) and some views of the mesh
- Details of any volume materials including the frequency responses of dielectric and magnetic materials
- Details of any surface materials including the frequency responses of impedance boundary materials
- Runtime information: solver version and compilation date, timestep, number of timesteps, number of processes, the state of solver flags, time and date of the run, memory usage information, run time and runtime per timestep.
- Results as produced by the gnuplot file plot_result.plt
- An appendix containing the input file

The **.rtf** file can be viewed and edited using various software packages including word.

The **TEMPLATE_PROJECT** directory provides a template project which includes commands for automating the running of the codes. In order make use of this, copy (and rename) the directory to where you wish to run the new problem.

cp -r TEMPLATE_PROJECT NEW_PROJECT

2. Edit the file **run_automatic_test** in the directory **NEW_PROJECT** so as to set the path to the GGI_TLM executables. Example:

GGI_TLM_DIRECTORY=/home/chris/SOFTWARE/GGI_TLM/GGI_TLM

3. move or copy the TEMPLATE_PROBLEM directory to give it a new name:

mv TEMPLATE_PROBLEM NEW_TEST_CASE
or
cp -r TEMPLATE_PROBLEM NEW_TEST_CASE

Note that you can create as many individual test case directories as you like.

4. In the directory NEW**_TEST_CASE/PROBLEM_SPECIFICATION_FILES** create a GGI_TLM input file either by renaming and editing the existing file **problem_name.inp** or creating a new input file. Please note that there must be only one input file (.inp) in the **PROBLEM_SPECIFICATION_FILES** directory.

edit the files:

new_problem_name.inp (GGI_TLM input file)
GGI_TLM_post_process_in.txt (GGI_TLM post processor input file)
create_reference_result (file to save reference results)
plot_result.plt (gnuplot file to plot results)
plot_result_with_reference.plt (gnuplot file to plot results along with reference results)

Please ensure that the material, cable and mode files (*.vmat, *.smat, *.cable, *.mode) required to run the test case must either be in the PROBLEM_SPECIFICATION_FILES along with the input file or specified using the absolute path to the file. For example to make use of the MATERIAL_DATA or CABLE_DATA provided with the TEST_DATA, the material file names and cable file names must be specified in the form:

/home/chris/SOFTWARE/GGI_TLM/GGI_TLM/TEST_DATA/MATERIAL_DATA/Biological_sample_material

/home/chris/SOFTWARE/GGI_TLM/GGI_TLM/TEST_DATA/CABLE_DATA/RG58_coax

Note that the GGI_TLM_post_process_in.txt may be created by running the test case the post processing the data in the **NEW_TEST_CASE/** directory then copying the GGI_TLM_post_process_in.txt created into the **PROBLEM_SPECIFICATION_FILES** directory.

5. Edit the FULL_TEST_CASE_LIST so as to include the directories for each of the GGI_TLM test cases required.
This is not a necessary step but it will allow all the test cases specified to be run with a single command. Example:

```
FULL_TEST_CASE_LIST="
NEW_TEST_CASE
NEW_TEST_CASE2
NEW_TEST_CASE3
"
```

The TEMPLATE_PROBLEM directory structure may be used as a template for running other solutions using the full functionality of **run_automatic_test** as described in the test case documentation.

The script:

run_automatic_test action

is used to run the test cases, post process, view the results or plot results to file. Results may also be plotted alongside reference results.

action is the process to run i.e.

action=run_mpi np NAME	: run GGI_TLM_MPI on the test case NAME only using np processors where np is the number of processors (integer)
action=run_seq NAME	: run GGI_TLM_SEQ on the test case NAME only
action=post_process NAME	: post_process results from test case NAME
action=clean NAME	: remove all files from the run directory
action=clean_vtk NAME	:remove all vtk files from the run directory

action=clean_all NAME :remove all files from the run directory and any results saved in RESULTS_* directories using **run_automatic_test save**

action=plot NAME : plot the results to screen

action=plot_jpg NAME : plot the results to jpeg file

action=plot_ref NAME : plot the results against reference results to screen

action=plot_ref_jpg NAME : plot the results against reference results to jpeg file

action=reference NAME : update the **GGI_TLM** reference results with the current set of results

action=check_reference NAME : Check whether the **GGI_TLM** reference results are the same as the current set of results by direct comparison of the ascii output files. This is a fast way to check results but will flag up the smallest differences. If there is a problem indicated by **run_automatic_test check_reference** then it is best to go back to **run_automatic_test reference** to view the files i.e. to check whether there is a real problem or if the differences are in the numerical noise and due to running on a difference machine for example.

action=view_geometry NAME : View the geometry; volumes, surfaces, lines and points.

action=view_meshed_geometry NAME : View the meshed geometry; volumes, surfaces, lines and points

action=view_mesh NAME : View the computational mesh i.e. the mesh for each material plus the cable routes.

action=document NAME : Create a document in rtf format with details of the problem i.e. geometry, materials, cables, mesh and results

action=save NAME TAG : save the .inp, info files as well as result files and any rtf documents to a directory RESULTS_TAG in the test case directory. _TAG is a tag for a particular result. This is useful for running a number of problems with small differences based on the same core input file.

NAME can be one of the existing test cases or left blank to run the whole set

Parameterised geometry specification

It is sometimes useful to specify a geometry in terms of parameters, especially if a coordinate value (for example) appears several times in a geometry specification and may be a parameter under study. In this case the geometric values can be given a name and the value specified elsewhere.

The **run_automatic_test** script allows for a parameterised geometry specification which will be illustrated by an example.

The input file is allowed to include values specified as parameters e.g in the MONOPOLE_PARAMETERISED test case the monopole end point is specified by the parameters #end_point_x, #end_point_y and #end_point_z. The coordinate must be specified in two places, once to define the line end point and once to define the point coordinate for the wire end junction. the relevant extract from the input file is shown below:

```
Line_list
1          Number of lines
1          LINE NUMBER
straight_line2
0.0 0.0 0.0    #end_point_x  #end_point_y  #end_point_z
0.0 0.0 0.0
0.0 0.0 0.0

Point_list
     3      Number of points
     1     POINT_NUMBER
0.0 0.0 0.0         Point coordinates
0.0 0.0 0.0
0.0 0.0 0.0
     2     POINT NUMBER
 #end_point_x  #end_point_y  #end_point_z       Point coordinates
0.0 0.0 0.0
0.0 0.0 0.0
     3    POINT_NUMBER
2.0 2.0 1.5         Point coordinates
0.0 0.0 0.0
0.0 0.0 0.0
```

The parameter values are specified in the file **parameter_definition.dat** in the **PROBLEM_SPECIFICATION_FILES** directory. The **parameter_definition.dat** file takes the form of a list of the parameters and its value, separated by an '=' sign. One parameter specification is placed on each line as shown below:

```
#end_point_x=0.0
#end_point_y=0.0
#end_point_z=5.0
```

The parameter replacement is achieved in the **run_automatic_test** script and works by using **sed** command when copying the input file from the **PROBLEM_SPECIFICATION_FILES** directory into the run directory.

The substitution of parameters is done by a simple find and replace (using sed) therefore care must be taken in the naming of parameters and the order in which they are used. The following **parameter_definition.dat** file will cause a problem:

```
#dl=1.0
#dl2=0.5
```

In this case when '#dl' is substituted for its value throughout the input file, '#dl2' will become '1.02'. If parameters '#dl' and '#dl2' must be used then they should be specified in the order

```
#dl2=0.5
#dl=1.0
```

To avoid any problems.

It is possible to include derived parameters in the geometry for example in the MONPOLE_WITH_DERIVED_PARAMETERS test case. Here the **parameter_definition.dat** file is

```
#dl=1.0
#end_point_x=0.0
#end_point_y=0.0
#end_point_z=5.0
#obxy=[#end_point_z*2.0+#dl/2.0]
#obz=[#end_point_z*2.0]
```

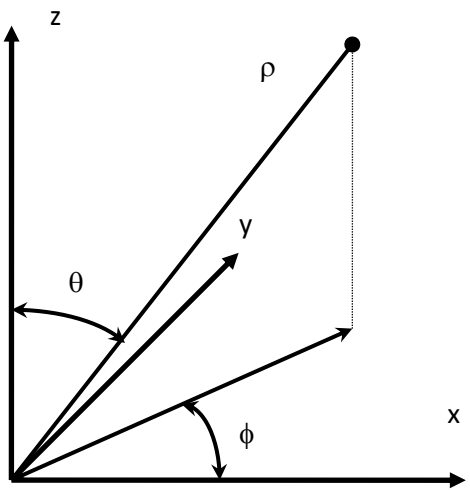
In this way the outer boundary position is a function of the monopole length and the cell size. The relevant part of the input file is:

```
Mesh outer boundary dimension
-#obxy  #obxy   -#obxy  #obxy   -#obz  #obz
```

4. Coordinate systems

GGI_TLM uses both Cartesian and spherical polar coordinate systems for the definition of different aspects of the geometry, excitation conditions and observed fields. All dimensions are in metres and angles are defined in degrees.

The Cartesian (x,y,z) and spherical polar (ρ,θ,ϕ) coordinates are related as shown in the figure below.



$$x = \rho \sin(\theta) \cos(\phi)$$
$$y = \rho \sin(\theta) \sin(\phi)$$
$$z = \rho \cos(\theta)$$

The specification of geometric objects (volumes, surfaces and lines) allows the transformation of the specified object by rotation and translation. The rotation is applied first followed by the translation.

The rotation is specified by three real numbers $\psi=\theta x$, $\theta=\theta y$, $\phi=\theta z$, which specify the angle to rotate the geometry around the x, y and z axes. The rotation is actually applied in the order: rotate in z then y then x.

The combination of the three rotations is given by

$$[A] = [B][C][D]$$

Where

$$[D]=\begin{bmatrix} cos\phi & sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[C]=\begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix}$$

$$[B]=\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\psi & sin\psi \\ 0 & -sin\psi & cos\psi \end{bmatrix}$$

[A] is then given by

$$a_{11} = cos\theta\, cos\phi$$
$$a_{12} = cos\theta\, sin\phi$$
$$a_{13} = -sin\phi$$
$$a_{21} = sin\psi\, sin\theta\, cos\phi - cos\psi\, sin\phi$$
$$a_{22} = sin\psi\, sin\,\theta sin\phi + cos\psi\, cos\phi$$
$$a_{23} = cos\,\theta sin\psi$$
$$a_{31} = cos\psi\, sin\,\theta cos\phi + sin\psi\, sin\phi$$
$$a_{32} = cos\psi\, sin\,\theta sin\phi - sin\psi\, cos\phi$$
$$a_{33} = cos\theta\, cos\psi$$

A translation is then specified by three (real) offsets in the x, y and z directions, ox, oy, oz.

4.1 Examples

Assume that we have defined a rectangular surface normal to z of width a in the x direction and width b in the y direction.

We can rotate this original surface to give a surface normal to x of width b in the y direction and width a in the z direction by the rotation:
θx=ψ=0
θy=θ=90
θz=ϕ=0

We can rotate this original surface to give a surface normal to y of width a in the x direction and width b in the z direction by the rotation:
θx=ψ=90
θy=θ=0
θz=ϕ=0

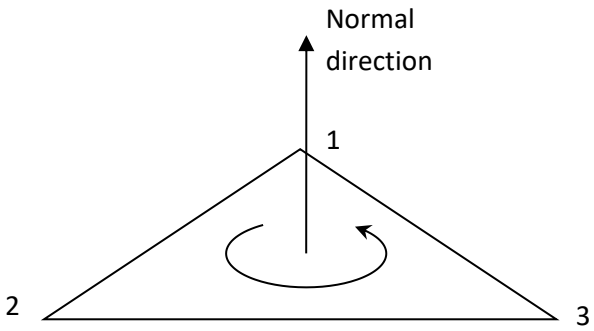
5. Geometry Specification

The problem geometry is defined as a set of volumes, surfaces, lines and points. The volumes may be given the properties of bulk dielectric/ magnetic materials, surfaces may be given impedance boundary properties (which include PEC and resistive sheets as special cases) and cable routes may be associated with sequences of lines through the problem spaces. Points are used to specify the location of outputs in the problem space as well as cable junctions (including terminations to surfaces.)

The geometry is built from a list of either simple geometric shapes such as spheres, cylinders, rectangles, circles, arcs etc or by specifying files containing data for tetrahedrally meshed volumes or triangulated surfaces. Within the code all volumes are constructed from tetrahedral meshes, all surfaces from triangulated meshes and all lines from a series of line segments thus the basis for mesh generation is common to all geometric entities.

As well as for defining the problem structure, the geometry specification may be used to specify a surface or volume for output. Surfaces may be used to define a Huygens surface for launching waves within the problem space or surfaces on which fields are collected and used to calculate far field patterns or radar cross section for example.

Geometric surfaces have a defined normal direction, this allows the different sides of a surface to be identified unambiguously. This is important for output on material surfaces for example in which fields are discontinuous across a surface, or for orientation of thin sheet models for non-symmetric materials. The normal direction for a surface is defined by the ordering of the triangles forming the surface description.



6. Mesh Generation

Line geometry is represented in GGI_TLM as a set of straight line segments, surface geometry is defined internally as a set of triangles and volume geometry is represented in the code as a set of tetrahedra. The GGI_TLM code runs on a structured, unform cartesian grid therefore the mesh generation process consists of projecting line segments, triangles and tetrahedral onto a Cartesian mesh.
There are a number of approaches to this geometry transformation, two are currently available in GGI_TLM.

The 'original' algorithm works as follows:

1. Line segments represent cable routes and meshed lines pass from cell centre to cell centre in the mesh. The line mesh generation algorithm works out the cells through which the line passes and then joins up the cell centres of these cells to give the meshed line.
2. Triangulated surfaces represent surface materials, triangles are meshed by projecting rays through the centres of mesh cells in the x, y and z directions. Where a ray intersects a triangle, the closest surface in the ray direction (x, y or z) becomes a part of the surface mesh.
3. Tetrahedral volumes are used to model volume materials. The volume mesh is the set of cells which have their centres inside the tetrahedron.

This mesh generation strategy works well in most cases but can on occasion cause problems such as holes in surface meshes, especially where triangle/ ray intersections lie equidistant from two TLM cell surfaces. In this instance, due to finite precision calculations, one ray/surface intersection may snap to one cell face whereas an adjacent ray/surface intersection may snap to the other cell face.

The 'new' algorithm works as follows:

1. Line segments represent cable routes and meshed lines pass from cell centre to cell centre in the mesh. The line mesh generation algorithm works out the cells through which the line passes and then joins up the cell centres of these cells to give the meshed line.
2. Triangulated surfaces represent surface materials, triangles are meshed by meshing the edges of the triangle to edges of TLM cells, then filling the enclosed cell faces in a manner which best fits the original triangle
3. Tetrahedral volumes are used to model volume materials. The volume mesh generated by meshing the four triangles enclosing the tetrahedron, then setting the volume mesh to be the set of cells enclosed

This strategy is somewhat complicated in its implementation but should be more robust than the original algorithm in that no holes should be present in the surface meshes. The new algorithm is now the default mesh generation technique for GGI_TLM.

The new mesh generation technique can be specified by including the line:

New_mesh_generation

In the input file.

7. Frequency dependent property specification

Materials, cable properties and impedances in the GGI_TLM code may be given frequency dependent properties. All frequency dependent properties are specified as rational functions of frequency. As an example, a frequency dependent impedance may be specified using the following form:

$$Z_{ij}(j\omega) = \frac{a_0 + a_1\left(j\frac{\omega}{\omega_0}\right) + a_2\left(j\frac{\omega}{\omega_0}\right)^2 + \ldots}{b_0 + b_1\left(j\frac{\omega}{\omega_0}\right) + b_2\left(j\frac{\omega}{\omega_0}\right)^2 + ..}$$

where the expansion may be to arbitrary order. In the above equation, ω_0 is a frequency normalisation constant which helps to prevent the expansion coefficients getting too large or small for high order functions.

Frequency dependent functions are specified in the code in the format shown in the following example:

```
# Z11 filter
75398223684.0000  # wnormalisation constant
2   # a order
   176.954882553514    0.0  1499.38774569197 # a coefficients
2   # b order
   1.00000000000000    0.0  3.94726504660646 # b coefficients
```

8. Volume Material Specification

All volume materials apart from Perfect Electrical Conductor (PEC) and Perfect Magnetic Conductor (PMC) materials are assumed to have frequency dependent properties. Volume material properties are specified in a material file with the extension **.vmat** which defines the frequency response of the material permittivity and permeability as a rational function in complex frequency. Also included are electric and magnetic conductivity terms.

Example

An example material file is shown below for a first order Debye dielectric material with constant magnetic properties

```
# debye dielectric model
0.0 1e8  # frequency range of validity
# permittivity filter
1.0e8 # wnormalisation constant
1   # a order
4.0 0.75 # a coefficients
1   # b order
1.0 0.50 # b coefficients
0.0  # electric conductivity
# permeability filter
1.0e8 # wnormalisation constant
0   # a order
1.0  # a coefficients
0   # b order
1.0  # b coefficients
0.0  # magnetic conductivity
```

9. Thin layer material models

Thin layer material models are divided into three categories:

- Isotropic thin layers
- Anisotropic thin layers
- Lumped component models

These will be described in the subsequent sections

9.1 Isotropic thin layer

Isotropic thin layers apart are assumed to have frequency dependent properties. Thin layers are represented by an impedance matrix which relates the orthogonal tangential electric and magnetic fields either side of the layer. For example for a layer normal to z we have:

$$\begin{bmatrix} E_{xl} \\ E_{xr} \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \begin{bmatrix} H_{yl} \\ H_{yr} \end{bmatrix}$$

And

$$\begin{bmatrix} E_{yl} \\ E_{yr} \end{bmatrix} = -\begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \begin{bmatrix} H_{xl} \\ H_{xr} \end{bmatrix}$$

Where the elements of the impedance matrix are frequency dependent functions.

We assume that there is no coupling between the orthogonal polarisations at a surface impedance boundary and that both polarisations see the same surface impedance.

Thin layer models are defined in a file with extension **.smat**.

9.1.1 Example

An example of an isotropic thin layer material definition file is shown below for an impedance boundary representation of a thin homogeneous conducting layer

```
Impedance sheet data
 0.000000000000000E+000   299251707.854092     # frequency range of validity
 Filter coefficients:         Z11
  1879300725.32370
        8
  2.50040212785157       454.893215171525       11670.5718399275
  97323.3862334169       335240.047895711       515526.784099386
  348135.557144271       91185.5275987549       6398.99890990202
        8
  1.00000000000000       56.0401259219104       782.422973955926
  4127.42531757411       9486.57419089856       9785.14659074041
  4265.68320424062       646.901413408572       19.4769289887590
 Filter coefficients:         Z12
```

1879300725.323708
2.49999649033029 -9.55465868956522 18.0042349540697
-22.0433756134523 19.4167681206317 -12.7677271190888
6.31051749582964 -2.17900452863388 0.494019932530020
1.00000000000000 59.1699410115711 958.338515063105
6652.82867895812 23787.1247684747 47714.8722886812
55091.1405036376 35027.9043415134 10062.8542607902
Filter coefficients: Z21
1879300725.323708
2.49999649033029 -9.55465868956522 18.0042349540697
-22.0433756134523 19.4167681206317 -12.7677271190888
6.31051749582964 -2.17900452863388 0.494019932530020
1.00000000000000 59.1699410115711 958.338515063105
6652.82867895812 23787.1247684747 47714.8722886812
55091.1405036376 35027.9043415134 10062.8542607902
Filter coefficients: Z22
1879300725.323708
2.50040212785157 454.893215171525 11670.5718399275
97323.3862334169 335240.047895711 515526.784099386
348135.557144271 91185.5275987549 6398.99890990202
1.00000000000000 56.0401259219104 782.422973955926
4127.42531757411 9486.57419089856 9785.14659074041
4265.68320424062 646.901413408572 19.4769289887590

9.2 Anisotropic thin layer

Anisotropic thin layers are also all assumed to have frequency dependent properties. Anisotropic thin layers are represented by an impedance matrix which relates the orthogonal tangential electric and magnetic fields either side of the layer. Three impedance matrices characterise the anisotropic layer, one for each Cartesian E-field component

For example for a layer normal to z we have:

$$\begin{bmatrix} E_{xl} \\ E_{xr} \end{bmatrix} = \begin{bmatrix} Z_{11x} & Z_{12x} \\ Z_{21x} & Z_{22x} \end{bmatrix} \begin{bmatrix} H_{yl} \\ H_{yr} \end{bmatrix}$$

And

$$\begin{bmatrix} E_{yl} \\ E_{yr} \end{bmatrix} = -\begin{bmatrix} Z_{11y} & Z_{12y} \\ Z_{21y} & Z_{22y} \end{bmatrix} \begin{bmatrix} H_{xl} \\ H_{xr} \end{bmatrix}$$

Where the elements of the impedance matrix are frequency dependent functions.

Similar conditions apply at layers normal to x and y.

We assume that there is no coupling between the orthogonal polarisations at an anisotropic surface impedance boundary.

Anisotropic thin layer models are defined in a file with extension **.smat**.

9.2.1 Example

An example of an anisotropic thin layer material definition file is shown below for an impedance boundary representation of a thin homogeneous conducting layer

```
# Anisotropic impedance sheet
0.0 1e8     # frequency range of validity
# Z11 filter: X polarisation
1.0e8       # wnormalisation constant
0           # a order
188.5  # a coefficients
0           # b order
1.0    # b coefficients
# Z12 filter: X polarisation
1.0e8       # wnormalisation constant
0           # a order
188.5  # a coefficients
0           # b order
1.0    # b coefficients
# Z21 filter: X polarisation
1.0e8       # wnormalisation constant
0           # a order
188.5  # a coefficients
0           # b order
1.0    # b coefficients
# Z22 filter: X polarisation
1.0e8       # wnormalisation constant
0           # a order
188.5  # a coefficients
0           # b order
1.0    # b coefficients
# Z11 filter: Y polarisation
1.0e8       # wnormalisation constant
0           # a order
377.0  # a coefficients
0           # b order
1.0    # b coefficients
# Z12 filter: Y polarisation
1.0e8       # wnormalisation constant
0           # a order
377.0  # a coefficients
0           # b order
1.0    # b coefficients
# Z21 filter: Y polarisation
1.0e8       # wnormalisation constant
0           # a order
377.0  # a coefficients
0           # b order
1.0    # b coefficients
# Z22 filter: Y polarisation
1.0e8       # wnormalisation constant
```

```
0           # a order
377.0  # a coefficients
0           # b order
1.0    # b coefficients
# Z11 filter: Z polarisation
1.0e8       # wnormalisation constant
0           # a order
754.0  # a coefficients
0           # b order
1.0    # b coefficients
# Z12 filter: Z polarisation
1.0e8       # wnormalisation constant
0           # a order
754.0  # a coefficients
0           # b order
1.0    # b coefficients
# Z21 filter: Z polarisation
1.0e8       # wnormalisation constant
0           # a order
754.0  # a coefficients
0           # b order
1.0    # b coefficients
# Z22 filter: Z polarisation
1.0e8       # wnormalisation constant
0           # a order
754.0  # a coefficients
0           # b order
1.0    # b coefficients
```

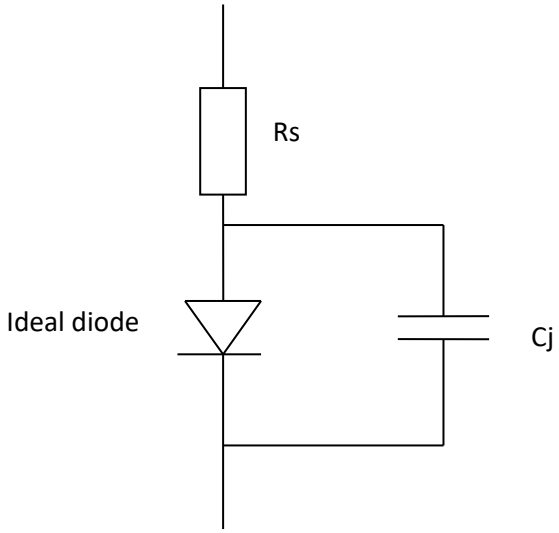
9.3 Lumped component models

In addition to the impedance boundary models a lumped component model may be placed on surfaces between TLM cells. Currently the only lumped component model implemented within GGI_TLM is a diode model. This may be used to represent a surface mount diode on a PCB track for example. The component has a direction associated with it hence it is an anisotropic model which interacts with a single E field polarisation on a surface, The E field in the orthogonal direction on the surface is assumed not to see any additional impedance and the connection process occurs as in free space.

In more recent developments, Ngspice component models may linked into GGI_TLM. This is described in the document **GGI_TLM_Ngspice_documentaion**.

9.3.1 Diode model

The diode equivalent circuit is shown in figure 8.3.1.1 below.



The model consists of an ideal diode in which the diode current is given by the expression

$$I_D = I_s\left(e^{\frac{V_d}{nVt}} - 1\right)$$

The ideal diode is in parallel with a junction capacitance (C_j) and the combination is in series with a resistance (R_s).

More details are available in the GGI_TLM_matrial_model_theory document.

Cable Specification

Within **GGI_TLM** cables are defined to be systems of conductors which run through the problem space without changing their cross section configuration along their length. A cable may have one conductor such as a simple cylindrical wire, or multiple conductors such as a coaxial cable or a ribbon cable. Cables may include shielded conductors. Cables must be terminated at both ends to a defined junction. The end of a cable may be connected to other cable(s) or terminated to a surface in the mesh. A termination to a surface should be situated at a TLM cell face.

Boundary conditions may be applied at a junction which allow conductors to be electrically connected to surfaces as well as being connected amongst themselves. A termination at a surface allows cables to be connected from either side of the cell surface thus for example a coaxial cable may have its shield terminated to a ground plane while the inner conductor is allowed to pass through. In this way coaxial feeds trough ground planes can be modelled.

A junction is normally situated at a cell centre. At a junction, conductors from the six faces belonging to the cell may be connected in an arbitrary manner so that complex networks can be formed.

Lumped loads and sources may be placed at the ends of cables i.e. at junctions.

Bundles are formed when more than one cable runs along the same path through the mesh.

Complex wiring within a model is specified in a three stage manner. Firstly a number of lines are specified with the problem space in the input file (figure 10.1.) These lines are then put onto the structured mesh automatically (figure 10.2.) by the **GGI_TLM_model_builder**. Meshed lines run between cell centres on the mesh. It should be noted that lines which run through the same cell in the structured mesh will be snapped together. Nodes of the cable network model are formed at cell centres and at points where lines intersect cell faces. Where lines meet at a cell centre a junction is formed.

The meshed line structure defines a cable network consisting of nodes connected by segments. The network will contain the junctions and terminations defined in the input file and also junctions formed by the meshing process as a result of changes in composition of the cable bundles formed by snapping together of the individual cables.

Cables are defined with a particular cross section which may include multiple cores and screens so each cable may be a multi-conductor. A cable route through the problem space is defined as a sequence of lines (figure 10.3.) The lines must be continuous. A termination condition must be specified at each end of the cable. This may take the form of an open circuit condition as in the ends of a dipole, a termination to a surface via a lumped load and source or a connection to a junction, again incorporating lumped elements and sources.

The cables running along each line segment in the mesh are combined to form bundles of cables (figure 10.4.) The cable bundles on the 1D cable bundle mesh are the basis for the computation.

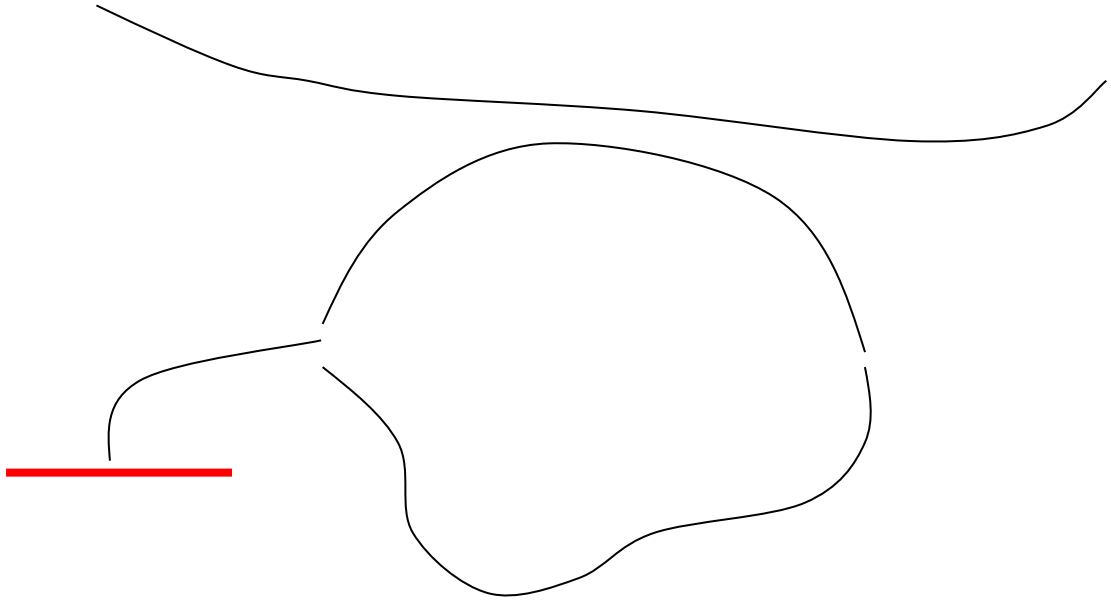


Figure 10.1 Initial line geometry

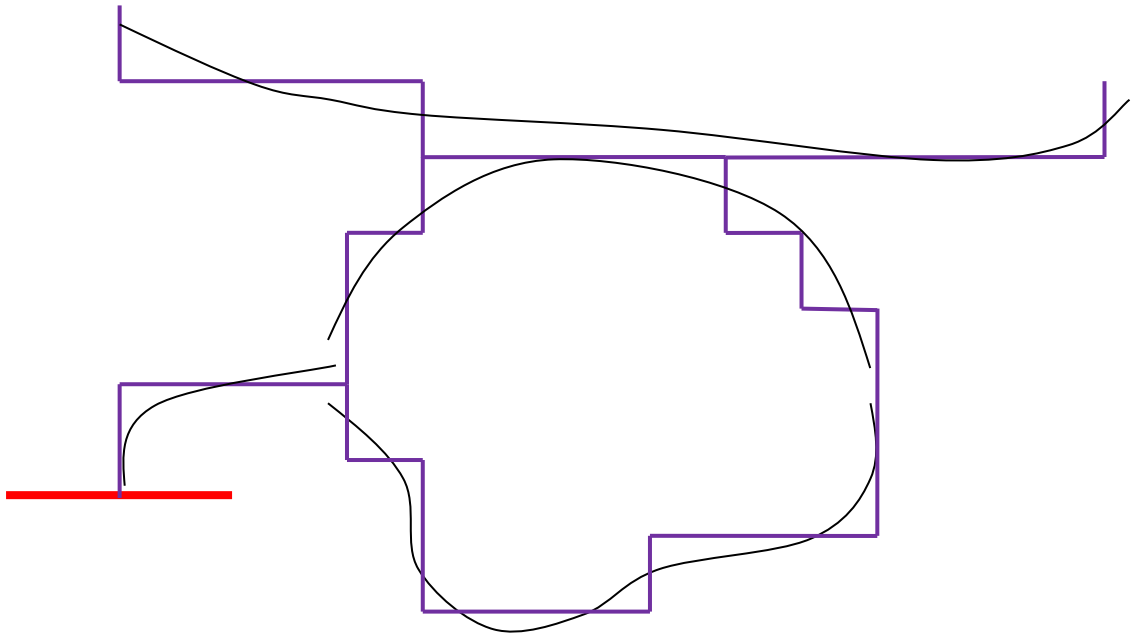


Figure 10.2.Create structured line geometry forming wire routes

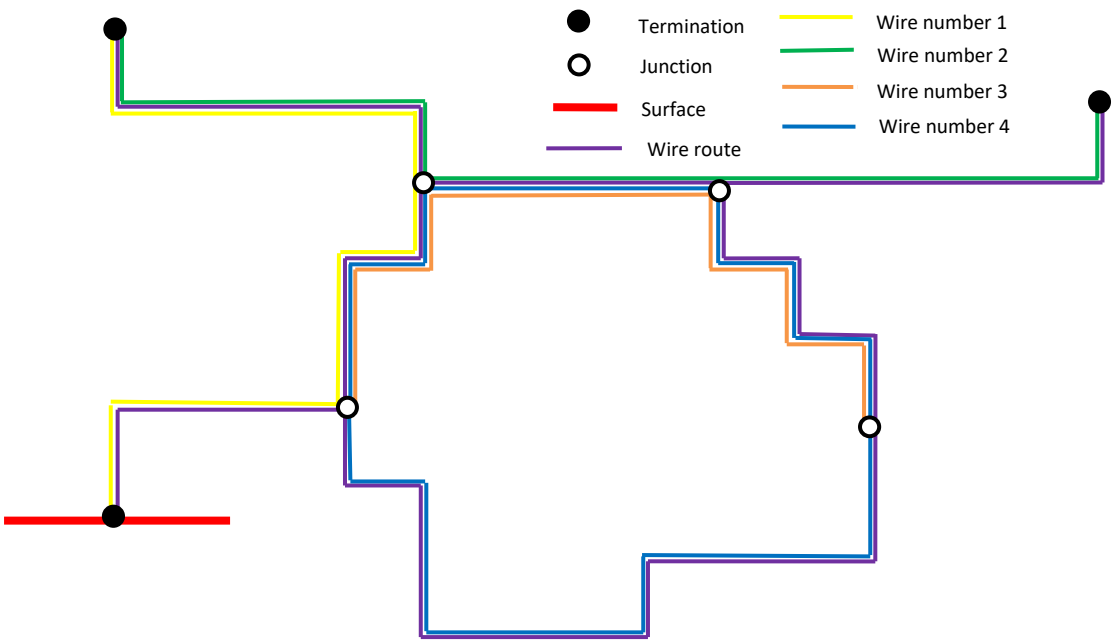


Figure 10.3. Cables follow the wire routes forming bundles.

Individual cables are assembled into bundles and the bundle parameters are calculated in the EM solver code.

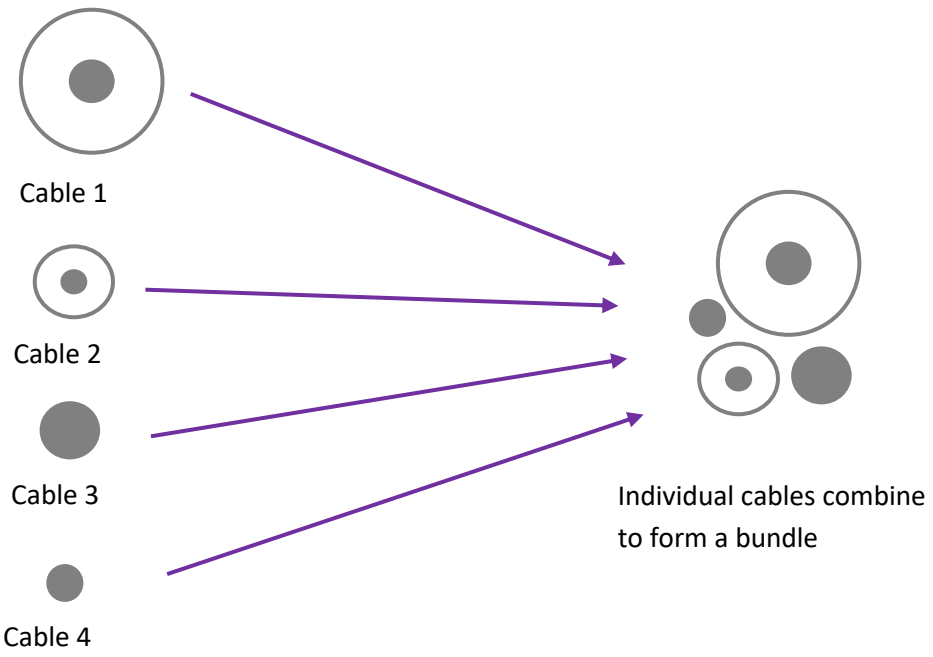


Figure 10.4. Individual cables are combined to form bundles.

The individual cable characteristics are defined in a **.cable** file in which the cable characteristics are defined.

The available cable types and their required parameters are as follows:

Cylindrical: single cylindrical conductor.

Parameters: radius of conductor, radius of insulation, relative permittivity of insulation

FD_Cylindrical: single cylindrical conductor with frequency dependent series impedance.

Parameters: radius of conductor, radius of insulation, relative permittivity of insulation

In addition to these parameters a single filter description specifies the series impedance.

```
FD_cylindrical
1          # number of conductors
3          # NUMBER OF PARAMETERS
0.001      # parameter 1: conductor radius
0.002      # parameter 2: insulation radius
1.0        # parameter 3: permittivity of outer dielectric
1          # NUMBER OF FILTERS
# Filter 1: Wire impedance filter
1.0e8      # w normalisation constant
0          # a order
2.5        # a coefficients
0          # b order
1.0        # b coefficients
```

Coaxial: cylindrical coaxial cable with dielectric between wire and shield, including $Z_t=R_t+j\omega L_t$ transfer impedance model.

Parameters: shield radius, outer insulation radius, permittivity of outer dielectric, inner radius, permittivity of inner dielectric, per-unit-length transfer inductance L_t, per-unit-length transfer resistance, R_t

```
coaxial
2          # number of conductors
7          # NUMBER OF PARAMETERS
0.00147    # parameter 1: shield radius
0.00147    # parameter 2: insulation radius
1.0        # parameter 3: permittivity of outer dielectric
0.00042    # parameter 4: inner radius
2.25       # parameter 5: permittivity of inner dielectric
6.59e-10   # parameter 6: Transfer inductance, Lt
0.0987     # parameter 7: Transfer resistance, Rt
0          # NUMBER OF FILTERS
```

FD_Coaxial: cylindrical coaxial cable with dielectric between wire and shield, including general frequency dependent transfer impedance model and frequency dependent impedance on both conductors.

Parameters: shield radius, outer insulation radius, permittivity of outer dielectric, inner radius, permittivity of inner dielectric. The transfer impedance and series impedance for inner and outer conductor are specified as filter functions i.e. three filter functions in total.

```
FD_coaxial
2          # number of conductors
5          # NUMBER OF PARAMETERS
0.00147    # parameter 1: shield radius
0.00147    # parameter 2: insulation radius
1.0        # parameter 3: permittivity of outer dielectric
0.00042    # parameter 4: inner radius
2.25       # parameter 5: permittivity of inner dielectric
3          # NUMBER OF FILTERS
# Filter 1: Inner wire series impedance filter
1.0e0           # w normalisation constant
0          # a order
0.0        # a coefficients
0          # b order
1.0        # b coefficients
# Filter 2: Shield series impedance filter
1.0e0           # w normalisation constant
0          # a order
0.0        # a coefficients
0          # b order
1.0        # b coefficients
# Filter 3: Transfer impedance filter
1.0e0           # w normalisation constant
1          # a order
0.0987 6.59e-10  # a coefficients
0          # b order
1.0        # b coefficients
```

Ribbon_cable: multiple conductor ribbon cable.

Parameters: number of conductors, radius of conductor, radius of insulation, relative permittivity of insulation,conductor separation.

```
ribbon_cable
3              # number of conductors
4              # NUMBER OF PARAMETERS
1.905e-4       # parameter 1: conductor radius
2.56e-4        # parameter 2: insulation radius
3.5            # parameter 3: permittivity of dielectric
1.27e-3        # parameter 4: conductor separation
0              # NUMBER OF FILTERS
```

FD_Ribbon_cable: multiple conductor ribbon cable with frequency dependent series impedance.

Parameters: number of conductors, radius of conductor, radius of insulation, relative permittivity of insulation,conductor separation. The series impedance of the conductors is specified as a filter function.

```
FD_ribbon_cable
3               # number of conductors
4               # NUMBER OF PARAMETERS
1.905e-4        # parameter 1: conductor radius
2.56e-4         # parameter 2: insulation radius
3.5             # parameter 3: permittivity of dielectric
1.27e-3         # parameter 4: conductor separation
1               # NUMBER OF FILTERS
# Filter 1: Wire impedance filter
1.0e8           # w normalisation constant
0               # a order
0.1944e0        # a coefficients
0               # b order
1.0             # b coefficients
```

Examples

Single conductor cylindrical wire

```
cylindrical
1          # number of conductors
3          # NUMBER OF PARAMETERS
0.0005     # parameter 1: conductor radius
0.0005     # parameter 2: insulation radius
1.0        # parameter 3: permittivity of outer dielectric
0          # NUMBER OF FILTERS
```

Frequency dependent cylindrical wire

```
FD_cylindrical
1          # number of conductors
3          # NUMBER OF PARAMETERS
0.001           # parameter 1: conductor radius
0.002           # parameter 2: insulation radius
1.0        # parameter 3: permittivity of outer dielectric
1          # NUMBER OF FILTERS
# Filter 1: Wire impedance filter
1.0e8           # w normalisation constant
0          # a order
2.5        # a coefficients
0          # b order
1.0        # b coefficients
```

Simple coaxial cable model with constant Rt+jωLt transfer impedance model

```
coaxial
2          # number of conductors
7          # NUMBER OF PARAMETERS
0.00147    # parameter 1: shield radius
0.00147    # parameter 2: insulation radius
1.0        # parameter 3: permittivity of outer dielectric
```

```
0.00042    # parameter 4: inner radius
2.25            # parameter 5: permittivity of inner dielectric
6.59e-10   # parameter 6: Transfer resistance, Rt
0.0987          # parameter 7: Transfer inductance, Lt
0          # NUMBER OF FILTERS
```

Advanced coaxial cable model with transfer impedance specified by a filter function

```
FD_coaxial
2          # number of conductors
5          # NUMBER OF PARAMETERS
0.00147    # parameter 1: shield radius
0.00147    # parameter 2: insulation radius
1.0        # parameter 3: permittivity of outer dielectric
0.00042    # parameter 4: inner radius
2.25            # parameter 5: permittivity of inner dielectric
3          # NUMBER OF FILTERS
# Filter 1: Inner wire series impedance filter
1.0e0           # w normalisation constant
0          # a order
0.0        # a coefficients
0          # b order
1.0        # b coefficients
# Filter 2: Shield series impedance filter
1.0e0           # w normalisation constant
0          # a order
0.0        # a coefficients
0          # b order
1.0        # b coefficients
# Filter 3: Transfer impedance filter
1.0e0           # w normalisation constant
1          # a order
0.0987 6.59e-10  # a coefficients
0          # b order
1.0        # b coefficients
```

9.4 Cable junctions

The connectivity of conductors at terminations and junctions can be complex. In addition, at terminations a 'V=0' boundary condition may be applied to indicate connection of conductors to a surface in the mesh. The connectivity of conductors is described by first setting up a number of internal connection nodes at the junction/ termination. A permutation matrix (P) then describes how individual conductors within a cable are connected to the internal connection nodes.

The number of rows in the permutation matrix is the number of internal connection nodes, the total number of columns is the total number of conductors incident on the junction. Permutation matrices may therefore be quite large if a number of cables are connected at a junction. In order to simplify the specification of P matrices they are defined on a cable by cable basis and the full permutation matrix for a junction is built within **GGI_TLM**. The elements of a P matrix defined on a cable by cable basis are defined as:

P_{ij}=1 if conductor j connects to internal connection node i.

P_{ij}=0 otherwise

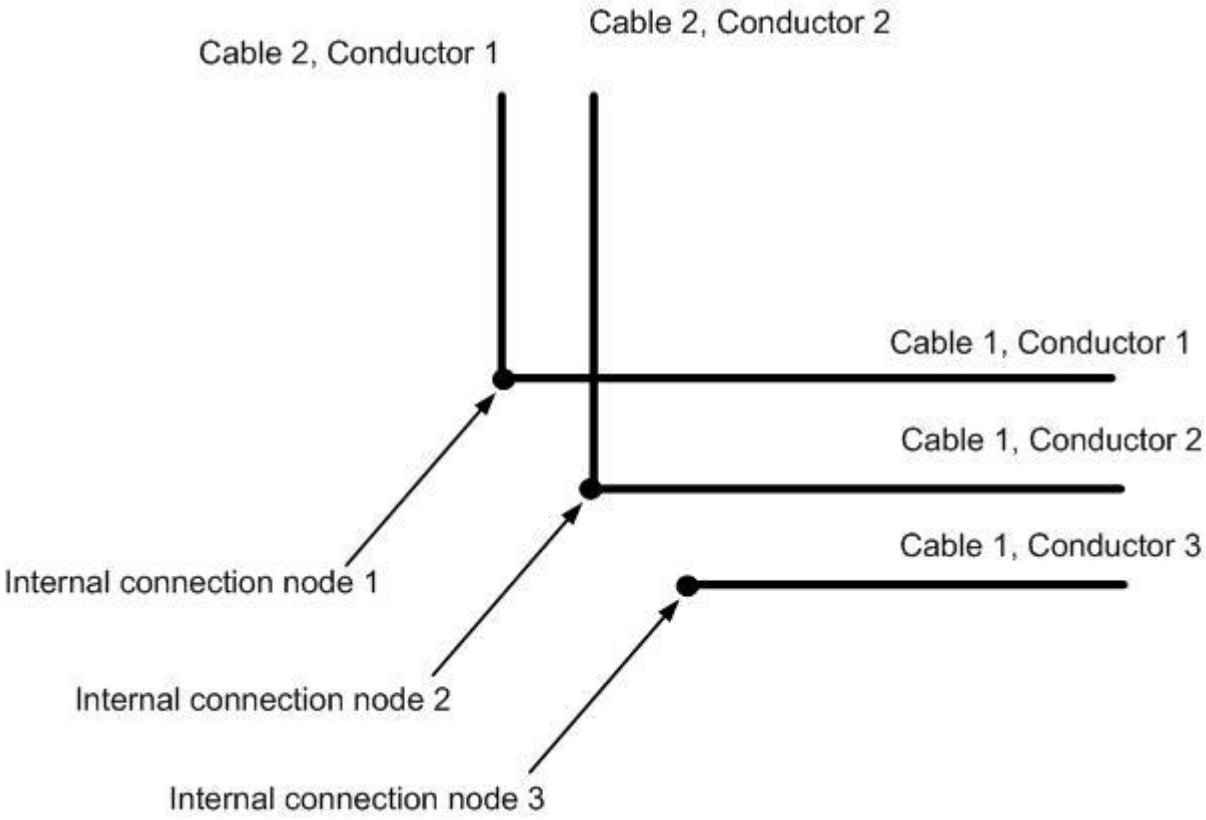


Figure 10.5. Normal junction

For the junction in figure 10.5 above there are two P matrices one for each of the two cables.

$$P_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

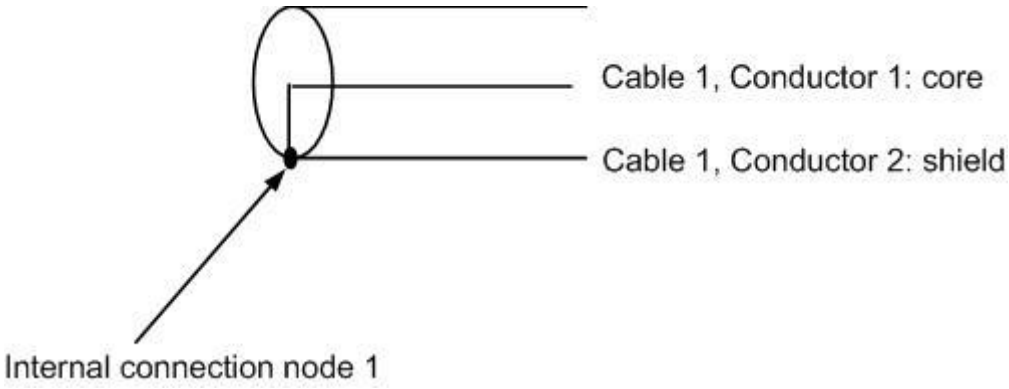


Figure 10.6 Coaxial cable termination.

In figure 10.6 a coaxial cable is terminated by a short circuit. For this case there is a single P matrix

$$P_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

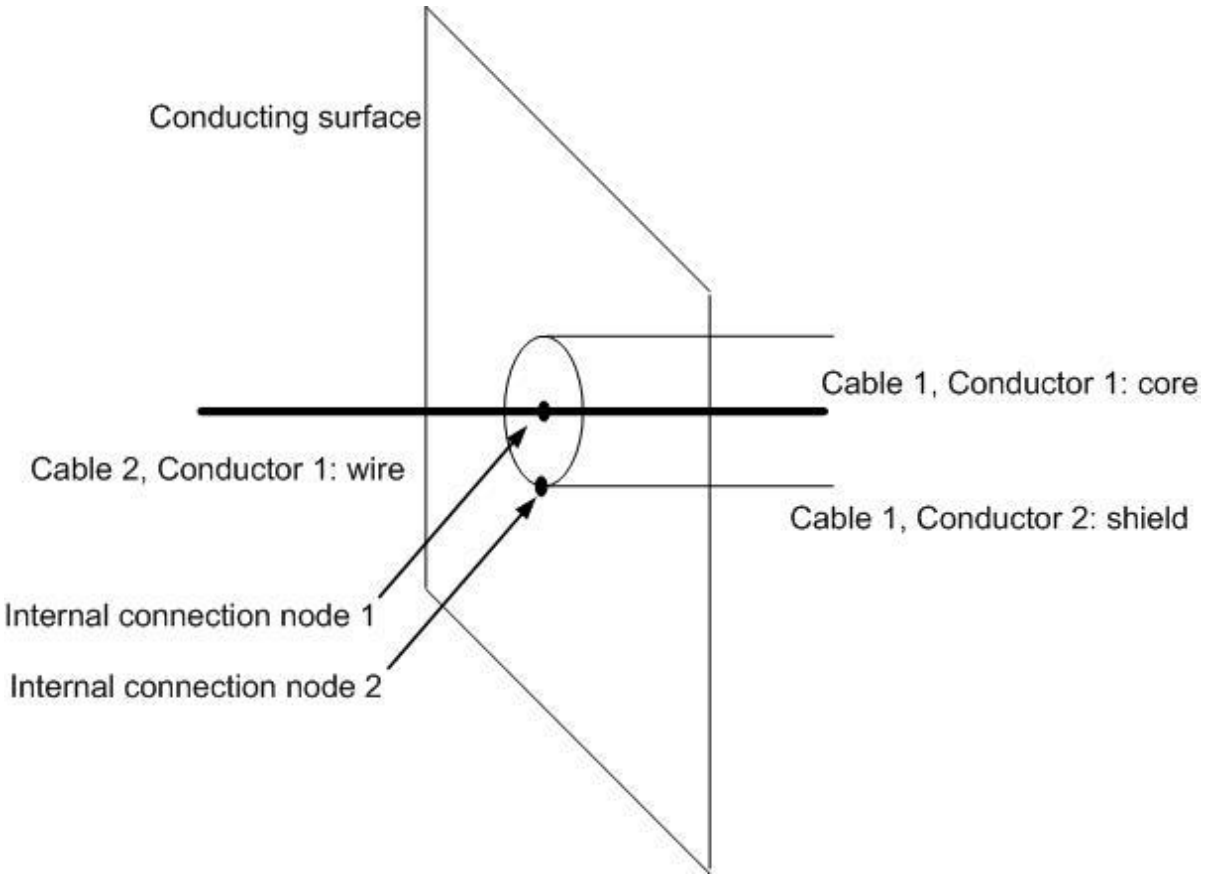


Figure 10.7. Coax to monopole transition

The P matrices to specify the coax to monopole transition in figure 10.7 are as follows

$$P_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

In this case the junction should be specified to be situated on a cell face since we are connecting to a surface. A boundary condition must also be set on internal connection node 2 as described in the Cable_junction_list packet data.

10. Geometry and mesh visualisation

The **GGI_TLM** model, cable and material check programs generate files which allow the visualisation of the geometry and the mesh for a particular problem. The visualisation files are in vtk format which may be viewed using **paraview**. The files generated for visualisation are:

NAME.v_geom.vtk.n: Volume number n defined in the input file
NAME.s_geom.vtk.n: Surface number n defined in the input file
NAME.l_geom.vtk.n: Line number n defined in the input file
NAME.p_geom.vtk.n: Point number n defined in the input file

NAME.v_mesh.vtk.n: Volume mesh, volume number n
NAME.s_mesh.vtk.n: Surface mesh, surface number n
NAME.l_mesh.vtk.n: line mesh, line number n
NAME.p_mesh.vtk.n: point mesh, point number n

NAME.b_mesh.vtk: Outer Boundary mesh

NAME.vmat.vtk.n: Volume material mesh, volume material number n
NAME.smat.vtk.n: Surface material mesh, surface material number n

NAME.c_mesh.vtk.n: cable mesh, cable number n.

Any number of geometric entities may be viewed simultaneously. Animated results may also be visualised with the geometry.

It is not within the scope of this document to describe all the capabilities of paraview however a short overview in order to get started is provided.

Paraview is started with the command

Paraview

This brings up the main paraview window as seen in the figure below.

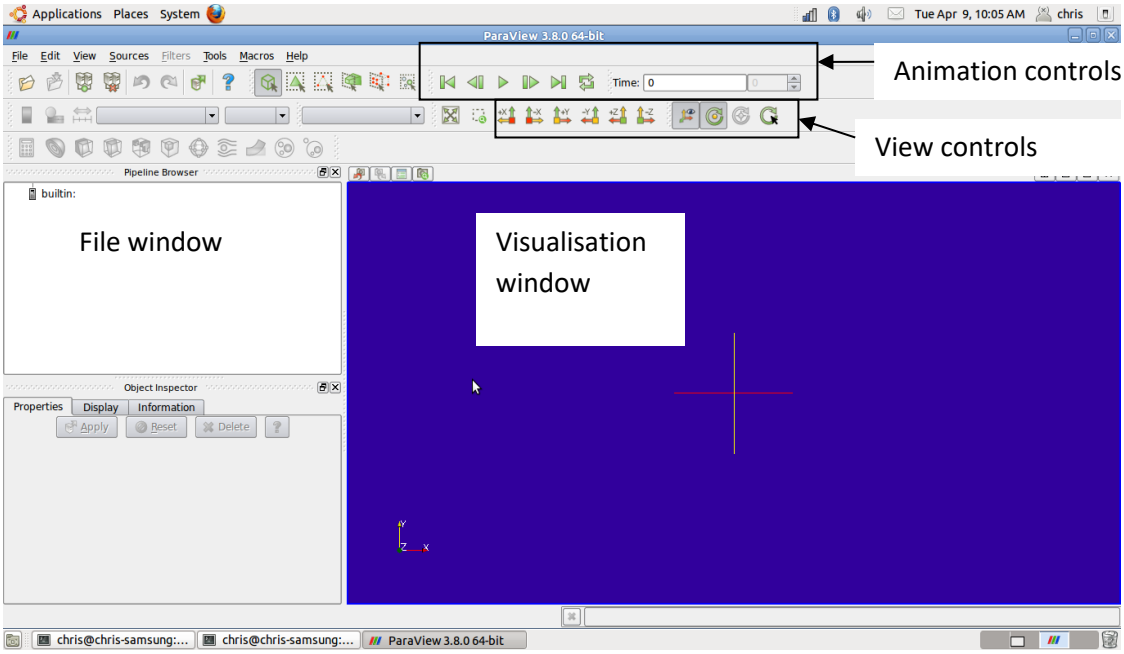


Figure 11.1. Main paraview screen.

Clicking on the 'open file' icon brings up a window showing the available files. This is seen in figure 11.2 for the dipole example.

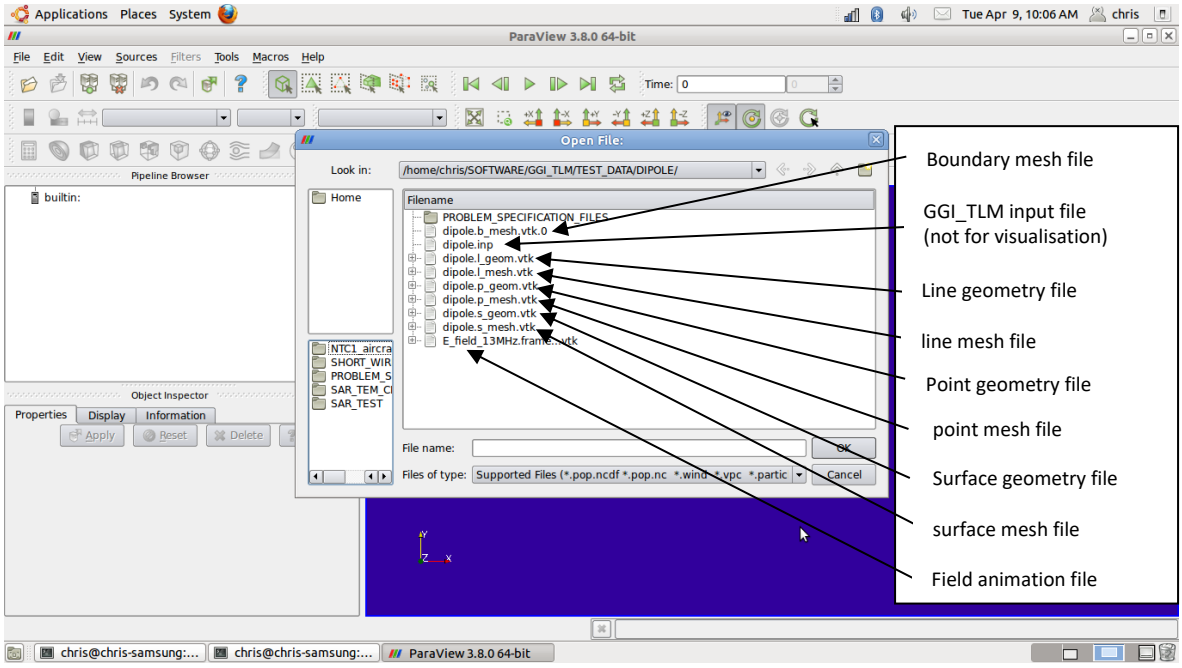


Figure 11.2. Paraview 'open file' window.

Once the files to view have been opened, the options for visualisation of each object can be specified as seen in figure 11.3 in which the boundary of the problem space is shown as a wireframe, the line meshes constituting the dipole arms are plotted as surfaces and a frame of the vector E field animation is shown.
Note that surface and volume visualisation files must be opened individually (click on the cross on the left hand side of the geom.vtk or mesh.vtk filename to expand the list). Animation files must be

loaded together i.e. don't click on the cross to expand the list as this will produce a list of the individual frames in the animation, rather click on the name of the list as in figure 11.2.

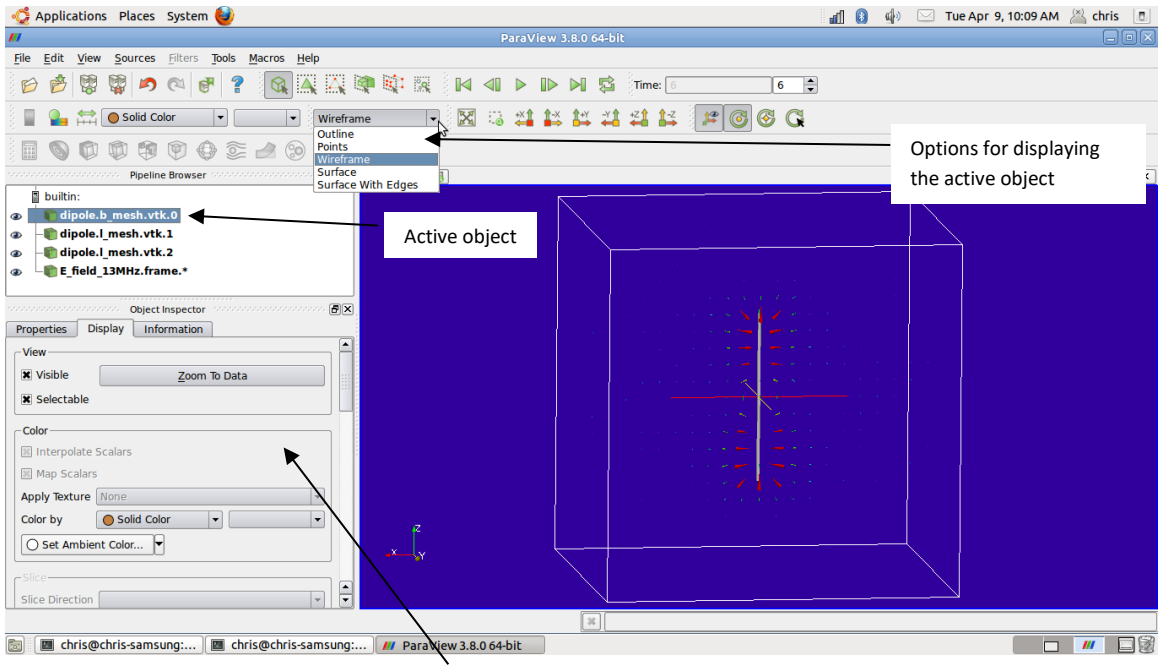


Figure 11.3. Specifying options for displaying objects.

In order to see objects within or behind other objects it may be useful to use the view controls to specify 'wire frame' or alternatively objects may be made semi-transparent (see the Opacity control in the display information). Changing an object's colour may also help to distinguish between parts of a complex geometry.

11. Parallel Implementation

GGI_TLM may be run in parallel. The parallel implementation uses mpi. The strategy is simply to decompose the mesh into np equal partitions in the z direction only thus the parallel implementation is not optimum by any means but is effective in many cases for moderate size problems. Cable and material models are included in the parallel implementation, each processor owns the cable and material cells which happen to be in its domain. No attempt is made to load balance to take into account the extra work involved in the cable and material updates at the moment. If a geometry is long and thin then it will always be advantageous to define the longest dimension to be in the z direction in the mesh.

A process manager such as mpd will have to be set up on the system (and started up using mpdboot for example) before mpi jobs can be run.

12. Input file format

The input file is packet based. Lines in the input file that are not understood as packets will be ignored. Once a packet name has been read the packet data must follow in the correct format for that packet or an error will occur.

The input file may start with a description of the simulation. This description should start with the line:

#START OF DESCRIPTION

and end with the line

#END OF DESCRIPTION

If the automatic documentation is used then this description will be copied into the simulation document .

There are no restrictions on the order in which packets should be specified.

Below is a list of the available packets understood by **GGI_TLM**. Details of the information required for each packet follows in subsequent sections along with an example packet references to test cases in which the packet is used in practice.

The input file also allows some of the solution default parameters to be specified by the user. These parameters should not normally need to be specified however in some circumstances the default parameters can lead to problems such as overflow in the per-unit-length inductance and capacitance matrix calculation for very small diameter cables (compared to the cell dimension).

12.1 GGI_TLM Packet list:

Mesh_outer_boundary_dimension
Mesh_dimensions_in_cells
Mesh_cell_dimension
Outer_boundary_reflection_coefficient
PML
Volume_list
Surface_list
Line_list
Point_list
Cable_geometry_list
Cable_list
Cable_junction_list
Cable_output_list
Volume_material_list
Surface_material_list
Excitation_function_list
Excitation_point_list
Excitation_surface_list

Excitation_mode_list
Huygens_surface
Far_field_surface
RCS_surface
Output_point_list
Frequency_domain_power_surface_list
Frequency_output_surface_list
Frequency_output_volume_list
New_mesh_generation
Output_surface_list
Output_volume_list
Output_volume_average_list
Output_mode_list
SAR_volume_list
Simulation_time
Mode_stir_surface_list
Random_number_seed
Frequency_scale_flag
Bicubic_warp_flag
Wrapping_boundary_conditions
Periodic_boundary
Periodic_boundary_far_field_surface
LC_correction_type_subtract_cell_inductance
LC_correction_type_geometry_scale
No_geometry_vtk_files
compress_output_files
reduced_c_factor

12.2 GGI_TLM Parameter list:

Number_of_fourier_terms_in_pul_lc_calc
TLM_equivalent_radius_factor
Capacitance_ equivalent_radius_factor
Inductance_ equivalent_radius_factor
Max_cable_bundle_diameter_factor

12.3 Mesh_outer_boundary_dimension

Defines the problem space dimensions. The packet requires the minimum and maximum coordinates in each of the coordinate directions i.e.

xmin xmax ymin ymax zmin zmax

example

Mesh_outer_boundary_dimension
-4.0 4.0 -4.0 4.0 -4.0 4.0

Example test cases:
All test cases.

12.4 Mesh_dimensions_in_cells

Defines the problem dimensions in terms of number of Cartesian cells in the x, y and z directions i.e.nx ny nz. This is an alternative to setting the mesh cell dimension.

example

Mesh_dimensions_in_cells
64 64 64 (3*integer)

Example test cases:
CAVITY_FREE_SPACE
COAX_COUPLING

12.5 Mesh_cell_dimension

Set the mesh cell dimension in metres. This is an alternative to setting the mesh dimension in cells.

Example
To set a 1cm grid:
Mesh_cell_dimension
0.01

Example test cases:
AIRCRAFT_SURFACE_CURRENT
DIELECTRIC_FILTER

12.6 Outer_boundary_reflection_coefficient

Specify the reflection coefficients to be applied on the xmin, xmax, ymin, ymax, zmin and zmax boundaries. Each reflection coefficient should be a real number between -1 and 1.

Examples

Outer_boundary_reflection_coefficient
0.0 0.0 0.0 0.0 0.0 0.0 ! absorbing boundaries everywhere

Outer_boundary_reflection_coefficient
-1.0 -1.0 +1.0 +1.0 0.0 0.0 ! Electric walls on xmin and xmax, Magnetic walls on ymin and ymax,
! absorbing boundaries on zmin and zmax

Example test cases:
TEM_MATERIAL_TEST
RESONANT_ABSORBER

12.7 PML

The perfectly matched layer is an absorbing boundary volume which is designed to absorb outgoing waves irrespective of angle of incidence and polarisation. It out-performs the usual TLM matched boundary. A PML consists of layers of TLM cells which run a variation of the normal scattering process which provides the wide angle matching and loss. Normally the PML is specified by the thickness of the layer, t, the order of the loss grading profile, o, and a parameter which controls the reflection coefficient of the PML, r.

The order of the loss profile is an integer where 1 indicates a linear increase of loss into the layer, 2 indicates a quadratic loss profile etc. The reflection coefficient parameter provides an estimate of the reflection coefficient of the layer when backed with a perfect conductor.

The choice of t, o and r are a compromise between performance and simulation overhead (memory, runtime). Typically 5 or more layers would be used ($t \geq 5\Delta l$), a second order conductivity profile is found to be effective (o=2), and the reflection coefficient r=0.0001 is also a good starting point for optimising a PML.

In GGI_TLM a PML layer is built within the defined outer boundary as defined by the Mesh_outer_boundary_dimension packet so increasing the thickness of the PML may require the outer boundary dimension to be increased. PML layers may be placed on any or all of the outer boundaries of the mesh (xmin,xmax,ymin,ymax,zmin,zmax) and the parameters may be specified individually for each boundary as in example 1 below or for all boundary surfaces together as in example 2. Setting a thickness of zero on a surface indicates that no PML should be placed on that surface.

Examples

```
PML
4.0  4.0   4.0  4.0   4.0 4.0   ! PML thickness on each outer boundary surfaces: xmin,xmax,ymin,ymax,zmin,zmax
1e-4 1e-4  1e-4 1e-4  1e-4 1e-4 ! PML reflection_coefficient on each outer boundary surface:
2                                ! PML order
```

```
PML
4.0   ! PML thickness on all outer boundary surfaces
1e-4  ! PML reflection_coefficient on all outer boundary surfaces
2     ! PML order
```

Example test cases:
WAVEGUIDE_PML

12.8 Volume_list

Volumes within the problem space are specified in the volume_list packet. A number of simple geometric volumes such as rectangular blocks or spheres may be specified by appropriate parameters or alternatively a file with a volume tetrahedral mesh may be specified.
Each volume has a transformation associated with it which allows rotation and translation of the specified volume as described in section 3. The translation is described by six parameters in two sets of three on separate lines of the file.

θx, θy, θz
ox, oy, oz

The available volume types and their parameters are as follows:

Name:
Rectangular_block
Description:
Rectangular block centred on the origin
Parameters:
Length in x, length in y, length in z
θx, θy, θz
ox, oy, oz

Name:
Rectangular_block2
Description:
Rectangular block specified by its xmin, ymin, zmin and xmax, ymax and zmax coordinates
Parameters:
Xmin ymin zmin xmax ymax zmax
θx, θy, θz
ox, oy, oz

Name:
cylinder
Description:
Circular cylinder centred on the origin. The cylinder is circular in the xy plane and extruded in the ±z direction to the total length specified.
Parameters:
Radius, length (in z direction)
θx, θy, θz
ox, oy, oz

Name:
sphere
Description:
Sphere centred on the origin
Parameters:
Radius
θx, θy, θz
ox, oy, oz

Name:
tet
Description:
Tetrahedron defined by 4 points in 3-D space
Parameters:
x1 y1 z1 x2 y1 z2 x3 y3 z3 x4 y4 z4
θx, θy, θz
ox, oy, oz
Name:
Pyramid
Description:

Square pyramid
Parameters:
half_side_length of base (m), Height (m)
θx, θy, θz
ox, oy, oz

Name:
Pyramid_ram
Description:
Square pyramid with square base as used in pyramid RAM material
Parameters:
half_side_length of base (m), Height (m), base height (m)
θx, θy, θz
ox, oy, oz

Name:
Tet_volume_mesh
Description:
Tetrahedral volume mesh
Parameters:
Volume_filename
Scale_factor
θx, θy, θz
ox, oy, oz

Name:
Fill_surface
Description:
Volume created by filling a surface. The surface should be defined in the surface_list packet.
Parameters:
Surface_number (integer)
θx, θy, θz
ox, oy, oz

example

```
volume_list
1   Number of volumes
1      volume_number
rectangular_block
0.3e-3   10.0e-3   0.3e-3   ! volume parameters
0.0 0.0 0.0                  ! Euler angles
0.0 0.0 10.15e-3             ! offset translation
2     volume_number
sphere
1.0      volume parameter:  radius
0.0 0.0 0.0
0.0 0.0 0.0
3      Volume_number (integer)
```

cylinder
1.0 2.0   Volume parameters, radius and height
0.0 0.0 0.0
-0.5 0.3 0.25

Example test cases:
DIELECTRIC_FILTER
SAR_TEM_CELL

12.9 Surface_list

Surfaces within the problem space are specified in the surface_list packet. A number of simple geometric surfaces such as rectangular blocks or spheres may be specified by appropriate parameters or alternatively a file with a surface triangulated mesh may be specified.
Each surface has a transformation associated with it which allows rotation and translation of the specified surface as described in section 3. The translation is described by six parameters in two sets of three on separate lines of the file.

θx, θy, θz
ox, oy, oz

Spherical surfaces and rectangular surfaces may have a surface roughness applied. This is achieved by moving points of the surface triangulation a random distance in the surface normal direction.
To include a surface roughness the following lines should be places between the surface parameters and the transformation parameters:

Surface_roughness
P1 P2

Where P1 and P2 are the parameters of the surface roughness. P1 is a required parameter and specifies the amplitude of the surface roughness applied. For a rectangular surfaces a second parameter, P2 may be specified. This parameter specifies the surface triangulation edge length and is related to the surface roughness correlation length. Examples will be given for the sphere and rectangle surface specifications at the end of this section.

The available surface types and their parameters are as follows:

Name:
    Rectangle
Description:
    Rectangle in the xy plane centred on the origin. Surface normal direction is +z.
Parameters:
    Length in x, length in y
    θx, θy, θz
    ox, oy, oz

Name:
    circle
Description:
    Circle in the xy plane centred on the origin. Surface normal direction is +z.
Parameters:

radius
θx, θy, θz
ox, oy, oz

Name:
Rectangular_block
Description:
Surface of a rectangular block centred on the origin. Surface normal direction is outwards.

Parameters:
Length in x, length in y, length in z
θx, θy, θz
ox, oy, oz

Name:
Rectangular_block2
Description:
Surface of a rectangular block specified by its xmin, ymin, zmin and xmax, ymax and zmax. Surface normal direction is outwards.
coordinates
Parameters:
Xmin ymin zmin xmax ymax zmax
θx, θy, θz
ox, oy, oz

Name:
cylinder
Description:
Surface of a circular cylinder centred on the origin. The cylinder is circular in the xy plane and extruded in the ±z direction to the total length specified.. Surface normal direction is outwards.

Parameters:
Radius, length (in z direction)
θx, θy, θz
ox, oy, oz

Name:
sphere
Description:
Spherical surface centred on the origin. Surface normal direction is outwards.

Parameters:
Radius
θx, θy, θz
ox, oy, oz

Name:
hemisphere
Description:
Hemispherical surface centred on the origin. Surface normal direction is outwards.

Parameters:
Radius
θx, θy, θz
ox, oy, oz

Name:
annulus
Description:
Annulus normal to z

Parameters:
Inner_radius (m), outer_radius (m)
θx, θy, θz
ox, oy, oz

Name:
Split_ring
Description:
Annulus with a gap as used in split ring resonators

Parameters:
Inner_radius (m), outer_radius (m), gap_angle (degrees)
θx, θy, θz
ox, oy, oz

Name:
Helix
Description:
Helical surface formed by wrapping a strip around a cylinder from a starting angle to a finishing angle. For multiple turns the finishing angle may be a larger than 360 degrees.

Parameters:
radius (m), height(m), angle1 (degrees), angle2 (degrees), pitch (m), direction (+1 or -1)
θx, θy, θz
ox, oy, oz

Name:
quad
Description:
Quadrilateral defined by 4 points in 3-D space.

Parameters:
x1 y1 z1 x2 y1 z2 x3 y3 z3 x4 y4 z4
θx, θy, θz
ox, oy, oz

Name:
xplane
Description:

Rectangular plane normal to x defined by 2 opposite diagonal points in 3-D space

Parameters:

x1 y1 z1 x2 y1 z2

θx, θy, θz

ox, oy, oz

Name:

yplane

Description:

Rectangular plane normal to y defined by 2 opposite diagonal points in 3-D space

Parameters:

x1 y1 z1 x2 y1 z2

θx, θy, θz

ox, oy, oz

Name:

zplane

Description:

Rectangular plane normal to z defined by 2 opposite diagonal points in 3-D space

Parameters:

x1 y1 z1 x2 y1 z2

θx, θy, θz

ox, oy, oz

Name:

triangle

Description:

Triangle defined by 3 points in 3-D space. Surface normal direction is defined by the ordering of the points 1-2-3.

Parameters:

x1 y1 z1 x2 y2 z2 x3 y3 z3

θx, θy, θz

ox, oy, oz

Name:

Triangulated_surface

Description:

Triangulated surface defined in a separate file

The surface can be scaled by a scale_factor once it is read in and a parameter (value +1 or -1) allows the surface normal to be reversed if required (reverse_surface_normal =-1)

The surface normal direction is defined by the ordering of the points 1-2-3 on each triangle.

Parameters:

Surface_filename

Scale_factor reverse_surface_normal

θx, θy, θz

ox, oy, oz

Name:

Vtk_triangulated_surface

Description:

Triangulated surface in vtk format defined in a separate file
The surface can be scaled by a scale_factor once it is read in and a parameter (value +1 or -1) allows the surface normal to be reversed if required (reverse_surface_normal =-1)
The surface normal direction is defined by the ordering of the points 1-2-3 on each triangle.

Parameters:

Surface_filename
Scale_factor reverse_surface_normal
θx, θy, θz
ox, oy, oz

Name:

Stl_triangulated_surface

Description:

Triangulated surface in stl format defined in a separate file
The surface can be scaled by a scale_factor once it is read in and a parameter (value +1 or -1) allows the surface normal to be reversed if required (reverse_surface_normal =-1)
The surface normal direction is defined by the ordering of the points 1-2-3 on each triangle.

Parameters:

Surface_filename
Scale_factor reverse_surface_normal
θx, θy, θz
ox, oy, oz

example

```
Surface_list
7   Number of surfaces (integer)
1      surface_number (integer)
sphere
1.0      surface parameters (n*real)
0.0 0.0 0.0
0.0 0.0 0.0
2      surface_number (integer)
cylinder
1.0  0.5     surface parameters (n*real)
0.0 0.0 0.0
0.0 0.0 0.0
3      surface_number (integer)
circle
1.0      surface parameters (n*real)
0.0 0.0 0.0
0.0 0.0 0.0
4      surface_number (integer)
rectangle
1.0     2.0  surface parameters (n*real)
0.0 0.0 0.0
0.0 0.0 0.0
```

5 surface_number (integer)
Rectangular_block
1.0 2.0 3.0 surface parameters (n*real)
0.0 0.0 0.0
0.0 0.0 0.0
6 surface_number (integer)
rectangle
0.25 0.25 surface parameters (n*real)
surface_roughness
0.02 0.025
0.0 0.0 0.0
0.0 0.0 0.0
7 surface_number (integer)
sphere
1.0 surface parameters (n*real)
surface_roughness
0.05
0.0 0.0 0.0
0.0 0.0 0.0

Example test cases:
SAR_TEM_CELL
SHIELDING_EFFECTIVENESS_BOX_SLOT

12.10 Line_list

Lines within the problem space are specified in the line_list packet. A number of simple geometric lines such as straight lines or arcs may be specified by appropriate parameters . Lines are used to specify cable routes and multiple lines may be used to generate a cable route.
Each line has a transformation associated with it which allows rotation and translation of the specified line as described in section 3. The translation is described by six parameters in two sets of three on separate lines of the file.

θx, θy, θz
ox, oy, oz

The available line types and their parameters are as follows:

Name:
Straight_line
Description:
Straight line in the x direction, centred at the origin.
Parameters:
Length (in the x direction)
θx, θy, θz
ox, oy, oz

Name:
Straight_line2
Description:
Straight line specified by its end points
Parameters:

Xmin ymin zmin xmax ymax zmax
θx, θy, θz
ox, oy, oz

Name:
circle
Description:
Circle in the xy plane centred at the origin
Parameters:
radius
θx, θy, θz
ox, oy, oz

Name:
arc
Description:
Arc of a circle in the xy plane centred at the origin. The arc extends from θ1 to θ2 where θ is measured in the anticlockwise direction from the x axis
Parameters:
Radius, θ1, θ2
θx, θy, θz
ox, oy, oz

Name:
rectangle
Description:
Rectangle in the xy plane centred at the origin. The rectangle is specified by its length in the x and y directions respectively
Parameters:
Length in x, length in y.
θx, θy, θz
ox, oy, oz

example

Line_list
4 Number of lines (integer)
1 line_number (integer)
straight_line
0.042 line parameters (n*real)
90.0 0.0 -90.0
0.375 0.185 0.2789
2 line_number (integer)
arc
0.056 1.0 175.0 line parameters (3*real: radius, theta1, theta2)
90.0 180.0 0.0
0.222 0.2505 0.240
3 line_number (integer)
circle
0.056 line parameters (real: radius)
0.0 0.0 0.0

0.5 0.1 0.2
4 line_number (integer)
Straight_line2
0.056 0.0 179.0 0.056 0.4 179.0 line parameters (6*real: xmin, ymin, zmin, xmax, ymax, zmax)
90.0 180.0 0.0
0.0 0.0 0.0

Example test cases:
YORK_BOX_STRAIGHT_WIRE
WIRE_OVER_LOSSY_GROUND

12.11 Point_list

A point is defined simply by its coordinates in 3D space, plus the usual geometric transformation.

example

```
Point_list
4   Number of points (integer)
1       POINT_NUMBER (integer)
0.0 0.0 -5.0    point coordinates (n*real)
0.0 0.0 0.0
0.0 0.0 0.0
2       POINT_NUMBER (integer)
0.0 0.0 0.0    point coordinates (n*real)
0.0 0.0 0.0
0.0 0.0 0.0
3       POINT_NUMBER (integer)
0.0 0.0 5.0    point coordinates (n*real)
0.0 0.0 0.0
0.0 0.0 0.0
4       POINT_NUMBER (integer)
0.0 1.0 1.0    point coordinates (n*real)
0.0 0.0 0.0
0.0 0.0 0.0
```

Example test cases:
YORK_BOX_STRAIGHT_WIRE
DIPOLE

12.12 Cable_geometry_list

The cable_geometry_list packet creates a numbered list of the cable geometries required in the problem. The number of cable geometries is specified followed by a list of the files specifying the cable geometries. The file specifying the cable geometry should have the extension **.cable** but this extension should be left off the filename.

example

Cable_geometry_list_packet_data

2 number of cable geometries
1 cable geometry number, Cable geometry type (file name) follows:
1.5mm_single_wire
2 cable geometry number, Cable geometry type (file name) follows:
RG58

Example test cases:
DIPOLE
COAX_MONOPOLE

12.13 Cable_list

The cable list creates cable routes from lists of line segments defined in the line_list packet. The cable end junctions are specified i.e. a cable must end on a specified junction.

example

cable_list
2 number of cables
1 CABLE_NUMBER,
1 cable geometry number
1 number of lines on cable route
1 cable line list
2 end 1 junction number
1   end 2 junction number
2 CABLE_NUMBER,
2 cable geometry number
5 number of lines on cable route
2 3 4 5 6 cable line list
3 end 1 junction number
4 end 2 junction number

Example test cases:
YORK_BOX_STRAIGHT_WIRE
MTL_CROSSTALK

12.14 Cable_junction_list

Cable junctions i.e. where cables connect together or terminate to cell faces are specified here. The junction specification starts with the point in space where the junction is situated is specified from the point list. The number of internal connection nodes is then specified, followed by the list of cables connecting to the junction and corresponding cable end numbers.

For each cable connecting to the junction, the topology of the individual cable connectivity is defined by a P matrix for each cable. This specifies how the conductors of the cable connect to the internal connection nodes. The number of rows in the P matrix for each cable is N_internal_connection_nodes and the number of columns is N_cable_conductors.
Following the connection P matrix, voltage source functions (one for each conductor, zero for no source) are specified followed by any resistances (again, one value for each conductor in the cable.)

Following the individual cable data we can specify any internal impedances to be connected between the internal connection nodes at the junction. This allows the creation of simple termination networks. Internal impedances are specified by the internal connection nodes between which they are connected followed by the impedance defined in the usual frequency dependent rational function form.

Finally there is an option to force a junction onto a TLM cell face (FACE) and furthermore to allow a V=0 boundary condition to be applied to one of the internal connection nodes. This models a connection to a surface in the mesh and allows current to flow from the cables and onto the surface. There is a boundary condition on each internal connection node, setting this to -1 connects the corresponding internal connection node to the surface. Note: for this to work there must be a conducting surface at the position of the termination face in the mesh. Also if boundary conditions are being applied then it is only strictly necessary to have a single internal connection node with boundary condition -1 as multiple internal connection nodes with boundary condition -1 will effectively be joined together anyway.

Note that a junction on a surface may have cables incident upon it from two opposite directions. The connectivity and boundary conditions may be set such that signals on cables can be transmitted through surfaces e.g. a coax feed through a ground plane.

example

```
2    number of junctions
1    JUNCTION NUMBER
2    junction point number
1   number of internal connection nodes (n_int)
2   number of cables
1 2   cable list
2 1   corresponding cable end number list
1    Cable 1 number of external conductors (n_ext)
1    Cable 1 P matrix: matrix (n_int rows* n_ext columns)
1    Cable 1 voltage source function list
25.0 Cable 1 impedance list
1    Cable 2 number of external conductors (n_ext)
1    Cable 2 P matrix: matrix (n_int rows* n_ext columns)
1    Cable 2 voltage source function list
25.0 Cable 2 impedance list
0    number of internal impedances
2    JUNCTION NUMBER
3    junction point number
2   number of internal connection nodes (n_int)
2   number of cables
1 3   cable list
2 1   corresponding cable end number list
1    Cable 1 number of external conductors (n_ext)
1    Cable 1 P matrix: matrix (n_int rows* n_ext columns)
0
1    Cable 1 voltage source function list
5.0  Cable 1 impedance list
1    Cable 2 number of external conductors (n_ext)
0    Cable 2 P matrix: matrix (n_int rows* n_ext columns)
1
1    Cable 2 voltage source function list
```

```
0.0   Cable 2 impedance list
1    number of internal impedances
1    Internal impedance number
1 2 ! between internal conection nodes
# Impedance filter function
1.0e0            # w normalisation constant
1          # a order
0.0987 6.59e-10  # a coefficients
0          # b order
1.0        # b coefficients
FACE
-1  boundary condition (-1 indicates V=0 i.e. surface connection)
```

Coax feeding a monopole through a ground plane. Cable 1 is the coaxial cable with two conductions, the screen being conductor 1, cable 2 is the single conductor monopole. The termination below specifies a coax to monopole feed through a ground plane where the screen is connected to the ground plane and the inner conductor feeds the monopole.

```
1    number of junctions
1   JUNCTION NUMBER
2    junction point
2   number of internal connection nodes (n_int)
2   number of cables
1 2   cable list
2 1   corresponding cable end number list
2    Cable 1 number of external conductors (n_ext)
 1 0       Cable 1 P matrix: matrix (n_int rows* n_ext columns)
-1 1
0    0     Cable 1 voltage source function list
0.0 0.0    Cable 1 impedance list
1    Cable 2 number of external conductors (n_ext)
1    Cable 2 P matrix: matrix (n_int rows* n_ext columns)
0
0    Cable 2 voltage source function list
0.0   Cable 2 impedance list
0    Number of internal impedances
FACE
0 -1  boundary condition (-1 indicates V=0 i.e. surface connection)
```

Example test cases:

CABLE_TEST
YORK_BOX_STRAIGHT_WIRE
COAX_MONOPOLE
AIRCRAFT_DIRECT_CURRENT_INJECTION

12.15 Cable_output_list

Cable outputs are defined in the cable_list packet using the cable number and also specifying a point number from the point_list packet. The cable current at the cable node closest to the specified point will be written to a file with extension **.cable_current.tout**.

example

```
cable_output_list
3  number of cable_outputs
1  cable output number
1 closest point number
1 cable end number
2 cable output number
2 cable number
5 closest point number
3 cable output number
2 cable number
4 closest point number
```

Example test cases:
YORK_BOX_STRAIGHT_WIRE
COAX_MONOPOLE

12.16 Volume_material_list

The volume_material_list associates numbered volumes defined in the volume_list packet with material properties. The volume material properties are either defined as PEC, PMC or are DISPERSIVE materials with properties defined in a file which specifies the frequency dependent relative permittivity and permeability parameters of the volume. The dispersive material file name specified does not include the **.vmat** extension used for volume material properties. More than one geometric volume may be given the same property in each entry of the list therefore it is only required to have a single entry for each material type in the problem

Example

```
volume_material_list
4   Number of volume materials (integer)
1   volume MATERIAL NUMBER
Dispersive
MATERIAL_DIRECTORY/epsr=1.5
2       number of volumes
1 3       volume list
2   volume MATERIAL NUMBER
Dispersive
MATERIAL_DIRECTORY/debye_1
2       number of volumes
2 4       volume list
3   volume MATERIAL NUMBER
PEC
2       number of volumes
5 6       volume list
4   volume MATERIAL NUMBER
PMC
1       number of volumes
7       volume list
```

Example test cases:
RCS_DIELECTRIC_IBC_SPHERE
DIELECTRIC_FILTER
SAR_TEM_CELL

12.17 Surface_material_list

The surface_material_list associates numbered surfaces defined in the surface_list packet with material properties. The surface material properties are either PEC, PMC, FREE_SPACE, DISPERSIVE, ANISOTROPIC DISPERSIVE or DIODE types.

DISPERSIVE and ANIOTROPIC_DISPERSIVE types have properties which are defined in a file specifying the frequency dependent impedance (Z) parameters of the surface. The file name specified does not include the **.smat** extension used for impedance boundary properties.

The DIODE type is specified by the diode model parameters, Is, nVt, Rs and Cj plus the diode forward (conducting) direction.

More than one surface may be given the same property in each entry of the list therefore it is only required to have a single entry for each surface material type in the problem.
Thin material sheet properties may not be symmetrical so the surface orientation must be specified for each surface. A surface orientation value of 1 puts side 1 of the material on the outward normal whereas a value of -1 sets side 1 of the material on the opposite side to the outward normal.

example

```
Surface_material_list
2   Number of surface materials (integer)
1   SURFACE MATERIAL NUMBER
PEC
1       number of surfaces
3       surface list
1       surface orientation list
2   SURFACE MATERIAL NUMBER
Dispersive
MATERIAL_DIRECTORY/Dispersive_sheet
3       number of surfaces
1 2 4       surface list
1 1 1      surface orientation list
2   SURFACE MATERIAL NUMBER
Anisotropic_Dispersive
MATERIAL_DIRECTORY/Aniostropic_dispersive_sheet
1       number of surfaces
5       surface list
1      surface orientation list
2   SURFACE MATERIAL NUMBER
DIODE
1e-12 0.026 6.0 0.2e-12 ! parameters Is nVt Rs and Cj
-z      diode forward (conducting) direction
1       number of surfaces
6       surface list
1       surface orientation list
```

Example test cases:
RCS_COATED_IBC_SPHERE

YORK_BOX_STRAIGHT_WIRE
SHIELDING_EFFECTIVENESS_BOX_SLOT
TEMX_ANISOTROPIC_THIN_LAYER_Y_POL_TEST
DIODE

12.18 Excitation_function_list

The excitation functions to be used in the simulation are defined as a numbered list in this packet. An excitation function may be chosen from a set of analytic functions or be defined in a file. The analytic excitation functions available and their parameters are as follows:

Impulse: Amplitude, delay (seconds)

$$f(t) = Amplitude \times \delta(t - delay)$$

Step: Amplitude, delay (seconds)

$$f(t) = Amplitude \times H(t - delay)$$

Gaussian: Amplitude, width (seconds) delay (seconds)

$$f(t) = Amplitude \times e^{-(t-delay)^2/width^2}$$

Gaussian_step: Amplitude, width (seconds) delay (seconds)

$$f(t) = \begin{cases} Amplitude \times e^{-(t-delay)^2/width^2} & t < delay \\ Amplitude & t \geq delay \end{cases}$$

Differential_gaussian: Amplitude, width (seconds) delay (seconds)

$$t_{peak} = -\frac{width}{\sqrt{2}}$$

$$f_{peak} = \left(-2\frac{t_{peak}}{width^2}\right) e^{-(t_{peak})^2/width^2}$$

$$f(t) = \frac{Amplitude}{f_{peak}} \times \left(-2\frac{(t - delay)}{width^2}\right) e^{-(t-delay)^2/width^2}$$

Sinusoid: Amplitude, frequency, f (Hz) phase (degrees)

$$f(t) = Amplitude \times \sin(2\pi ft - \pi \times \frac{phase}{180})$$

Gaussian_sinusoid: Amplitude, width (seconds) delay (seconds) frequency, f (Hz) phase (degrees)

$$f(t) = Amplitude \times e^{-(t-delay)^2/width^2} \quad \sin(2\pi f(t - delay) - \pi \times \frac{phase}{180})$$

Gaussian_step_sinusoid: Amplitude, width (seconds) delay (seconds) frequency, f (Hz) phase (degrees)

$$f(t) = \begin{cases} Amplitude \times e^{-(t-delay)^2/width^2} \sin(2\pi f(t-delay) - \pi \times \frac{phase}{180}) & t < delay \\ Amplitude \times \sin(2\pi f(t-delay) - \pi \times \frac{phase}{180}) & t \geq delay \end{cases}$$

Double_exponential: Amplitude, rise time constant, τ_1,(seconds) fall time constant, τ_2,(seconds) delay (seconds) frequency, f (Hz) phase (degrees)

$$t_{peak} = \frac{\ln\left(1/\tau_2\right) - \ln\left(1/\tau_1\right)}{\left(\frac{1}{\tau_2} - \frac{1}{\tau_1}\right)}$$

$$f_{peak} = \left(e^{-t_{peak}/\tau_1} - e^{-t_{peak}/\tau_2}\right)$$

$$f(t) = Amplitude \times \left(e^{-t/\tau_1} - e^{-t/\tau_2}\right) / f_{peak}$$

Sinusoidal_pulse: amplitude, frequency (Hz), phase (degrees) , t1 (seconds), t2(seconds), t3(seconds), t4(seconds).

This is a sinusoid multiplied by a trapezoidal pulse:

0≤ time < t1: A=0
t1 ≤ time < t2: A=amplitude*(time-t1)/(t2-t1)
t2 ≤ time < t3: A=amplitude
t3 ≤ time < t4: A=amplitude*(t4-time)/(t4-t3)
time ≥ t4: A=0

$$f(t) = A * \sin(2\pi f t - \pi \times \frac{phase}{180})$$

Noise: Amplitude

This function sets the source time series as a sequence of random real numbers between $-\frac{Amplitude}{2}$ and $\frac{Amplitude}{2}$. The random numbers are chosen to have a uniform distribution in this range.

The final excitation specification allows the user to specify a filename for time domain sample data. The additional parameters are an amplitude and delay. The file must be in the format of two columns, the first containing time values and the second containing the corresponding function values i.e.

Function_filename.dat

t0 f0
t1 f1
t2 f2
t3 f3
.

.
.
tn fn

for times t<t0 the function is assumed to take the value f0 and for times t>tn the function takes the value fn.

All the excitation functions are written to the **.excitation.tout** file when the simulation is run.

example

```
Excitation_function_list
3                    ! Number of excitation functions
1                    ! EXCITATION NUMBER
gaussian
1.0 2.5e-9 10e-9
2                    ! EXCITATION NUMBER
Step
1.0 0.0
3                    ! EXCITATION NUMBER
Function_filename.dat
12.0 1e-9
```

Example test cases:
DIPOLE
COAX_PROPAGATION
WAVEGUIDE_IRIS

12.19 Excitation_point_list

This packet excites the given field component on a point specified from the point_list. The excitation function is set from the numbered list defined in the Excitation_function packet. A soft source must be chosen (hard sources will come in the future if required) and it may be applied either at the centre of a cell or on a specified face (xmin xmax ymin ymax zmin or zmax.) More than one source may be specified at a particular point and the source applied will be the sum of the specified sources.

example

```
Excitation_point_list
1            ! number of output points
1            ! EXCITATION POINT NUMBER
1            ! excitation function number
2            ! excitation point
Ex
centre
soft
```

Example test cases:
CAVITY_FREE_SPACE

MATERIAL_TEST

12.20 Excitation_surface_list

This packet excites the given field component on a surface specified from the surface_list. The excitation function is set from the numbered list defined in the Excitation_function packet. A soft or a hard source may be chosen. More than one source may be specified at a particular surface and the source applied will be the sum of the specified sources.
The side of the surface for the excitation can be specified. If the side of the surface is -1 then a field is added onto the –ve side of the surface, if the side of the surface is +1 then it is added onto the +ve side of the surface and if it is zero, the source is added to both sides of the surface. A value of 0 is the usual value for this parameter.
(Note: If a single sided excitation is used then the requested field (E or H) is excited by adding the appropriate voltage (current) on the specified side of the surface, this will also excite an associated current (voltage). When the double sided excitation is used then the associated current (voltage) is zero as the contributions to the associated current(voltage) from the two sides cancel out.)

Example

```
Excitation_surface_list
1            ! number of excitation surfaces
1            ! EXCITATION SURFACE NUMBER
1            ! excitation function number
2            ! excitation surface
0            ! side of surface for excitation
Ex
Soft
```

Example test cases:
WAVEGUIDE_RESONATOR

12.21 Excitation_mode_list

This output packet allows a mode shape to be read from a file and used to excite the problem over a defined plane. The plane must (at the moment) be defined in the same orientation as the mode. More than one source may be specified at a particular surface and the source applied will be the sum of the specified sources.

example

```
Excitation_mode_list
1            ! number of excitation modes
1            ! EXCITATION MODE NUMBER
1            ! excitation function number
5            ! surface number
1          ! side of surface for excitation
Ex
soft
WR90.mode
```

```
1            ! x column
2            ! y column
3            ! z column
6            ! data column to use
```

Example test cases:
WAVEGUIDE_IRIS
WAVEGUIDE_STRAIGHT

12.22 Huygens_surface

The Huygens surface allows a wave to be launched from a (normally closed) surface inside the problems space. The computational volume is then divided into total field (within the Huygens surface) and scattered field (outside the Huygens surface) regions. This then allows the scattered field to be calculated.
The Huygens surface is specified to be a previously defined surface from the surface_list, an exception is if the outer boundary is to be used as the Huygens surface, in this case set the Huygens surface number to zero. An excitation function is specified from the excitation function list. The direction of the wave vector and the polarisation of the Electric field component of thin incident wave are defined in spherical polar coordinates.
The direction of the plane wave and its polarisation can both be set to random values by specifying the wave vector and/or the polarisation to be 'random'

example

Incident wave travelling in the –z direction with Ex polarisation:

Huygens_surface
8 surface number
1 side of surface for excitation
1 excitation function number
180.0 0.0 wave vector Theta and Phi
1.0 0.0 Polarisation theta and Phi

Incident wave with random direction and polarisation

Huygens_surface
8 surface number
1 side of surface for excitation
1 excitation function number
Random wave vector Theta and Phi
Random Polarisation theta and Phi

Example test cases:

HUYGENS_SURFACE
RCS_PEC_SPHERE
SHIELDING_EFFECTIVENESS_BOX_SLOT

12.23 Far_field_surface

This packet defines a surface as a far field surface. Fields are collected on this surface as the simulation progresses and a near to far field transformation is used to calculate the far field pattern over the specified ranges of theta and phi at the given frequency.
The output data is written to the **.far_field.fout** file.

example

```
Far_field_surface
1     number of far field surfaces
1     FAR_FIELD_SURFACE_NUMBER
1     surface number
1          ! side of surface for output (+1 or -1)
1.3E7 frequency for far field calculation
0.0 180.0 1.0  Theta_min  Theta_max Theta_step
0.0 90.0 90.0    Phi_min  Phi_max Phi_step
```

Example test cases:
DIPOLE

12.24 RCS_surface

This packet defines a surface on which fields are recorded and, using a time domain near to far field transformation, the time domain far field is calculated in a specified direction. Note the direction defined is the wave vector for the radiated wave this for a monostatic RCS calculation the direction of the incident wave defined in the Huygens_surface packet and the scattered wave defined in the monostatic RCS surface are opposite.

The time domain normalised far field response is written to the **.rcs.tout** file and the frequency domain RCS is written to the **.rcs.fout** file

example

```
RCS_surface
2     surface number
1     Side of surface for RCS output
0.1e8 3e8 0.01e8 frequency range for far field calculation
180.0  Theta
0.0  Phi
```

Example test cases:
RCS_MATERIAL_SPHERE
RCS_PEC_SPHERE
RCS_SMAT_NON_SYMMETRIC_SPHERE
RCS_SMAT_SPHERE
RCS_COATED_IBC_SPHERE
RCS_COATED_SPHERE
RCS_DIELECTRIC_IBC_SPHERE

12.25 Output_point_list

This output packet writes the requested field component at the specified point as a function of time. The point may be output at either the centre of a cell or on a specified face (xmin xmax ymin ymax zmin or zmax.)
Output_timestep information may be specified, this allows the first and last timestep plus timestep interval to be specified or alternatively the first time, last time and time interval. If this information is not specified then output will be written for all timesteps

The output data is written to a **.field.tout** file.

example

```
Output_point_list
1          ! number of output points
1          ! OUTPUT NUMBER
4          ! output point
Ey
centre
output_timestep_information
1      ! first output timestep
2000    ! last output timestep
1     ! output timestep interval
```

Example test cases:
CAVITY_FREE_SPACE
DIPOLE
HUYGENS_SURFACE

12.26 Frequency_domain_power_surface_list

This output packet specifies a surface and returns the total power transmitted through the surface as a function of frequency. The power transmitted is defined to be in the outward normal direction of the specified surface. The output data is written to the **.frequency_domain_power_surface.fout** file.

The frequencies for the output data are specified as fmin, fmax and number of frequencies.

Example

```
Frequency_domain_power_surface_list
1          ! number of frequency domain power surfaces
1          ! FREQUENCY DOMAIN POWER SURFACE NUMBER
3          ! geometric surface number
1          ! side of surface for output
100e6 10e9 1000 ! fmin fmax n_frequencies for power output
```

Example test cases:
POWER_CALCS
TCS_MODE_STIR
TCS_MODE_STIR_CIRCULAR_APERTURE

12.27 Frequency_output_surface_list

This packet allows the output of a field component at a single frequency on a specified surface to a **.frequency_output_surface.fout** file.

This type of output may be post processed to produce field animations.

It is important to note that the timestep factor is neglected in the numerical evaluation of the Fourier integral in order to prevent the very small resulting values from causing problems for visualisation in paraview. If direct comparison is to be made with the frequency domain output from post processing using the discrete fourier transform then the output of this process must be multiplied by the solution timestep.

example

```
Frequency_output_surface_list
1          ! number of Frequency output surfaces
1          ! FREQUENCY OUTPUT SURFACE NUMBER
1          ! surface number
1          ! side of surface for output (+1 or -1)
1.3E7 frequency for output
Ex
```

Example test cases:
DIELECTRIC_FILTER
WAVEGUIDE_IRIS

12.28 Frequency_output_volume_list

This packet allows the output of a field component at a single frequency on a specified surface to a **.frequency_output_volume.fout** file.

This type of output may be post processed to produce field animations.

It is important to note that the timestep factor is neglected in the numerical evaluation of the Fourier integral in order to prevent the very small resulting values from causing problems for visualisation in paraview. If direct comparison is to be made with the frequency domain output from post processing using the discrete fourier transform then the output of this process must be multiplied by the solution timestep.

example

```
Frequency_output_volume_list
1          ! number of Frequency output volumes
1          ! FREQUENCY OUTPUT VOLUME NUMBER
1          ! volume number
1.3E7 frequency for output
Ex
```

Example test cases:
CYLINDRICAL_RESONATOR

12.29 New_mesh_generation

This packet applies the new mesh generation algorithm as described in section 6. The default is to use the 'original' mesh generation algorithm.

Example test cases:
GTEM_CELL

12.30 Output_surface_list

This packet allows the output of a field component on a specified surface **.surface_field.tout** file. Output_timestep information may be specified, this allows the first and last timestep plus timestep interval to be specified or alternatively the first time, last time and time interval. If this information is not specified then output will be written for all timesteps.

This type of output may be post processed to produce field animations.

example

```
Output_surface_list
1          ! number of output surfaces
1          ! OUTPUT SURFACE NUMBER
2          ! geometric surface number
-1         ! side of surface for output (+1 or -1)
Ez
output_timestep_information
0     ! first output timestep
600   ! last output timestep
60    ! output timestep interval
```

Example test cases:
AIRCRAFT_SURFACE_CURRENT
HUYGENS_SURFACE

12.31 Output_volume_list

This packet allows the output of a field component on a specified volume **.volume_ field.tout** file. Output_timestep information may be specified, this allows the first and last timestep plus timestep interval to be specified or alternatively the first time, last time and time interval. If this information is not specified then output will be written for all timesteps.

This type of output may be post processed to produce field animations.

example

```
Output_volume_list
2          ! number of output volumes
1          ! OUTPUT NUMBER
1          ! volume number for output
Ex
output_time_information
0.0   ! first output time
1e-6  ! last output time
1e-7  ! output time interval
```

```
2          ! OUTPUT NUMBER
3          ! volume number for output
Hz
output_timestep_information
0     ! first output timestep
200   ! last output timestep
10    ! output timestep interval
```

Example test cases:
DIELECTRIC_FILTER

12.32 Output_volume_average_list

This packet allows the output of the average of a field component over a specified volume **.volume_average_field.tout** file.

Output_timestep information may be specified, this allows the first and last timestep plus timestep interval to be specified or alternatively the first time, last time and time interval. If this information is not specified then output will be written for all timesteps.

This type of output may be post processed to produce field animations.

Example

```
Output_volume_average_list
1          ! number of output volume_averages
1          ! OUTPUT NUMBER
1          ! volume number for output
Ex
output_time_information
0.0   ! first output time
1e-6  ! last output time
1e-7  ! output time interval
```

Example test cases:
RESONANT_ABSORBER
TCS_MODE_STIR

12.33 Output_mode_list

This output packet allows a mode shape to be read from a file, the amplitude of this mode on a surface in the mesh is then used as output. The amplitude is calculated as the overlap integral of the mode shape with the field over the surface. The surface must (at the moment) be defined in the same orientation as the mode.

example

```
Output_mode_list
1          ! number of output modes
1         ! OUTPUT MODE NUMBER
6          ! surface number
1          ! side of surface for output
```

```
Ex
WR90.mode
1                ! x column
2                ! y column
3                ! z column
6                ! data column to use
```

Example test cases:
WAVEGUIDE_IRIS
WAVEGUIDE_STRAIGHT

12.34 SAR_volume_list

Output the Specific Absorption Rate calculated over a specified volume at at given frequency.
SAR is conventionally calculated as:

$$SAR = \frac{P_{absorbed}}{mass}$$

In the TLM process, the SAR calculated is not strictly the SAR as defined above, the output is the Energy absorbed divided by the mass (assuming a pulse excitation and a runtime sufficient for all the energy in the problem to be absorbed or radiated from the problem space). The TLM SAR may then be normalised to the excitation in such a way as to produce the desired result.

The Energy absorbed in a dielectric material with conductivity, σ is

$$E_{absorbed} = \int_V \frac{\sigma}{2}\left(\left|\tilde{E}_x^2\right| + \left|\tilde{E}_y^2\right| + \left|\tilde{E}_z^2\right|\right) dx\, dy\, dz$$

Where ~ denotes the field amplitude at the frequency of interest.

The SAR as output by GGI_TLM is then

$$SAR_{TLM} = \frac{E_{absorbed}}{mass} = \frac{\sum_{Material\ cells} \frac{\sigma}{2}\left(\left|\tilde{E}_x^2\right| + \left|\tilde{E}_y^2\right| + \left|\tilde{E}_z^2\right|\right)(\Delta l)^3}{\sum_{Material\ cells} \rho(\Delta l)^3}$$

Where ρ is the material density (kg/m^3) and the units of SAR_{TLM} are J/Kg

The volume material must be specified as a simple zero order dispersive material with loss included as an electric conductivity only. The material density must be given in `kg/m^3`.

SAR is calculated by a running Fourier Transform as the solution progresses. The output data is written to the **.SAR.fout** file.

example

```
SAR_volume_list
2 ! number of SAR volumes
1  ! SAR VOLUME NUMBER
1  ! material number
900E6  ! frequency
968.6  ! material density kg/m^3
2  ! SAR VOLUME NUMBER
2  ! material number
900E6  ! frequency
968.6  ! material density kg/m^3
```

Example test cases:
SAR_TEM_CELL
SAR_TEST

12.35 Simulation_time

This packet sets the time period of the simulation in seconds.
example

Simulation_time
7e-8

Example test cases:
All test cases

12.36 Mode_stir_surface_list

The mode_stir_surface_list specifies a set of TLM surfaces on which the random connection boundary condition is to be imposed. The surfaces specified may be internal surfaces or set to zero to indicate the outer boundary. All internal surfaces to be included in the list must have PEC properties set. A voltage scaling factor is specified and this multiplies the voltage pulses at every interaction with the boundary.
This boundary condition is being used experimentally to investigate efficient means of modelling mode stirred environment.

Example

```
Mode_stir_surface_list
2   Number of mode_stir_surfaces (integer)
1    MODE_STIR_SURFACE NUMBER
0.999      ! boundary condition voltage reflection coefficient
0.0  ! amplitude of random impulse excitation on the boundary
1    ! number of geometric surfaces in this mode_stir_surface
1    ! surface list
1    ! surface orientation list
2    MODE_STIR_SURFACE NUMBER
0.999      ! boundary condition voltage reflection coefficient
1.0  ! amplitude of random impulse excitation on the boundary
1    ! number of geometric surfaces in this mode_stir_surface
0    ! surface list - surface 0 is the outer boundary
1    ! surface orientation list
```

Example test cases:
TCS_MODE_STIR
TCS_MODE_STIR_CIRCULAR_APERTURE
NESTED_MODE_STIRRED_CHAMBERS
MODE_STIRRED_CHAMBER

12.37 Random_number_seed

The random number generator is used in GGI_TLM to randomise the boundary connection process in the mode stir boundary condition and also in the generation of random values for the noise source calculation.

This packet allows the seed values used by the random number generator to be specified by the user. The seed values may be set to specific values so that the code performance will be repeatable or set to random values based on date, time and process id. In this second case the results will be different on each occasion on which the code is run.

The gnu fortran random number generator requires 12 integer seed values. GGI_TLM checks whether the correct number of seed values has been specified .

Example1: Set specified seed values

```
Random_number_seed
12    ! number of values
12345
75653
724978
934157
928345
734586
1345
572
9786564
8934572
38564
23
```

Example2: set to random values

```
Random_number_seed
0    ! number of values =0 indicating a randomisation of seed values
```

Example test cases:

12.38 Frequency_scale_flag

This flag specifies the use of linear frequency scaling in all frequency dependent parameters defined using the rational function format and implemented as digital filters. This is being used experimentally to investigate the use of TLM in a similar manner to a power balance method for

over moded problems. The frequency scaling may be reversed in the output results by using the appropriate Fourier Transform in the post_processor.

example

Frequency_scale_flag
TRUE
2.0

Example test cases:
YORK_BOX_STRAIGHT_WIRE_FSCALE

12.39 Bicubic_warp_flag

This flag specifies the use of the bicubic transformation in place of the normal bilinear transformation in all frequency dependent parameters defined using the rational function format and implemented as digital filters. This is being used experimentally to investigate the use of TLM in a similar manner to a power balance method for over moded problems. The frequency warping may be reversed in the output results by using the appropriate Fourier Transform in the post_processor (Fourier Transform with frequency warping).
This flag should not be used with lossy volume materials as the TLM solution becomes unstable.

example

Frequency_warp_flag
TRUE

Example test cases:

12.40 Wrapping_boundary_conditions

This packet allows wrapping boundary conditions to be specified on the ±x, ±y and ±z boundaries. The wrapping boundary conditions transfers outward going voltage pulses at the + boundary to be transferred to become inward going voltage pulses at the – boundary and vice-versa. The effect of this is that a wave leaving the problem space through the + boundary comes back into the problem space from the – boundary. I can be used in the modelling of periodic structures.

The packet data takes the form of three integers set to either 0 (no wrapping boundary) or 1 (set wrapping boundary) as follows

x_wrap(1 or 0) y_wrap(1 or 0) z_wrap(1 or 0)

Note that the Outer_boundary_reflection_coefficient should be set to 1 on boundaries specified as 'wrapping' boundaries.

example

In order to set wrapping boundaries on x and y boundaries but not z:

Wrapping_boundary_conditions
1 1 0

Example test cases:
WRAPPING_BOUNDARY
RESONANT ABSORBER

12.41 Periodic_boundary

Note that this is an experimental process and should be used with caution at the moment...

This packet allows periodic boundary conditions to be specified on the ±x, ±y boundaries for the modelling of infinite arrays with an arbitrary angle incident wave (or equivalently, time delay between cell responses).

The implementation uses the method of Lee and Smith, "An alternative approach for implementing periodic boundary conditions in the FDTD method using multiple unit cells," IEEE trans AP, vol 54, no 2, 2006, pp 698-705.

We note that the solution may be late-time unstable when dielectric or magnetic materials are present in the unit cell. This is due to the TLM dispersion relation in the presence of cell centre stubs. Note also that the unit cell cannot include any thin wires at the moment.

Note that the Outer_boundary_reflection_coefficient should be set to 1 on the ±x, ±y boundaries. It may be required to set an appropriate reflection coefficient on the ±z boundaries to match the incident wave as the specified direction may not be normal to the boundary.

example

The periodic boundary is turned on with the following line:
periodic_boundary

The properties of the boundary condition are determined by the angle of incidence of the plane wave specified in the Huygens surface packet:
Huygens_surface
5 surface number
1 side of surface for excitation
1 excitation function number
20.0 30.0 wave vector theta and phi values
1.0 0.0 Polarisation theta and phi values

Example test cases:

12.42 Periodic_boundary_far_field_surface

Note that this is an experimental process and should be used with caution at the moment...

This packet allows the calculation of the far field in periodic structures. The periodic_boundary flag must be set to use this packet.
The frequency domain transmitted or reflected wave in the specular direction (determined by the Huygens surface wave direction) is calculated.

example

```
periodic_boundary_far_field_surface
2          number of periodic far field surfaces
1          PERIODIC FAR FIELD SURFACE NUMBER
2          surface number
1          side of surface for output
0.1e9  18e9 0.1e9          fmin  fmax and fstep for frequency domain output
Transmission
0    0    order of grating lobe in x and y  ! not used yet
1    PERIODIC FAR FIELD SURFACE NUMBER
4    surface number
2    side of surface for output
0.1e9  18e9 0.1e9          fmin  fmax and fstep for frequency domain output
Reflection
0 0        order of grating lobe in x and y  ! not used yet
```

Example test cases:

12.43 LC_correction_type_subtract_cell_inductance

This line in the input file indicates that the code should correct the TLM in-cell cable model inductance by subtracting the inductance of the TLM cell which is seen by the cable common mode current. This is the correction published by John Paul:

J.Paul, C.Christopoulos and D.W.P.Thomas, "Correction to ``Time-Domain Modeling of Electromagnetic Wave Interaction with Thin-Wires using TLM," IEEE Transactions on Electromagnetic Compatibility, vol 50, no 2, pp 450-451, 2008.

12.44 LC_correction_type_geometry_scale

This line in the input file indicates that the code should correct the TLM in-cell cable model by scaling the TLM reference return conductor radius for the inductance and capacitance calculations independently. The geometric scaling factors are

The default value is 1/1.08 for the Inductance_ equivalent_radius_factor

The default value is 1.08 for the Capacitance_ equivalent_radius_factor

See the **Inductance_ equivalent_radius_factor** and **Capacitance_ equivalent_radius_factor** parameters which may be specified by the user as described below.

12.45 No_geometry_vtk_files

This line in the input file suppresses the writing of geometry vtk files during the model building process. This may be useful for a familiar geometry for instance in an optimisation loop.

12.46 Compress_output_files

It is recognised that some output files can become extremely large. As an attempt to address this the output files can be compressed using gzip. At the moment this is implemented for frequency_output_volume data only but could be extended to all output types.

In the software the compression is achieved by a system command of the form

gzip -9 -c < file > file.gz &

Example

Compress_output_files

12.47 Reduced_c_factor

Reduced c is a technique for situations in which magnetic field diffusion dominates the problem and very long runtimes are required, for example in some lightning strike simulations. The reduced c algorithm increases the timestep for the simulation by artificially increasing the permittivity and permeability of free space and thus reducing the velocity of light and hence increasing the timestep. In order to maintain the L/R time constant dominating the magnetic diffusion terms, the conductivity is decreased by the same factor. This packet has a single parameter which is the timestep increase factor.

Example

reduced_c_factor
100

12.48 GGI_TLM Parameters:

Number_of_fourier_terms_in_pul_lc_calc
The per-unit-length inductance and capacitance calculation uses a Fourier series expansion of charge density around the conductors, This parameter sets the number of terms to be used in the expansion. The default value is 10. This parameter may be changed (reduced) if there is an overflow in the per-unit-length inductance and capacitance matrix calculation for very small diameter cables (compared to the cell dimension).

TLM_equivalent_radius_factor
The per-unit-length inductance and capacitance calculation assumes that the TLM reference return conductor is cylindrical (as opposed to the square cross section). The equivalent radius is set to be (Δl/2)* TLM_equivalent_radius_factor. The default value is 1.08. Note that the radius may be further modified for the inductance and capacitance calculations as described below.

Capacitance_ equivalent_radius_factor
It has been found by experience that the TLM cable model operates most accurately when the TLM equivalent radius used in the per-unit-length inductance and capacitance calculations are not the same. The Capacitance_ equivalent_radius_factor multiplies the TLM_equivalent_radius in the capacitance calculation. The default value is 1.08.

Inductance_ equivalent_radius_factor
It has been found by experience that the TLM cable model operates most accurately when the TLM equivalent radius used in the per-unit-length inductance and capacitance calculations are not the same. The Inductance_ equivalent_radius_factor multiplies the TLM_equivalent_radius in the inductance calculation. The default value is 1/1.08.

Max_cable_bundle_diameter_factor
In some models, a cable bundle diameter may approach or even exceed the TLM dell dimension. If this occurs then the TLM_equivalent_radius must be increased so as to enclose the full cable bundle. The maximum cable bundle diameter is set as a proportion of the TLM_equivalent_radius. The default value is 0.67. If the TLM_equivalent_radius is too small then it is reset such that the bundle radius= TLM_equivalent_radius× Max_cable_bundle_diameter_factor.

13. Files used by GGI_TLM

```
input_file_extension='.inp'

warning_file_extension='.warning'

tet_volume_file_extension='.v_geom.vtk'

volume_mesh_file_extension='.v_mesh.vtk'

triangulated_surface_file_extension='.s_geom.vtk'

surface_mesh_file_extension='.s_mesh.vtk'

line_segment_file_extension='.l_geom.vtk'

line_mesh_file_extension='.l_mesh.vtk'

point_file_extension='.p_geom.vtk'

point_mesh_file_extension='.p_mesh.vtk'

boundary_mesh_file_extension='.b_mesh.vtk'

volume_material_cells_file_extension='.vmat_cells.vtk'

surface_material_faces_file_extension='.smat_faces.vtk'

cable_mesh_file_extension='.c_mesh.vtk'

mesh_file_extension='.mesh'
```

```
surface_material_file_extn='.smat'

volume_material_file_extn='.vmat'

cable_geometry_file_extn='.cable'

info_file_extn='.info'

cable_info_file_extn='.cable_info'

cable_model_file_extn='.cable_model'

field_output_extn='.field.tout'

excitation_output_extn='.excitation.tout'

cable_current_output_extn='.cable_current.tout'

volume_field_output_extn='.volume_field.tout'

volume_average_field_output_extn='.volume_average_field.tout'

surface_field_output_extn='.surface_field.tout'

far_field_output_extn='.far_field.fout'

frequency_output_surface_extn='.frequency_output_surface.fout'

trcs_output_extn='.rcs.tout'

rcs_output_extn='.rcs.fout'

SAR_output_extn='.SAR.fout'

mode_output_extn='.mode.tout'

frequency_domain_power_surface_extn='.frequency_domain_power_surface
.fout'

progress_filename='progress'
```

14. Post_processing

Once the solution has finished, the results may be processed for analysis and viewing using the post processor:

GGI_TLM_post_process

The options available are as follows:

```
1.  Extract Time Domain Data
```

2. Plot Time Domain Data
3. Discrete Time Fourier Transform (Time Domain to Frequency Domain)
4. Plot Frequency Domain Data
5. Create Animation of Time Domain Surface/Volume Field/ Current Output
6. Combine Frequency Domain Data: S(f)=A f1(f)/(B f2(f))+C
7. Combine Frequency Domain Magnitude Data: S(f)=A f1(f)/(B f2(f))+C
8. Frequency average
9. Oneport analysis
10. Twoport analysis for symmetric structures
11. IELF analysis
12. Create Animation of Frequency Domain Surface Field Output
13. Extract mode from Frequency Domain Surface Field Output
14. Sum Frequency Domain Data
15. Transmission cross section calculation
16. Discrete Time Fourier Transform with frequency warping (Time Domain to Frequency Domain)
17. Create vector field Animation of Frequency Domain Surface Field Output
18. Create vector field Animation of Frequency Domain Volume Field Output
19. S parameter to Z parameter transformation
20. Calculate correlation matrix from time domain data
21. Calculate correlation matrix from frequency domain data
22. Calculate complex antenna factor from (S21) measurement for two identical antennas
23. Calculate complex antenna factor from antenna coupling (S21) measurement with one unknown antenna
24. Fast Fourier Transform (Time Domain to Frequency Domain)
25. Generate random sample sequence either with or without an imposed correlation matrix
26. Apply a filter function to time domain data
27. Apply a filter function to frequency domain data
28. Calculate the frequency domain transfer function of a filter function
29. Combine Frequency Domain Data: S(f)=f1(f) f2(f)
30. Calculate PDF and CDF from a data set
31. Calculate correlation function from time domain data
32. Calculate correlation function from frequency domain data
33. Create Vector Animation of Time Domain Volume Field Output
34. Create time domain near field scan
35. Set random_number_seed
36. Generate Far field plot data
37. Re-format data file
38. Visualise multi-dimensional data sets (up to 4D)
39. Calculate the impulse response of a filter function
40. S11 to VSWR
41. Convert Frequency Domain Volume Field Output to x y z Re{field} Im{field} |field| format
42. Calculate spatial covariance for 1D and 2D complex data sets
43. Calculate and propagate Wigner function from complex spatial correlation data
44. Sum Time Domain Data
45. Square root Sum of squares/RMS calculation for Frequency Domain Data
46. Reciprocal Frequency Domain Data
47. Time-frequency analysis
48. Statistical tools
49. Create Poynting Vector or real/ imag E or H vector plot
50. Multiply Time Domain Data
51. Create Time Domain Force Vector Animation in conducting volumes
52. Create_surface_frequency_domain_plot
53. Calculate filter impulse response
54. Create filter function (LPF, HPF)
55. interpolate function(s)

```
56. Subtract d.c. from Time Domain Data
57. Multiply Frequency domain Data
58. Post processing for time domain conducted emissions data
```

The action of the post processor is stored in the file GGI_TLM_post_process_in.txt. The process can be rerun by

GGI_TLM_post_process < GGI_TLM_post_process_in.txt

The GGI_TLM_post_process_in.txt file is annotated and should be simple enough to understand to allow complex post processing files to be constructed using the original file as a template.

The post processing procedures are described in the following sections:

14.1 Extract Time Domain Data

GGI_TLM writes all time domain data of a particular type (field output at a point for example) to a single file. This process allows a single output point to be extracted to a separate file.

14.2 Plot Time Domain Data

This process allows the plotting of simple time domain data i.e. field at a point or cable current.

14.3 Fourier Transform (Time Domain to Frequency Domain)

Simple time domain output data may be transformed into the frequency domain. The user will need to specify the minimum and maximum frequencies, the number of frequency samples to calculate and whether to use a linear or logarithmic frequency scale.

The Fourier transform of a function of time g(t), evaluated at a frequency, f, is:

$$G(f) = \int_0^\infty g(t) e^{-j\omega t} dt$$

Where the function g(t) is assumed to be zero outside the time period of the simulation. The Fourier integral is evaluated as

$$G(f) = \sum_{i=1}^{n} g(i\Delta t) e^{-j\omega i\Delta t} \Delta t$$

The frequency domain output file has the following form:

Frequency (Hz) output number Re{G(f)} Im{G(f)} |G(f)| phase (G(f)) $20\log_{10}(|G(f)|)$

The phase is output in radians.

14.4 Plot Frequency Domain Data

This option allows the plotting of frequency domain data to either the screen, gif or jpg file using gnuplot. The user may specify whether to plot real part, imaginary part, magnitude, phase or magnitude in dB.

14.5 Create Animation of Time Domain Surface Field/Volume Field/ Surface Current Output

Time domain data output on surfaces and within volumes may be turned into a format suitable for visualisation with **paraview**.

14.6 Combine Frequency Domain Data

It is often necessary to combine frequency domain data in order to produce the required result. This option allows frequency domain data to be combined in the form:

$$R(f) = \frac{a \times f_1(f)}{b \times f_2(f)} + c$$

Where $f_1(f)$ and $f_2(f)$ are complex functions of frequency (data from file produced by the Fourier Transform process) and a, b and c are real constants.

This form allows calculations such as normalisation of a response to an excitation, calculation of antenna impedance and calculation of S parameters. The frequency domain functions $f_1(f)$ and $f_2(f)$ must be evaluated at exactly the same frequencies.

14.7 Combine Frequency Domain Magnitude Data

This process combines the magnitude of Fourier transform data in much the same manner as the complex frequency domain data combination described in the previous section however only the magnitude of the Fourier Transform data is used.

14.8 Frequency average

This process allows the averaging of frequency domain data over a specified frequency band. The bandwidth may be either a fixed bandwidth or a bandwidth specified relative to the centre frequency of the band.

14.9 Oneport analysis

The oneport analysis allows the analysis of the reflection of a mode for example the reflection of a plane wave from a material or a waveguide mode from a waveguide termination. The oneport analysis requires four frequency domain inputs, these are two E field and two H field outputs separated by a small offset either side of the measurement port (figure 15.9.1)

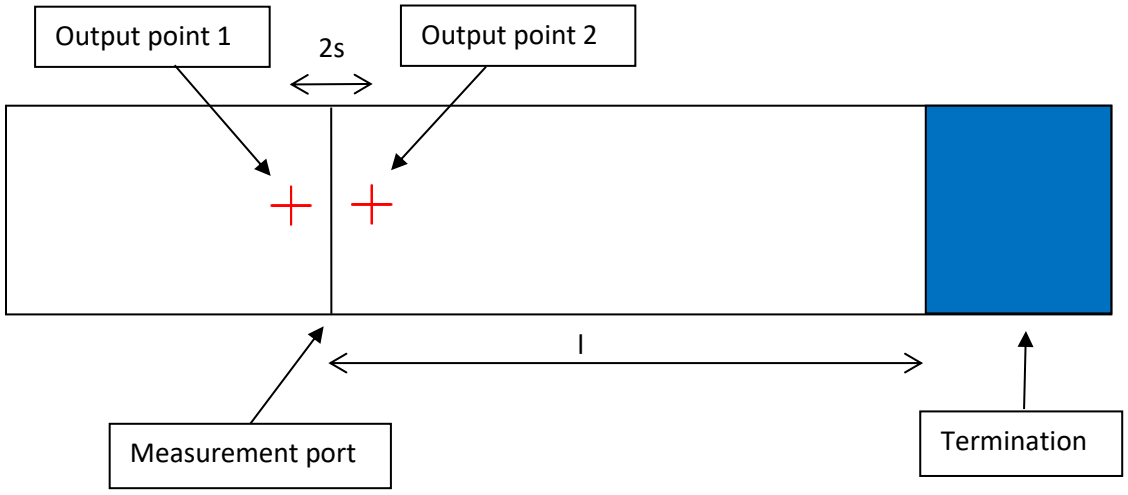


Figure 15.9.1.Configuration for oneport analysis.

In the frequency domain the fields at the output ports may be expressed as:

$$E_1 = E_i e^{j\beta(l+s)} + E_r e^{-j\beta(l+s)}$$

$$E_2 = E_i e^{j\beta(l-s)} + E_r e^{-j\beta(l-s)}$$

$$H_1 = \frac{E_i e^{j\beta(l+s)} - E_r e^{-j\beta(l+s)}}{Z}$$

$$H_2 = \frac{E_i e^{j\beta(l-s)} - E_r e^{-j\beta(l-s)}}{Z}$$

Where E_i is the mode E field incident on the termination, E_r is the mode E field reflected from the termination, β is the mode propagation constant and Z is the mode impedance. Note that E_i and E_r are referenced to the position of the termination.
Given frequency domain complex E_1, E_2, H_1 and H_2 plus the distances l and s, this system of equations may be solved for complex E_i, E_r, β and Z. The reflection coefficient at the interface is then given as:

$$R = \frac{E_r}{E_i}$$

Note that the oneport analysis assumes that there is only a single mode present, if there are any other propagating modes this will corrupt the analysis to some extent. Experience shows that the oneport analysis is very sensitive to this type of corruption.

14.10 Twoport analysis

The twoport analysis allows the analysis of the reflection *and* transmission of a mode for example the reflection and transmission of a plane wave from a material or a waveguide mode from a waveguide discontinuity. The twoport analysis requires eight frequency domain inputs, these are two E field and two H field outputs separated by a small offset either side of the two measurement ports (figure 15.10.1)

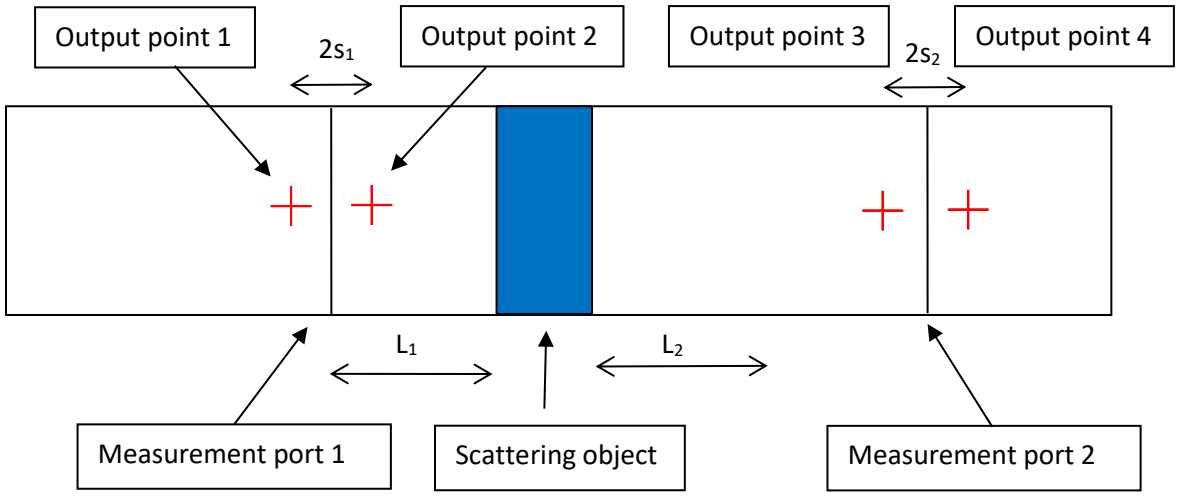


Figure 15.10.1.Configuration for twoport analysis.

In the frequency domain the fields at the output ports may be expressed as:

$$E_1 = E_{1i}e^{j\beta_1(l_1+s_1)} + E_{1r}e^{-j\beta_1(l_1+s_1)}$$

$$E_2 = E_{1i}e^{j\beta_1(l_1-s_1)} + E_{1r}e^{-j\beta_1(l_1-s_1)}$$

$$H_1 = \frac{E_{1i}e^{j\beta_1(l_1+s_1)} - E_{1r}e^{-j\beta_1(l_1+s_1)}}{Z_1}$$

$$H_2 = \frac{E_{1i}e^{j\beta_1(l_1-s_1)} - E_{1r}e^{-j\beta_1(l_1-s_1)}}{Z_1}$$

$$E_3 = E_{2i}e^{j\beta_2(l_2-s_2)} + E_{2r}e^{-j\beta_2(l_2-s_2)}$$

$$E_4 = E_{2i}e^{j\beta_2(l_2+s_2)} + E_{2r}e^{-j\beta_2(l_2+s_2)}$$

$$H_3 = \frac{-E_{2i}e^{j\beta_2(l_2-s_2)} + E_{2r}e^{-j\beta_2(l_2-s_2)}}{Z_2}$$

$$H_4 = \frac{-E_{2i}e^{j\beta_2(l_2+s_2)} + E_{2r}e^{-j\beta_2(l_2+s_2)}}{Z_2}$$

Where:
E_{1i} is the mode E field incident on the scatterer from side 1
E_{2i} is the mode E field incident on the scatterer from side 2
E_{1r} is the mode E field reflected from the scatterer on side 1
E_{2r} is the mode E field reflected from the scatterer on side 2
β_1 is the mode propagation constant on side 1
β_2 is the mode propagation constant on side 2
Z_1 is the mode impedance on side 1
Z_2 is the mode impedance on side 2

Note that incident and reflected fields are referenced to the position of the termination.
Given frequency domain complex E_1, E_2, E_3, E_4, H_1, H_2, H_3, and H_4 plus the distances l_1, l_2, s_1 and s_2 , this system of equations may be solved for complex E_{1i}, E_{1r}, E_{2i}, E_{2r}, β_1, β_2, Z_1 and Z_2.

Provided that we can assume that the excitation is at port 1 and there is no incident mode from side 2 of the scatterer then the S parameters S_{11} and S_{12} are given as

$$S_{11} = \frac{E_{1r}}{E_{1i}}$$

$$S_{12} = \frac{E_{2r}}{E_{1i}}$$

Similarly, provided that we can assume that the excitation is at port 2 and there is no incident mode from side 1 of the scatterer then the S parameters S_{21} and S_{22} are given as

$$S_{21} = \frac{E_{1r}}{E_{2i}}$$

$$S_{22} = \frac{E_{2r}}{E_{2i}}$$

Two simulations are therefore required in order to fully characterise a scatterer in this way.

Note that the twoport analysis assumes that there is only a single mode present at the field observation points, if there are any other propagating modes this will corrupt the analysis to some extent. Experience shows that the twoport analysis is very sensitive to this type of corruption.
The assumption of zero incident field from the transmission side of the scatterer does rely on either a very good absorber for the transmitted mode or a very long continuation of the problem space in order that no field is scattered back into the problem from the transmission side.

14.11 IELF analysis

The Integrated Error over Log Frequency is a parameter which may be used to assess the level of agreement between frequency domain data sets. The IELF value is defined as follows:

$$IELF_{\text{mod}} = \frac{\sum_{0}^{n} |error_n| \left\{ \ln\left(\frac{f_{n+1} + f_n}{2} \right) - \ln\left(\frac{f_n + f_{n-1}}{2} \right) \right\} / 2}{\ln(f_n) - \ln(f_0)}$$

A judgement on the degree of agreement between the data sets is given as follows:

IELF<1 : Excellent

1≤IELF<2 : Very Good

2≤IELF<4 : Good

4≤IELF<7 : Moderate

7≤IELF<10 : Poor

IELF ≥ 10 : Bad

14.12 Create Animation of Frequency Domain Surface Field Output

This option allows the creation of an animation of frequency domain surface field data.

14.13 Extract mode from Frequency Domain Surface Field Output

Mode data is extracted from frequency domain surface field output for subsequent use as a mode excitation or mode output.

14.14 Sum Frequency Domain Data

This option allows the sum of an arbitrary number of frequency domain functions to be summed and output to file. The frequency samples of all the functions to be added must coincide in order for this process to be applied.

14.15 Transmission cross section calculation

For the situation of a PEC or mode stirred cavity with a material window or aperture the Transmission Cross Section is calculated as

$$TCS = \frac{Transmitted\ Power}{Incident\ Power\ Density}$$

The process takes frequency domain transmitted power and a singe component of the cavity internal field as input. The incident power density is calculated as

$$Internal\ Power\ Density = \frac{3}{2}\frac{E_{RMS}^2}{Z_0}$$

The frequency samples of transmitted power and Incident field must coincide.

14.16 Fourier Transform with frequency warping (Time Domain to Frequency Domain)

A Fourier transform which undoes the frequency warping introduced by the bilinear transformation applied in the frequency dependent material/ thin layer/ impedance implementation. This is experimental and is designed to be used alongside techniques for expanding the frequency range of simulations in over-moded situations. See C.Smartt, P. Sewell, C. Christopoulos, “Expanding the Useful Bandwidth of Time Domain CEM Techniques in HIRF Analysis” EMC Europe, Brugge, 2013.

14.17 Create vector field Animation of Frequency Domain Surface Field Output

This option allows the creation of a vector field animation of frequency domain surface field data. Frequency domain data is required for each field component to be included in the animation.

14.18 Create vector field Animation of Frequency Domain Volume Field Output

This option allows the creation of a vector field animation of frequency domain volume field data. Frequency domain data is required for each field component to be included in the animation.

14.19 S parameter to Z parameter transformation

Convert Scattering (S) parameters to impedance (Z) parameters, given the impedance of ports 1 and 2.

14.20 Calculate correlation matrix from time domain data

Calculate the correlation matrix from a number of time domain data sequences.

14.21 Calculate correlation matrix from frequency domain data

Calculate the correlation matrix from a number of frequency domain data sequences.

14.22 Calculate complex antenna factor from (S21) measurement for two identical antennas

Given a simulation of coupling between two identical antennas, calculate the complex antenna factor (E/V) for the antenna.

14.23 Calculate complex antenna factor from antenna coupling (S21) measurement with one unknown antenna

Given a simulation of coupling between an antenna with known antenna factor, and an unknown antenna, calculate the complex antenna factor (E/V) for the unknown antenna.

14.24 Fast Fourier Transform (Time Domain to Frequency Domain)

Fourier transform time domain data to the frequency domain using the FFT algorithm.

14.25 Generate random sample sequence either with or without an imposed correlation matrix

This process creates sequences of random numbers which may have a correlation (described by a correlation matrix) imposed on them. These sequences can be used to investigate radiation from partially correlated sources.

14.26 Apply a filter function to time domain data

A specified filter function to a time domain data set and write the resulting time response to file.

14.27 Apply a filter function to frequency domain data

A specified filter function to a frequency domain data set and write the resulting frequency response to file.

14.28 Calculate the frequency domain transfer function of a filter function

For a specified filter function (transfer function) in the normal rational function form, write the frequency response of the transfer function to file.

14.29 Combine Frequency Domain Data: S(f)=f1(f) f2(f)

It is often necessary to combine frequency domain data in order to produce the required result. This option allows frequency domain data to be combined in the form S(f)=f1(f)f2(f) where $f_1(f)$ and $f_2(f)$ are complex functions of frequency (data from file produced by the Fourier Transform process). This process also allows the complex conjugate of f2(f) to be used i.e. $S(f)=f1(f)f2^*(f)$

14.30 Calculate PDF and CDF from a data set

From a set of sample data (single column of data taken from a data file) calculate the probability density function and the cumulative density function from the samples.

14.31 Calculate correlation function from time domain data

Calculate the correlation function of two time domain data sequences over a specified range of time lags.

14.32 Calculate correlation function from frequency domain data

Calculate the correlation function of two frequency domain data sequences over a specified range of frequencies.

14.33 Create Vector Animation of Time Domain Volume Field Output

This option allows the creation of a vector field animation of time domain volume field data.

14.34 Create time domain near field scan

Read time domain output over a volume and output in the form:

X y z time Field_x Field_y Field_z

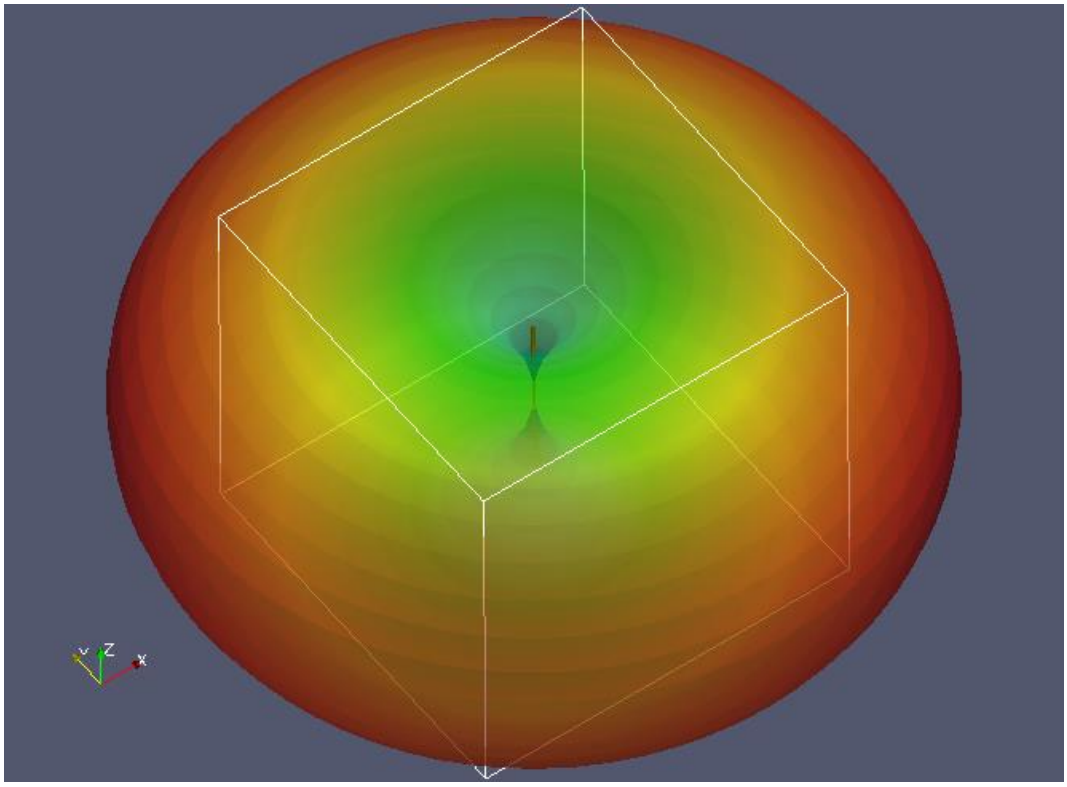
14.35 Set random_number_seed

The random number generator is used in GGI_TLM_post_process in the generation of random values for the correlated noise source calculation.

The seed values are set to random values based on date, time and process id, the results will then be different on each occasion on which the code is run.

14.36 Generate Far field plot data

Read a name.far_field.fout file and create a 3D surface plot of the far field pattern for plotting in paraview. The far field may be normalized to scale the 3D plot for plotting together with the problem geometry. An example for the dipole test case is shown below.



14.37 Re-format data file

This process allows the user to re-format tabulated data sets for example to re-arrange columns of data, output data over a specified range of values, read and convert csv format files, scale columns of data (for example to convert GHz to Hz).

The processes available are:

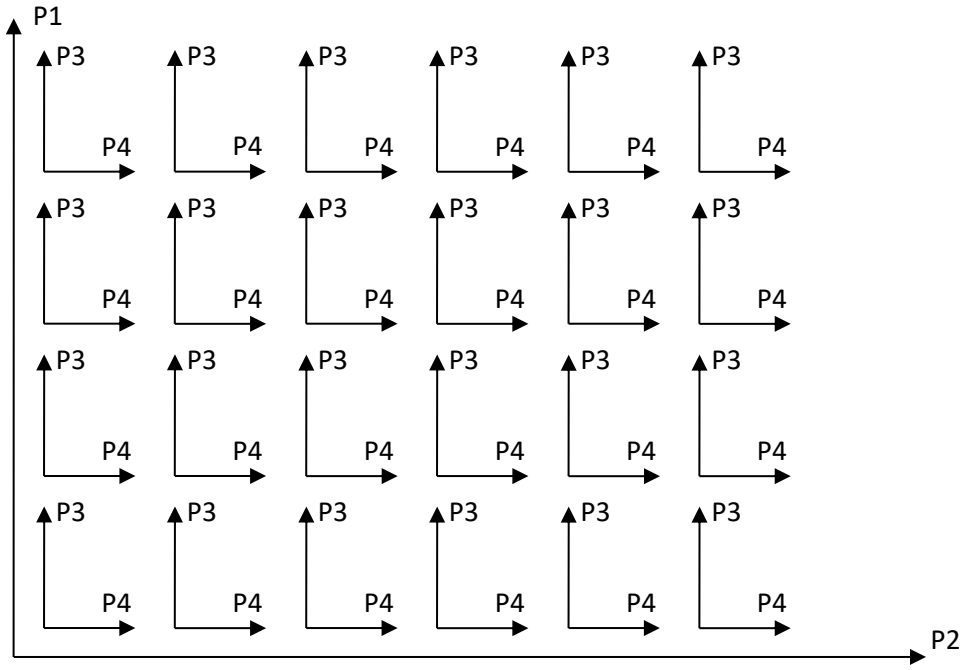
1. Strip header data from a file
2. Read csv format data
3. Re-arrange columns of data
4. Multiply a column by a given factor (say to convert from GHz to Hz)
5. Only write data within a given parameter range
6. Combine data files

14.38 Visualise multi-dimensional data sets (up to 4D)

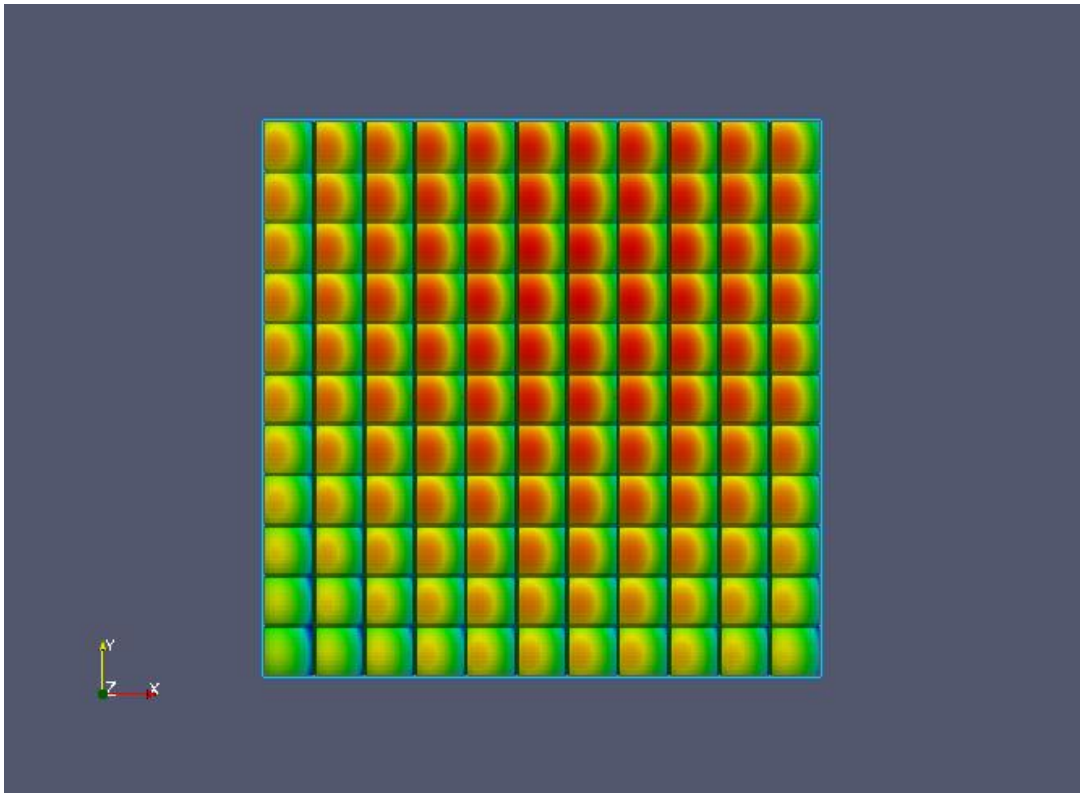
This process allows multi-dimensional data sets (up to 4D) to be plotted in paraview. For example a 4D data file in which a value is a function rof four parameters p1-p4 of the form:

P1 p2 p3 p4 value

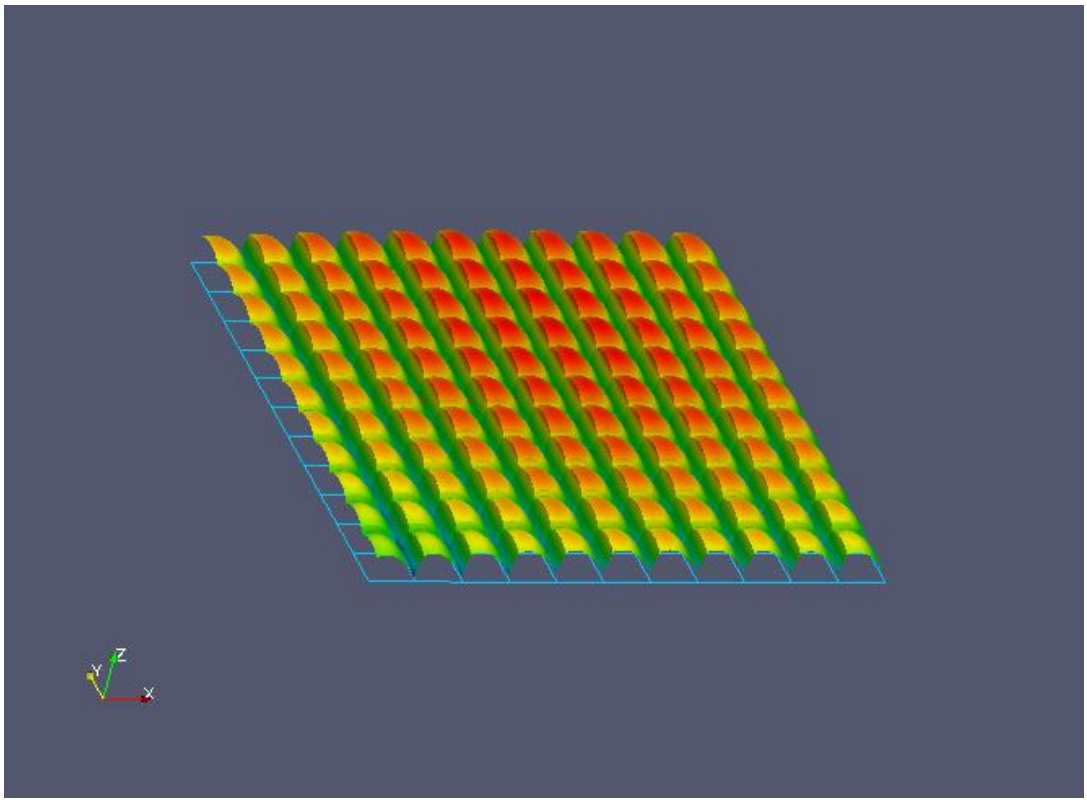
Can be plotted as shown in the figure below:



I.e. in each tile p4 varies along the horizontal axis and p3 along the vertical axis. From tile to tile in the horizontal direction, p2 varies and from tile to tile in the vertical direction, p1 varies. An example is shown below.



Visulisation of a 4D data set in paraview



Visulisation of a 4D data set (3D view) in paraview

14.39 Calculate the impulse response of a filter function

Read a filter function in the format:
w normalisation constant (real)
numerator order (integer)
numerator coefficients (numerator_order+1 * real)
denominator order (integer)
denominator coefficients (denominator_order+1 * real)

A timestep and maximum time are requested.

The impulse response of this filter function is calculated by firstly applying
a bilinear transform to calculate the Z domain filter function and subsequently
calculating the impulse response of the z domain filter function.

14.40 S11 to VSWR

Complex frequency domain S11 data is read from a file
VSWR is then calculated as (1+|S11|)/(1-|S11|)
The VSWR is then written to file.

14.41 Convert Frequency Domain Volume Field Output to x y z Re{field} Im{field} |field| format

Frequency domain volume field output is read.

For each chosen volume the field output dataset (which is based on cell centre data)
the field data from each cell in the volume is written out to a text based file in the format:
x y z Re{field} Im{field} |field|

where x y z are the coordinates of the centre of the cell.

14.42 Calculate spatial covariance for 1D and 2D complex data sets

This process calculates the covariance for an ensemble of 1D and 2D complex data sets (a 1D dataset here is a 2D dataset with only one sample in the second direction).
The process reads a datafile which contains the input complex dataset. The columns may be in an arbitrary order however there should be data columns for x, y, Re{f(x,y)}, Im{f(x,y)}

The output of the process is a 4D dataset (for 2D input dataset) with the format:

x1 y1 x2 y2 Re{C(x1,y1,x2,y2)} Im{C(x1,y1,x2,y2)} |C(x1,y1,x2,y2)|

where

$$C(\mathrm{x1},\mathrm{y1},\mathrm{x2},\mathrm{y2}) = \sum_{ns}(\mathrm{f(x1,y1)} - <\mathrm{f(x1,y1)}>)(\mathrm{f(x2,y2)} - <\mathrm{f(x2,y2)}>)^{*}$$

C(x1,y1,x2,y2)=sum(n_ensemble) ((f(x1,y1)-<f(x1,y1)>)(f(x2,y2)-<f(x2,y2)>)*)/ns

where * denotes the complex conjugate and

ns=n_ensemble if we assume a known zero mean OR
ns=n_ensemble-1 if we use the sample mean

and <f(xi,yj)> is the mean value of f(xi,yj) over the ensemble of data at point xi,yj

$$< \mathrm{f(xi,yi)} >= \frac{1}{n\_ensemble} \sum_{n\_ensemble} \mathrm{f(xi,yi)}$$

Spatial covariance functions may be plotted using the nd visualization process.

14.43 Calculate and propagate Wigner function from complex spatial correlation data

(Reference required)

Read covariance data (from process calculate spatial covariance)
Enter the number of samples in p required (the spectral variable
Enter the frequency
Enter the maximum value of p required (range of p is from -pmax to pmax) pmax<=1 excludes evanescent waves
Enter the z propagation distance
Respond as to whether you want to propagate the evanescent field or set it to zero
Calculate the Wigner distribution function
Write the Wigner function to file
Propagate the Wigner function a distance z using the Frobenius Perron method
Write the propagated Wigner function to file
Evaluate the covariance function from the propagated Wigner distribution function
Write the propagated covariance function to file

Spatial covariance and Wigner functions may be plotted using the nd visualization process.

14.44 Sum Time Domain Data

Read a number of time domain data files, f_n, and sum the time domain data with specified weighting factors, w_n.

$$Result = \sum_{1..N} fn \times W_n$$

14.45 Square root Sum of squares/RMS calculation for Frequency Domain Data

calculate the square root of the sum of squares or RMS value of a set of complex frequency domain datasets

The user should enter the nmumber of frequency domain datasets to combine and the filenames for each of the datasets. The frequencies for each of the datasets must be the same.

14.46 Reciprocal Frequency Domain Data

Calculate Result=1/fn where fn is a complex frequency domain quantity

14.47 Time-frequency analysis

Time-frequency analysis i.e. perform a short time FFT on a windowed version of the input time sequence, allowing the window to move through the input time domain data

Note that this process operates on simple data files with no header data i.e. it could come from measurement or wherever...

The input format is assumed to be two column data:

time value

GGI_TLM output files will require format conversion before using this process i.e. post processing process number 37: Re-format data file

The user is asked to enter the width of the time window (in timesteps) and then enter the number of time windows to analyse in the time-frequency analysis. The frequency range for the output should then be specified and the number of frequency samples in this range.

The time-frequency analysis results are output in to a file in the three column format:

frequency time magnitude (or frequency time 20log10(magnitude) if dB output has been requested)

14.48 Statistical tools

Read a set of data samples from a file. The data samples may be transfmored if required as follows:

Convert from dB to field amplitude
Convert from field amplitude to dB
Convert from field to field magnitude

The distribution of the sample data is then calculated by putting the data into a specified number of sample bins.

The probability density function and the cumulative density function of the data are written to file.

The mean and variance of the data are also calculated

14.49 Create Poynting Vector or real/ imag E or H vector plot

Read frequency domain vector E and H fields and calculate the Poynting vector P=ExH* then plot average power flow or Reactive power or real and imaginary E and H field vectors may also be plotted

14.50 Multiply Time Domain Data

Read a number, N, of time domain data files, f_n and multiply the time domain data with specified weighting factors, Wn

$$Result = \prod_{1..N} f_n \times W_n$$

14.51 Create Time Domain Force Vector Animation in conducting volumes

Read time domain output over volumes and create a vector animation of forces on conductors from the data

All components of Electric and Magnetic field are specified in a volume from time domain volume output data.

The electrical conductivity is specified and the reduced_c factor used in the simulation (set to 1.0 in reduced_c has not been used).

Calculate the force vector on this cell
Force per unit length is F=IxB=dA*sigma ExB= dA*sigma*mu0 ExH
then multiply by dl to give the total force on the cell

A vector plot of magnetic force is then created where the force is measured in N with one vector within each cell of the output volume.
Note please use with caution when using reduced_c as this is little tested at the moment.

14.52 Create_surface_frequency_domain_plot

This process produces a surface plot which shows the magnitude of frequency domain data based on frequency_output_surface data

14.53 Calculate filter impulse response

As for 39. Calculate the impulse response of a filter function except the impulse response may be calculated for a material filter applied to a problem with a specifiend TLM cell size (the timestep is calculated from this.

14.54 Create filter function (LPF, HPF)

The process creates s domain filter functions for low pass or high pass butterworth filters of orders between 1 and 9. Note that higher order filters may not lead to accurate time domain filter implementations due to the specification as high order rational functions. These filters are better implemented using cascaded lower order functions.

The filters are output in the standard frequency domain format. The cutoff frequency is used to set the normalisation constant and the filter coefficients are set for a 1 rad/second cutoff.

The format is as follows:

w normalisation constant (real)
numerator order (integer)
numerator coefficients (numerator_order+1 * real)
denominator order (integer)
denominator coefficients (denominator_order+1 * real)

14.55 Interpolate function(s)

Read a data file with a set of x values x1 or alternatively set a uniform set of n_samples sampling points x1 from x1min to x1max.
Read a second file with a set of real or complex function values f(x) at a set of x values x2 interpolate the values f(x) to the set of samples x1.

14.56 Subtract d.c. from Time Domain Data

Read a time domain dataset, calculate the d.c. value and write a time domain output dataset with the d.c. value subtracted.

14.57 Multiply Frequency domain Data

Scale complex frequency domain data by a real scale factor

14.58 Post processing for time domain conducted emissions data

This is documented separately in the separate document:

GGI_TLM_Ngspice_documentation