Incorporating Ngspice circuit models into GGI_TLM

Contents

Introduction

GGI_TLM is open source software for the time domain simulation of Electromagnetic fields [1] with applications including EMC, Antenna design and scattering analysis. It is an implementation of the TLM method [2], a 3D time domain electromagnetic field simulation technique based on an equivalence between electric and magnetic fields and voltages and currents on a network of transmission lines. The GGI_TLM software includes the ability to include thin wires, including shielded cables, frequency dependent materials and thin layers into models. In order to enable GGI_TLM to simulate circuits consisting of lumped components, including active devices, the open source circuit simulation tool Ngspice [3] has been linked into the simulation software. This enables 3D models of PCBs to be created and simulated in the time domain which allows both conducted and radiated emissions to be assessed. This capability allows dc-dc converters to be simulated for example.

In order for a 3D circuit simulation to be performed the layout of the circuit must be specified. This layout is most commonly available in the gerber format [4] which is used for design and manufacture of PCBs. In addition, the circuit topology and spice component models must be specified.

There are a number of published approaches to the embedding of Spice models in TLM [5]-[9] however in order to simplify the interface between GGI_TLM and Ngspice, lumped components are placed on faces between TLM cells in the problem space.

In order to aid the model creation process some tools have been developed which bring together the required information and automatically creates the appropriate input files for GGI_TLM. These are:

GGI_TLM_create_PCB_simulation_model

This tool creates the main input file for GGI_TLM by bringing together appropriate information, data files and processes. The process sequence is as follows:

1. Specify the problem space dimensions

2. Specify the number of PCB layers to include in the model. For each layer, convert the gerber file to stl format using the **GGI_TLM_gerber_to_stl** process and specify the position within the problem space to place the PCB layer

3. Specify the number of dielectric layers. For each layer specify the position in space of the dielectric and specify the material properties (i.e. give the name of the appropriate material file.)

4. Specify the number of vias to include and their positions

5. Specify the number of lumped components. For each lumped component:

i. Specify the number of ports and port number(s)

ii. Provide the ngspice node numbers corresponding to each port

iii. for each connection, provide the x, y and z connection point coordinates

iv. z position of the component

v. package information (dimensions, material model for package, any conducting surfaces on package (e.g. transistor case heatsink contact plate)

4. Specify the number of additional components (heatsinks or dielectric volumes (eg ceramic tiles for heatsink connection or output volumes))

For each additional component, component details (physical dimensions and material model)

5. Give additional information to be included in the GGI_TLM input file

e.g.

Ngpspice node outputs

Ngspice options (ngspice_timestep_factor, ngspice_lpf_alpha)

Additional outputs (e.g E and H fields, current density on surfaces, E and H fields on volumes)

simulation_time

6. Give information to be included in the Spice circuit file

Components to be attached to ports (with Ngspice nodes specified above)

GGI_TLM_gerber_to_stl

This tool converts files in gerber format to stl format (surface triangulation format) which allows the PCB layer geometries to be imported into GGI_TLM.

Linking GGI_TLM and Spice

A 3D model of a circuit is a combination of lumped and distributed elements. Lumped circuit elements such as diodes, transistors etc. will be simulated using Ngspice and the distributed elements such as PCB tracks, heatsinks etc. will be simulated in 3D using GGI_TLM. Since GGI_TLM is based on an equivalent circuit principle the link between the TLM based circuit simulator and the Spice based circuit simulation is straightforward in principle.

The most basic example of the way in which a lumped circuit element is embedded within a TLM simulation is embedding of a single two terminal lumped component into a TLM model for example a diode load on a wire. This is illustrated in Figure 1. In this model, the wire is represented in the 3D TLM model as short circuits in the link lines between the TLM nodes. The presence of the short circuit will be taken into account in the TLM model by locally modifying the connect procedure.

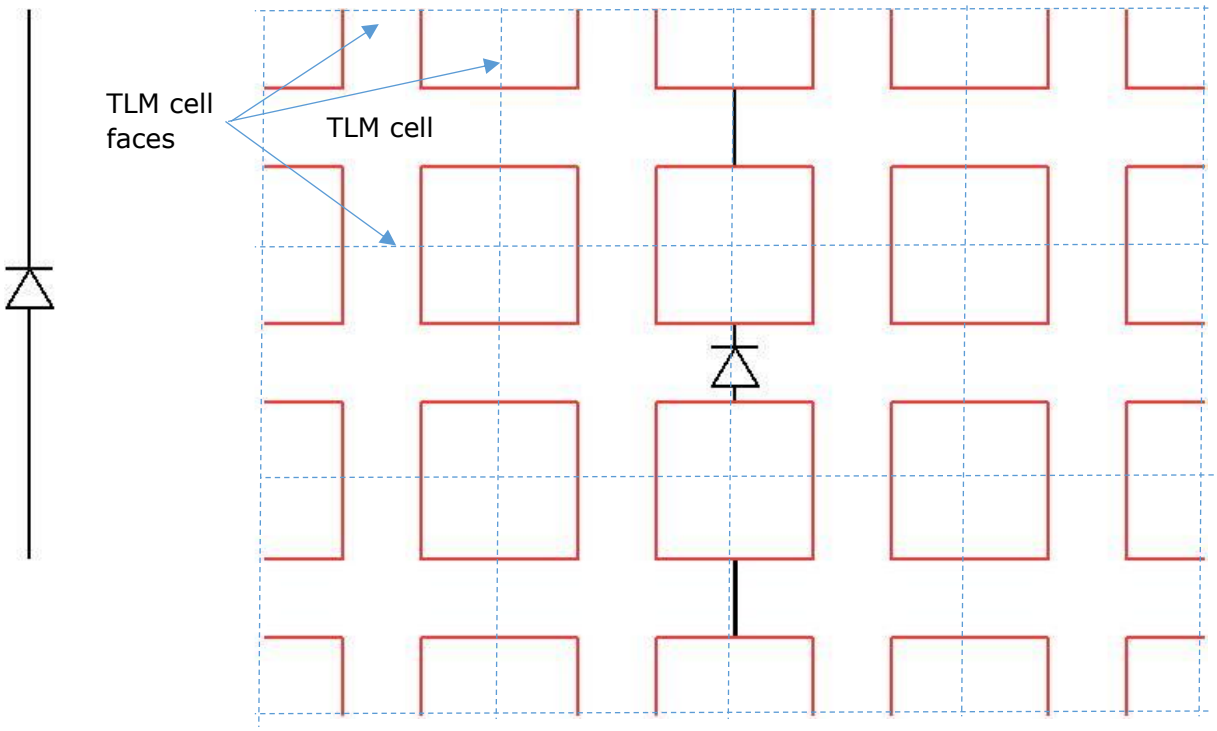

Figure 1. Diode at the centre of a conductor and the TLM model of this situation with the diode and conductors placed at cell faces. Link lines are shown in orange and TLM cell faces as dotted blue lines.

The diode may also be placed on the link lines at a TLM cell face and the connect procedure must now take into account the presence of the diode. In order to do this we may draw a Thevenin equivalent circuit for the connect process as seen in Figure 2.

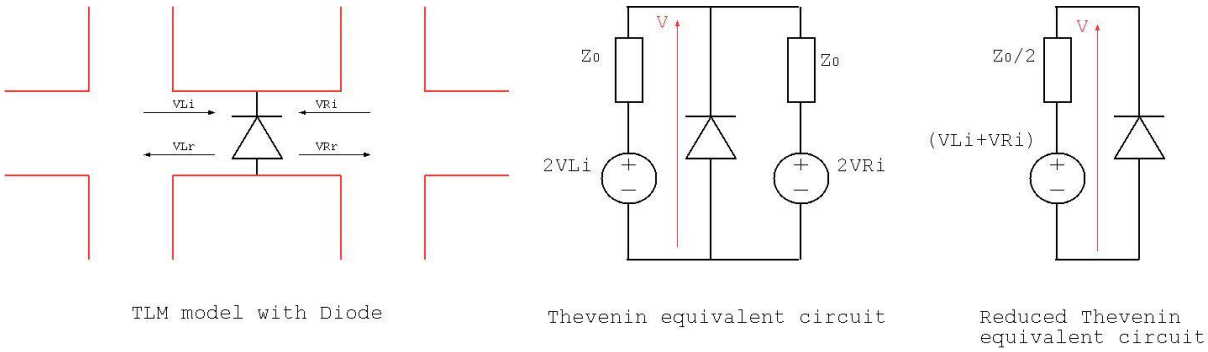

Figure 2. Example diode model and the Thevenin equivalent model of the TLM-Ngspice link circuit

The central figure shows the Thevenin equivalent circuit of the two link lines interacting with the diode. These two Thevenin equivalent circuits may be combined to give the simple circuit on the right hand side of Figure 2 in which a resistance of $Z_0/2$ is in series with a voltage source whose value depends on the voltage pulses incident on the diode from the link lines (VLi+VRi). This circuit may be solved using Ngspice for the diode voltage, V, and hence the scattered voltages in the TLM solution may be calculated from:

$$VLr = V - VLi$$

$$VRr = V - VRi$$

Filtering the interaction of GGI_TLM and Ngspice processes

The implementation of the GGI_TLM-Ngspice link requires voltages to be exchanged between the Ngspice and GGI_TLM processes at each timestep in the connection process i.e. GGI_TLM must send the link line voltages, (VLi+VRi), to Ngspice. Ngspice is then required to calculate the component voltage, V, and return this to GGI_TLM so that the scattered voltages from the component can be returned to the link lines. It has been found that high frequency oscillations may be excited in this process – above the normal cutoff frequency of the TLM solution. In this work we have implemented the TLM-Ngspice link using a low-pass filter with an adjustable parameter, α, to control the cutoff frequency. This approach recognises the potential for the hybrid solution to develop oscillations whilst also bearing in mind the limited bandwidth of validity of the TLM solution due to dispersion issues. Typically the minimum wavelength (maximum frequency) of validity of a TLM solution is determined from the requirement for 10 cells per wavelength i.e. $\lambda min = 10\Delta l$ or $fmax = c/(10\Delta l)$=1/(20Δt) where c is the speed of light.

A simple first order low pass filter is implemented with the transfer function

$$H(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{1}{1 + \alpha s}$$

A discrete time implementation may be derived using the bilinear transformation to the z-domain

$$s \to \frac{2}{\Delta t}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right)$$

Where z^{-1} indicates a delay of one timestep. The first order discrete time filter function is found to be implemented by the formula

$$V_{out}(t) = \frac{V_{in}(t) - V_{in}(t - \Delta t) - k_2 V_{out}(t - \Delta t)}{k_1}$$

Where

$$k_1 = 1 + \frac{2\alpha}{\Delta t}$$

And

$$k_2 = 1 - \frac{2\alpha}{\Delta t}$$

In practice the cutoff frequency parameter, α, is normalised to the TLM timestep, Δt such that α'=α/Δt. When α'=0 the filter passes everything unchanged i.e. it does not act as a filter at all. A value of α'=0.1 should be sufficient to prevent any high frequency oscillation and instability without introducing too much loss into the system. This parameter may be set in the GGI_TLM input file as described below.

GGI_TLM – Ngspice linking process

The voltage pulses incident on the component are available following the scattering process i.e. at time nΔt. The voltage pulses scattered from the component must be returned to GGI_TLM for the next scattering process i.e. at (n+1) Δt thus we have a time period of Δt which the GGI_TLM – Ngspice link algorithm can utilise.

TLM is normally assumed to be based on the propagation of time synchronised impulses on a network of transmission lines. The Spice circuit simulation method on the other hand is based on a modified nodal analysis technique with a trapezoidal integration scheme and variable time stepping thus care must be taken in how TLM and Spice are interfaced; the propagation of impulses in non-linear, dispersive circuits is not possible without dispersion of the impulses.

It is normally assumed that the 'connect' operation in TLM is instantaneous, and occurs at time (n+½)Δt however when we are coupling to a lumped circuit solution we may be required to modify this process in order to maintain stability of the coupled system. A number of different algorithms are discussed in [6] in which algorithms based on averaging and integration of the voltage calculated in Spice are proposed.

For convenience of the GGI_TLM- Ngspice link, Ngspice is run at a timestep smaller than the TLM timestep, in practice the Spice timestep cannot be imposed as a variable timestep algorithm is used.

The interaction of GGI_TLM and Ngspice implemented in this work is summarised as follows:

Time=nΔt:

- Scattering process at TLM nodes.
- Calculate the Ngspice link voltage Vtlm=VLi+VLr
- Apply the Low Pass Filter process to Vtlm as required to give Vtlm'
- Set a breakpoint in Ngspice at (n+1)Δt and resume the Ngspice simulation using the voltage source value Vtlm'

Time=(n+½)Δt:

- Connection process at TLM cell faces

Time=(n+1)Δt.

- Transfer the component voltage, Vspice from Ngspice to GGI_TLM
- Apply the Low Pass Filter process to Vspice as required to give Vspice'
- Calculate VLr=Vspice'-VLi, VRr=Vspice'-VRi

There is a complication that Ngspice can use variable timestepping and is therefore not guaranteed to use a specific timestep however with a sufficiently small Ngspice maximum timestep set, and the use of the Low Pass Filter process it is assumed that this will not lead to too much accumulated error. This issue does need to be studied in more detail as for some applications it may become significant however for dc-dc converter simulations, bearing in mind the very small TLM timestep (1.67ps for a 1mm mesh size) this issue has not been found to cause any problems.

Voltage references in GGI_TLM-Ngspice models

In a circuit simulation in Ngspice for example, all voltages are referenced to a single node, node zero. Once the Ngspice circuit model is separated into a multi-port model linked by GGI_TLM there is no common reference by which absolute circuit voltages may be determined as the voltage drops in both the Spice circuit model and in the TLM model

must be taken into account. Voltages across individual components may be output directly from GGI_TLM however great care must be taken in calculating voltages relative to a particular reference node as this will involve integration of the Electric field in the 3D TLM solution and this in turn may depend on the integration path through the TLM mesh.

Interaction between GGI_TLM and Ngspice: setting up the input files

GGI_TLM with Ngspice requires two input files:

name.inp, the usual GGI_TLM input file which contains details of the geometry, TLM excitations, outputs, simulation time etc plus the surfaces to which Ngspice circuit elements are connected (along with the associated Ngspice nodes).

Spice_circuit_TEMPLATE.cir which contains the lumped circuit elements which are to be embedded in the TLM solution with corresponding node numbering.

Before the simulation starts the Ngspice circuit file is created from the template circuit file so as to include GGI_TLM Thevenin equivalent circuit impedance values, timestep values and simulation time.

The circuit file must include the line:

Vbreak time_node 0 DC 0.0

This voltage source is used to control the breakpoints in the Ngspice simulation. At each timestep a breakpoint is set so that Ngspice will stop after the simulation time is increased by Δt, the GGI_TLM timestep. The breakpoint is set using the command 'stop when time > v(time_node)', the voltage source connected between node 0 and time_node is set to the breakpoint time (this eliminates the need to continuously set and remove breakpoint commands.)

In the software Ngspice is run in a separate thread.

Following the scattering process the GGI_TLM voltage pulses are updated in the Ngspice model using Ngspice commands of the form 'alter Vtlm1 = 0.325'. Similarly the next breakpoint time is set with a command of the form 'alter V(time_node) = 1.02e-10'. Once the new voltages have been set the Ngspice simulation is restarted with the Ngspice command 'bg_resume'.

Once Ngspice stops again at the specified breakpoint the voltages on all nodes from 1 to 100 are returned to GGI_TLM in an array (for this reason nodes between 1 and 100 should be reserved for nodes which interact with GGI_TLM.) These node voltages are then used to calculate the voltage pulses scattered from the Ngspice component.

The interaction between Ngspice and GGI_TLM may be controlled by two parameters in the GGI_TLM input file.

Ngspice_timestep_factor sets the number of timesteps in Ngspice for each GGI_TLM timestep. The default is set to 8.

Example:

Ngspice_timestep_factor

4

The second parameter is the low pass filter parameter, alpha. The default value is 0 (i.e. no low pass filter is applied to the ngspice link).

Example

Ngspice_LPF_alpha

0.10

Compiling and running GGI_TLM with Ngspice

The implementation of the GGI_TLM – Ngspice link requires Ngspice to be compiled as a library which may be linked to GGI_TLM during the make process.

The first stage is to compile Ngspice as follows:

Download Ngspice (here Ngspice-30). (Make sure that you download the file ngspice-30.tar.gz (the default download from the green button may be different on a windows machine))

Unpack the .tar.gz file.

Ngspice is then built using the following sequence of commands:

cd ngspice-30

mkdir ngspice_GGI_TLM

cd ngspice_GGI_TLM

../configure --with-ngshared --enable-xspice --enable-cider

make clean

make

sudo make install

cd ../..

Download GGI_TLM from github [1] then compile as follows:

cd GGI_TLM/SRC

make clean

make NGSPICE=TRUE

The appropriate paths to the Ngspice libraries are set in the Makefile (Makefile_GGI_TLM). You may need to ensure that the LD_LIBRARY_PATH environment variable is set appropriately (so as to include the path to the ngspice library) to run GGI_TLM_SEQ with Ngspice e.g

export LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib

Currently GGI_TLM linked to Ngspice can only be run with the sequential version of GGI_TLM.

There are some test cases in the directory GGI_TLM/TEST_DATA which demonstrate the GGI_TLM – Ngspice link. These are:

NGSPICE_DIODE_Test

PCB_SIMULATION_Test

PCB_SIMULATION_Test_2

PCB_SIMULATION_Test_3

PCB_PORT_Test_1

PCB_PORT_Test_2

NGSPICE_CONVERTER_MODEL

The Ngspice link test cases may be run by going to the **TEST_DATA** directory and using the command:

run_automatic_test run_seq NGSPICE_TEST_CASE_LIST

The user can plot the results with the command:

run_automatic_test plot NGSPICE_TEST_CASE_LIST

And the results can be checked against the available reference results with the command:

run_automatic_test check_reference NGSPICE_TEST_CASE_LIST

If only a single test case is to be run then replace NGSPICE_TEST_CASE_LIST in the above commands with the name of the specific test case e.g.

run_automatic_test run_seq NGSPICE_CONVERTER_MODEL

GGI_TLM Model Preparation

The GGI_TLM- Ngspice coupled simulation process required both a GGI_TLM input file (**name.inp**) plus any associated material and cable specification files and in addition an Ngspice circuit file which must have the name **Spice_circuit_TEMPLATE.cir**.

As described in the Theory section above, Ngspice circuit elements are implemented in GGI_TLM on surfaces between TLM cells they are therefore specified in the GGI_TLM input file in the Surface_material_list packet. They have the surface type 'SPICE'. The Ngpsice port number must be given – this indicates the number of the TLM port source voltage number, np, in the circuit file. In addition the Ngspice node numbers on the + and – sides of the component are given. The direction of the component (+x, -x, +y, -y, +z or -z) must also be specified. It is assumed that the surface material consists of a single TLM cell face and that the direction of the component lies in the plane of the TLM cell face.

The format to specify an Ngspice circuit element in the Surface_material_list is illustrated in the example below:

```
 Surface_material_list

1 # Number of surface materials

1 # Surface material number

SPICE

1 #Ngspice port number
```

```
1 0 # Ngspice node numbers on + and – terminals of the surface

-y # port direction i.e. the direction of the component in 3D space

1 # number of surfaces (this should always be 1 for a SPICE surface

5 # Surface list (this is the number of the surface in the Surface_list)

1 # Surface orientation list (this has no effect as the Spice interface is
symmetrical)
```

Ngspice port and node numbering

The Ngspice port number is an integer required to uniquely identify the GGI_TLM Thevenin equivalent circuit in the Spice circuit file. This is shown in red in Figure 3. The Ngspice node numbers are the nodes associated with the + and – terminals of the Ngspice port, these are shown in blue in Figure 3. The Ngspice node numbers should be integers between 0 and 99 as currently these are the only nodes which support the transfer of voltage information between GGI_TLM and Ngspice. It is recommended that node numbers below 100 are reserved for the link between GGI_TLM and Ngspice and nodes which are internal to the Ngspice circuit whose voltages are not required to be communicated between processes should take numbers outside this range (>1000 in the example below).

The portion of the spice circuit file corresponding to figure 3 is shown below:

```
*Voltage source with series resistance: equivalent circuit of TLM link

Vtlm1 1001 0 DC 0.0

Rtlm1 1001 1 #Z0_TLM
```

The component (or network) to be embedded is then connected between nodes 1 and 0 in the Spice circuit file.

(note that the source voltage is updated at every TLM timestep and the value of the resistance #Z0_TLM is substituted with the appropriate value automatically at the start of the simulation.)

The order of the node specification is important as it specifies the orientation of the Ngspice circuit element model in the GGI_TLM mesh. Reversing the Ngspice nodes reverses the orientation of the circuit element. This may also be achieved by reversing the port direction (e.g. +z -> -z.)

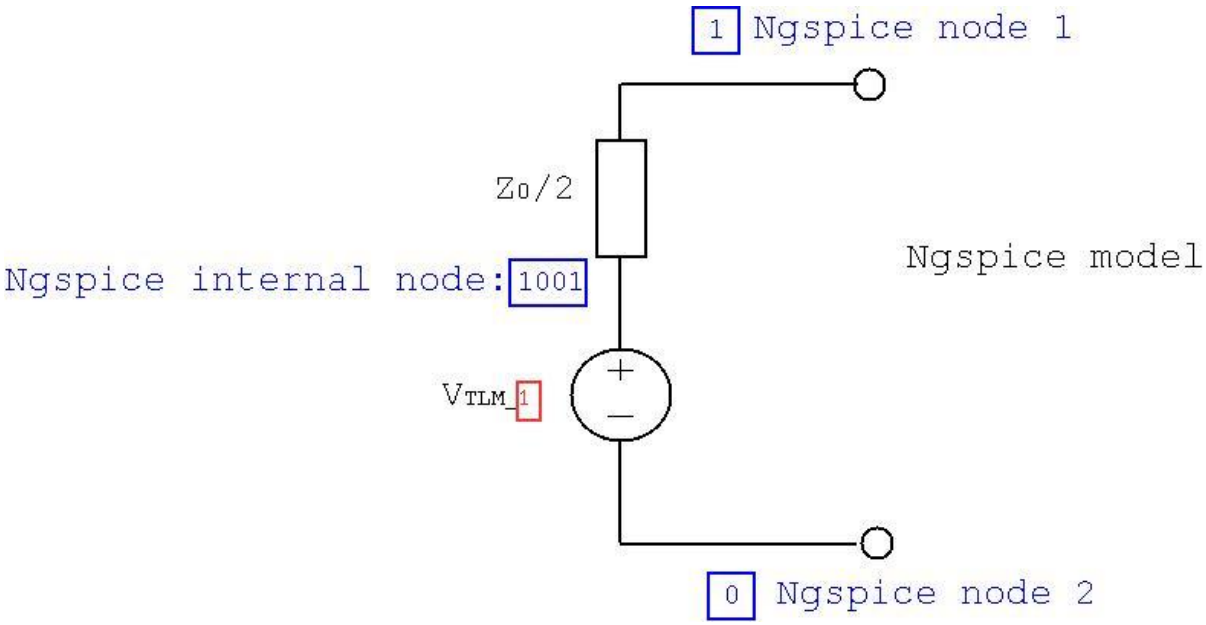

Figure 3 GGI_TLM-Ngspice link circuit showing port (red) and node (blue) numbering.

The port direction specified the direction of the component in the 3D TLM mesh. The component direction must be in the plane of the defined surface e.g. if the surface is normal to z then the possible port directions are +x, -x, +y and -y. Following the port direction comes the number of the geometric surface that the component model should be placed on the format of this information is the same as that for any other surface material. The surface should consist of a single TLM face i.e. Ngspice components cannot (yet) be distributed over multiple TLM faces (although this is possible in principle).

GGI_TLM_create_PCB_simulation_model

The **GGI_TLM_create_PCB_simulation_model** software has been developed in order to aid the setting up of linked GGI_TLM/ Ngspice models. The purpose of this is to simplify the setup of the simulation files for PCB simulations in GGI_TLM.

GGI_TLM_create_PCB_simulation_model generates surface geometry for the PCB structure from gerber files. Dielectric layers may then be specified in order to model the PCB substrate materials. Vias connecting different layers are then specified. Following this, lumped component models are specified by their terminal coordinates and the z position of the component (height above the PCB layer).

The process creates the geometry for the connecting wires and the surface which will be used as the 'active' element. Models of both two terminal and three terminal devices may be generated. A heatsink model may be included in the geometry (this can be important for conducted emissions simulations for power converters) plus additional dielectric volumes which may be used as output volumes in GGI_TLM or additional dielectric regions for example ceramic tiles placed between transistors and heatsinks.

Additional information to be included in the GGI_TLM input file can then be given e.g. required outputs, ngspice_timestep_factor, simulation time etc.

Finally the part of the Ngspice circuit file which describes all of the circuit elements to be linked into the GGI_TLM model are included.

Figure 4 shows a typical model, here a simple dc-dc converter, which may be readily built using the **GGI_TLM_create_PCB_simulation_model** software. The model includes a PCB tracks on a dielectric substrate, lumped components connected to the PCB pads on both sides of the board. The model also includes a heatsink.

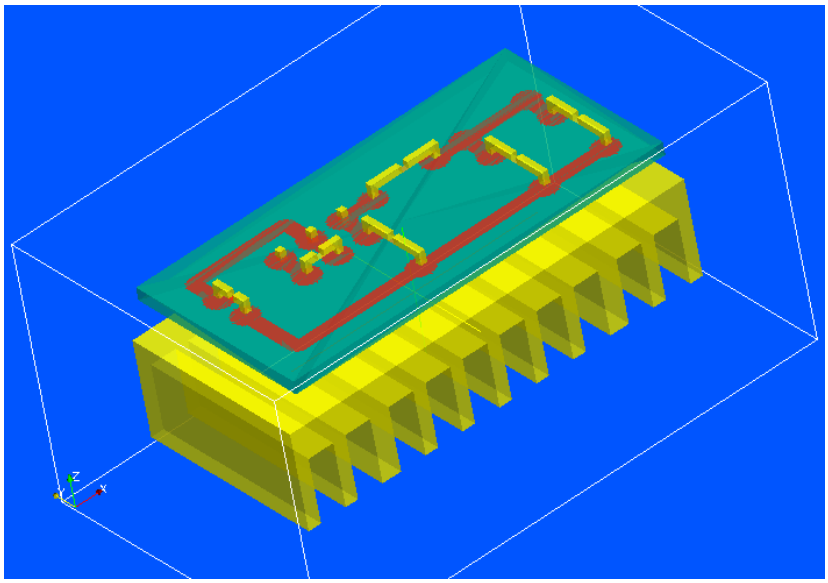

Figure 4 Full converter model

The components on the PCB are either one port (two terminal) components as seen in Figure 5 or two port (three terminal) components as seen in Figure 6. In Figure 5 the component position in the 3D GGI_TLM model is shown as the blue surface. This is connected to the PCB pads by conductors (which are represented by PEC surfaces in the GGI_TLM model) shown in yellow (this includes some inductance in the model which may need to be compensated for if models of packaged devices are being used). In Figure 6 the three conductors connecting the device model to the PCB pads are again shown in yellow and the two ports are shown by the blue and green surfaces.

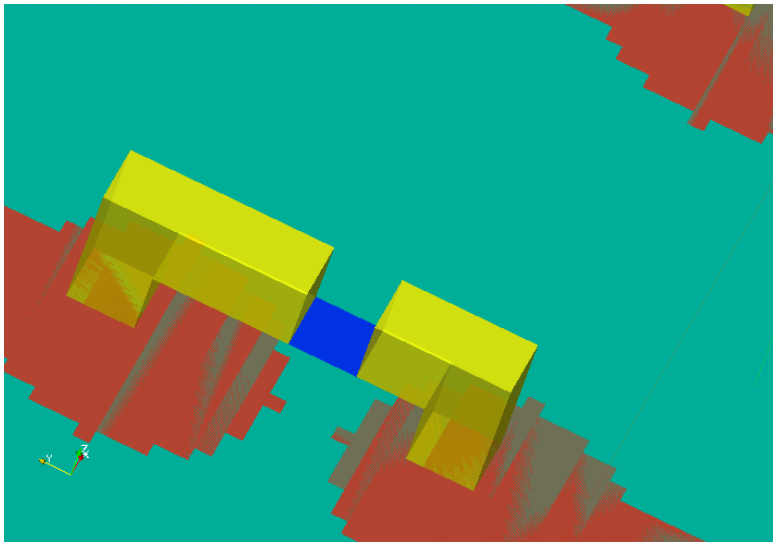

Figure 5 One port component model

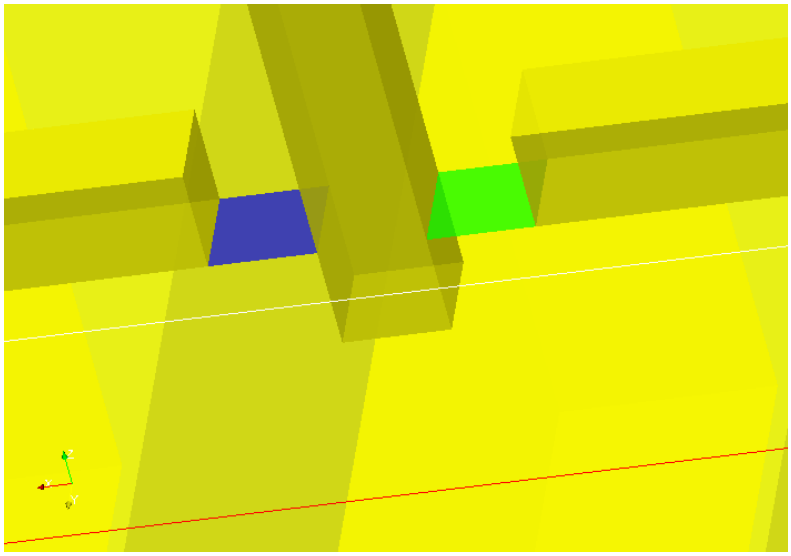

Figure 6 Two port component model

In the **GGI_TLM_create_PCB_simulation_model** software a one port model is specified using the port number, the Ngspice nodes corresponding to each of the two connections, the 3D coordinates of the PCB pads to which it is connected and the z position of the actual component surface in the GGI_TLM model. A package model (at the moment this consists of a dielectric volume) may also be added. The package model may also include a PEC surface on an outer face of the package which is connected to one of the terminals. The format of the one_port_model component information is shown below for a one port component with a rectangular package whose dielectric properties are defined in the material file **component_case.vmat**:

```
1 #component number
one_port_model
1 # number of ports
1 # port number
3 0 # Ngspice node numbers for connection 1 and connection 2
x1 y1 z1 # coordinates of connection 1
x2 y2 z2 # coordinates of connection 2
zc       # z position of component
rectangular # package type
xmin ymin zmin  xmax ymax zmax # min and max coordinates of package volume
component_case
1   # Number of PEC surfaces
zmin   2   # PEC surface and the terminal to electrically connect to
```

If no package model is to be included then the following package definition should be used:

```
none # package type
```

The software automatically creates a Spice model surface at the specified z position of the component, the x and y position of the surface is calculated as the average of the x and y coordinates of the two connections. The surfaces constituting the PEC connection wires are also generated automatically. Figure 7 shows the orientation of the component for the example of a diode where the Ngspice circuit file line is:

```
D1   node1    node2    diode_model
```

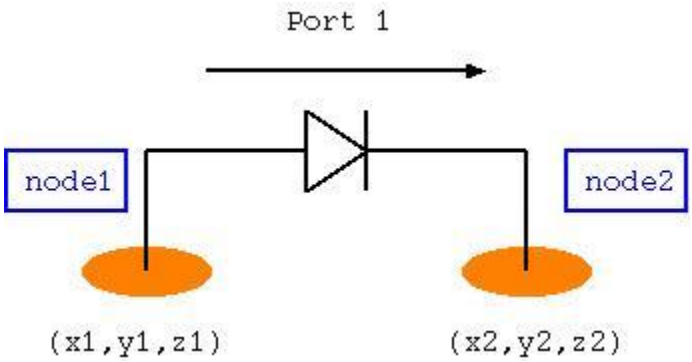

Figure 7 one port component orientation

A two port model has two ports and three electrical connections as seen in Figure 8. Each of the ports has two Ngspice nodes associated with it therefore there are four Ngspice nodes associated with the two port model. A two port model is specified using the two port numbers, the 4 Ngspice nodes corresponding to each of the two ports, the 3D coordinates of the PCB pads to which it is connected and the z position of the actual component surface in the GGI_TLM model. A package model (at the moment this consists of a dielectric volume) may also be added. As for the one port package model, the package may also include a PEC surface on an outer face of the package which is connected to one of the terminals. This could be used to include the capacitance of a transistor to ground via the transistor package heatsink plate to heatsink capacitance for example.

The format of the two_port_model component information is shown below for a two port component with a rectangular package whose dielectric properties are defined in the material file component_case.vmat:

```
1 #component number
two_port_model
2 # number of ports
1 2 # port numbers
3 0 # Ngspice node numbers for port 1 (connection 1 and connection 2)
0 4 # Ngspice node numbers for port 2 (connection 2 and connection 3)
x1 y1 z1 # coordinates of connection 1
x2 y2 z2 # coordinates of connection 2
x3 y3 z3 # coordinates of connection 3
zc       # z position of component
rectangular # package type
```

```
xmin ymin zmin  xmax ymax zmax # min and max coordinates of package volume
component_case
1          # Number of PEC surfaces
zmin   2   # PEC surface and the terminal to electrically connect to
```

If no package model is to be included then the following package definition should be used:

```
none # package type
```

The software automatically creates a Spice model surface at the specified z position of the component, the x and y position of the surface is calculated as the average of the x and y coordinates of the three connection points. The surfaces constituting the PEC connection wires are also generated automatically. Figure 8 shows the orientation of the component for the example of a diode where the Ngspice circuit file lines are:

```
D1   node1    node2    diode_model
V1   node3    node4    dc 1.0
```

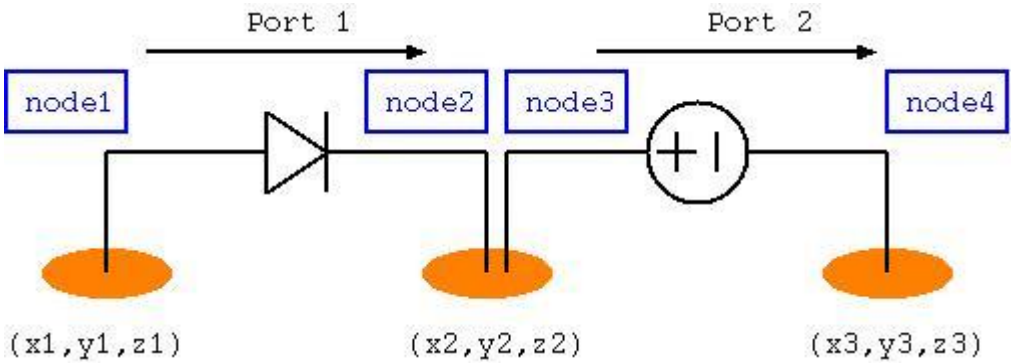

Figure 8 two port orientation

Great care must be taken to ensure the correct orientation of components such as diodes and voltage sources in which the correct orientation of the component is significant. This includes components on which initial conditions are applied for example inductor currents or capacitor voltages. Figure 9 illustrates the orientation of a selection of components in order to clarify the component orientation with respect to the node specification in Ngspice.

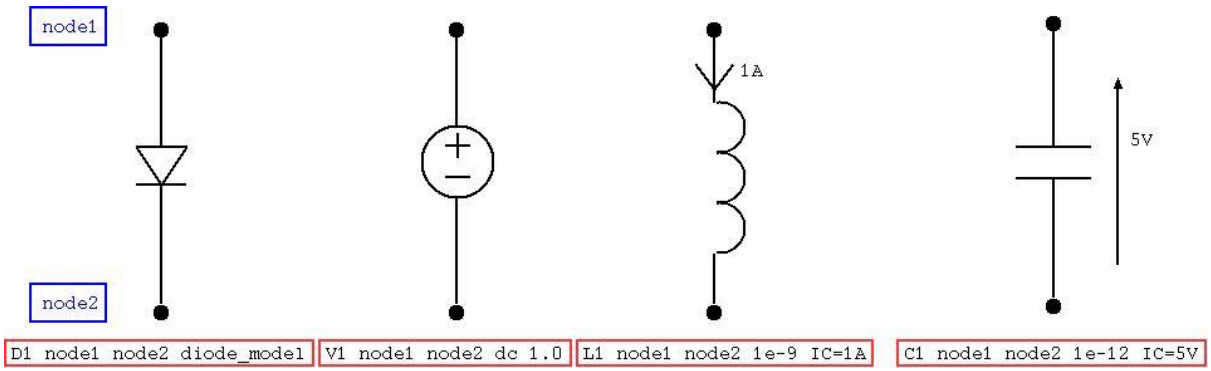

Figure 9 Component orientations with respect to node order, including initial conditions

Additional components may be specified in the input to the **GGI_TLM_create_PCB_simulation_model** software. Currently these additional components may be a heatsink model or a dielectric volume.

The heatsink model consists of a cuboid block with slots cut into in to form the vanes of the heatsink. The specification of a heatsink model requires the outer dimensions of the heatsink, the number of slots, the direction across the width of the slots, the direction in which the slots are cut plus the width and depth of the slots. The format of this information is shown below for the example of the heatsink shown in Figure 10:

```
heatsink
-0.002  0.0 -0.020    0.062 0.03 -0.005  # outer dimensions of the heatsink
10                                       # number of slots
y                                        # slot width direction
zmin                # slot depth direction (face from which slots are cut)
0.004               # width of slots
0.010               # depth of slots
```

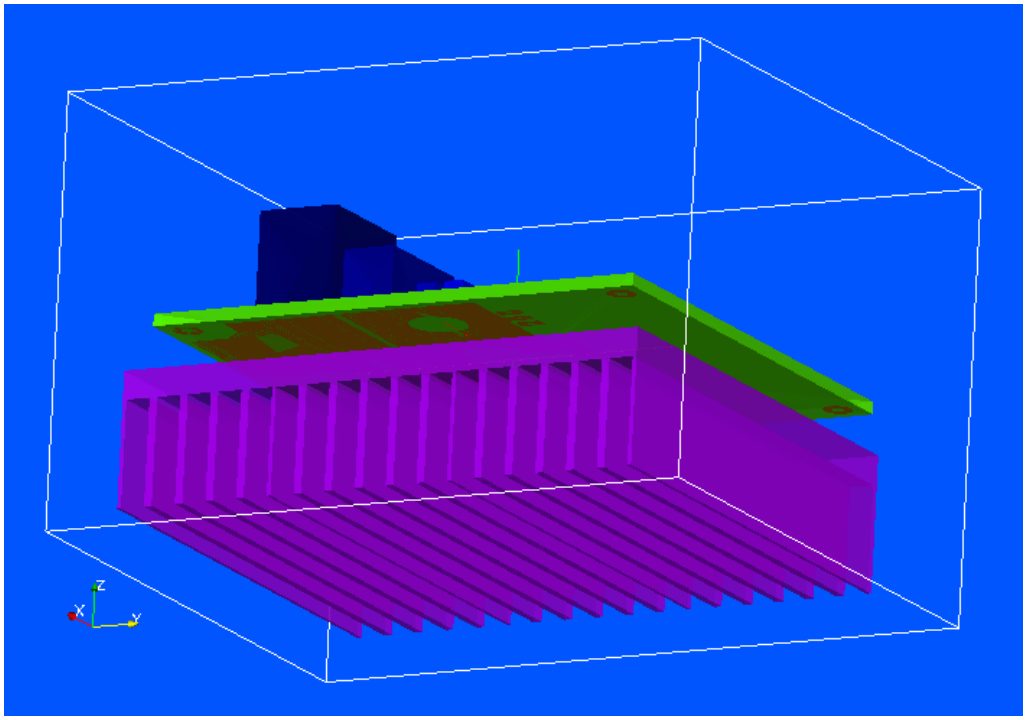

Figure 10 Converter model highlighting the heatsink model (purple).

A dielectric volume may also be specified as an additional component. This could be a ceramic tile used between a component package and a heatsink for example. An example of the format of this information is shown below:

```
dielectric
0.015  0.010  -0.005     0.026  0.020   -0.004 # outer dimensions of the
dielectric. Material name follows (without .vmat extension)
ceramic_tile
```

PCB Simulation Specification file format:

The format of the PCB simulation specification file format is shown in the table below alongside an example of each element of the file.

Format specification	**Data type**	**Example**
GGI_TLM input filename (*.inp) (string)	String	`circuit_test.inp`
dl: cell size for the TLM solution	real	`0.001`
TLM problem space dimensions xmin, ymin, zmin, xmax, ymax, zmax	6*real	`-0.05 -0.02 -0.03 0.05 0.02 0.03`
Number of gerber files to include	integer	`1`
For each gerber file:		
gerber file name	string	`two_track.gbr`
z position of layer	real	`0.003`
Number of dielectric layers	integer	`1`

For each dielectric layer:		
Rectangular dielectric region dimensions xmin, ymin, zmin, xmax, ymax, zmax	6*real	-0.04 -0.01 0.0 0.04 0.01 0.003
material filename (without .vmat extension)	string	FR4
Number of vias	integer	1
For each via:		
x and y coordinates then zmin and zmax of via	4*real	-0.03 0.005 0.0 0.003
Number of lumped components	integer	2
For a one port lumped component:		
component number (should be numbered in order)	integer	
component type ('one_port_model' or 'two_port_model')	string	one_port_model
Number of ports	integer	1
Port number(s)	integer	1
Ngspice node numbers for port	2*integer	1 0
connection point coordinates for connection 1	3*real	0.02 0.005 0.003
connection point coordinates for connection 2	3*real	0.03 0.005 0.003
z offset for component	real	0.005
Package type (none, rectangular)	string	rectangular
package parameters	rectangular: 6*real	-0.006 -0.003 0.0 0.006 0.003 0.008
For a two port lumped component:		
component number (should be numbered in order)	integer	
component type ('one_port_model' or 'two_port_model')	string	two_port_model
Number of ports	integer	2
Port number(s)	2*integer	2 3
Ngspice node numbers for port1	2*integer	1 0
Ngspice node numbers for port2	2*integer	2 0
connection point coordinates for connection 1 (port 1)	3*real	0.025 -0.005 0.003
connection point coordinates for connection 2 (reference of ports 1 and 2)	3*real	0.03 -0.005 0.003
connection point coordinates for connection 3 (port 2)	3*real	0.035 -0.005 0.003
z offset for component	real	0.005
Package type (none, rectangular)	string	rectangular
package parameters	rectangular: 6*real	-0.004 -0.003 0.0 0.006 0.003 0.008

material filename (without .vmat extension)	string	component_case
Number of PEC surfaces	integer	1
For each PEC surface on the package		
# PEC surface (xmin, ymin, zmin, xmax, ymax, zmax) and the terminal to electrically connect to	String integer	zmin 2
Number of additional components (heatsinks etc)	integer	1
For each additional component:		
additional component type	string	heatsink
outer dimensions of heatsink	6*real	-0.04 -0.01 -0.02 0.04 0.01 -0.001
Number of slots	integer	10
slot width direction (x, y or z)	character	x
slot depth direction (x, y or z)	character	z
width of slots	real	0.002
depth of slots	real	0.015
* START of GGI_TLM input file text *	string	* START of GGI_TLM input file text *
	n*string	ngspice_output_node_list
Multiple lines of GGI_TLM input file text may follow until it is ended with the line:		1 # Number of output nodes 1 # output node number 1 0 # Ngspice output node numbers
		Simulation_time 1e-8
* END of GGI_TLM input file text *	string	* END of GGI_TLM input file text *
* START of Ngspice input file text *	string	* START of ngspice input file text *
Multiple lines of Ngspice input file text may follow until it is ended with the line:	n*string	*GGI_TLM port 1 model
		Rs1 1001 1 100.0
		Vs1 1001 0 exp(0 1 1e-10 1e-9 1e-10)
		* GGI_TLM port 2 model
		D1 0 2 Dsch
		.MODEL Dsch D(IS=0.0002)
* END of Ngspice input file text *	string	* END of ngspice input file text *

Known problems

There is a known issue regarding memory usage with the GGI_TLM/Ngspice linked software/ Ngpsice appears to store node voltages at all the previous time-steps of the simulation thus as a simulation proceeds, Ngspice uses more and more memory. This is illustrated in Figure 11 which shows the memory usage for the GGI_TLM_SEQ process as a function of time for a dc-dc converter model. For some simulations with very large numbers of nodes and many timesteps (of order of millions) this may cause problems if the whole of the memory is used before the simulation finishes.

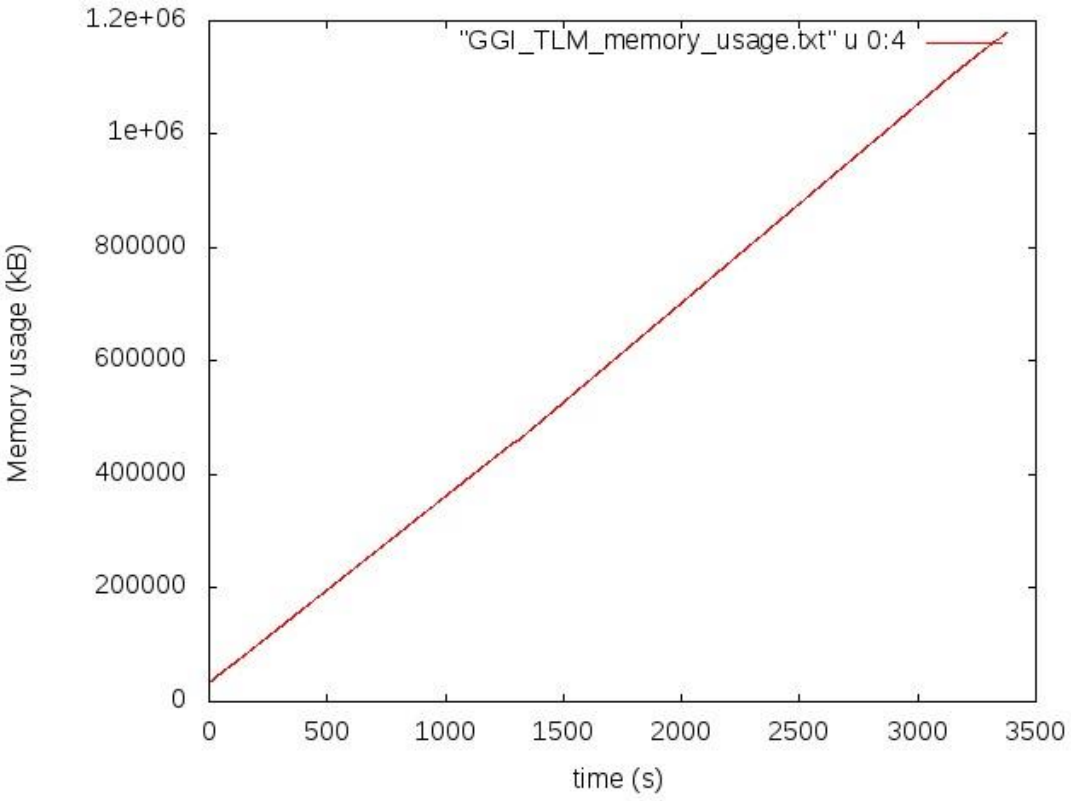

Figure 11. Memory usage as a function of time for a GGI_TLM – Ngspice simulation.

The impact of this 'memory leak' may be reduced by using the Ngspice 'SAVE' command to tell Ngspice to only save data for certain nodes in the circuit (i.e. those used in the GGI_TLM interface and those required for output).

The way in which a simulation is approached may also be used to reduce the required simulation time. An initial circuit simulation in Ngspice to allow initial slow transient circuit effects to reach their steady state may be performed and then the individual component voltages (for a capacitor) or currents (for an inductor) used as initial conditions in the GGI_TLM – Ngspice coupled solution.

Post processing conducted emissions data

Post processing procedures have been implemented in order to calculate conducted emissions data from raw time domain voltage data. These processes are applicable to

both simulated and measured data. The post processing software allows a number of processes to be applied to the data in both time and frequency domains to allow investigation of different data analysis procedures and to allow comparison with standards. The software allows the following processes to be applied to the raw time domain data:

- Scale the voltage data (e.g. to micro volts) as required
- Apply a low pass filter to the time domain data
- Apply a high pass filter to the time domain data
- Extract a series of sub-samples from the dataset i.e. divide a long period of data into a number of shorter sub-samples distributed through the time period of the input data. The sub-samples may be sampled at a different sample rate to the initial dataset. The sub-sampled data may be padded with zeros or a periodic extension of the dataset added.
- A window function may be applied to each of the sub-sampled datasets.
- The d.c. component of the signal can be subtracted
- The FFT of each of the sub-samples may be calculated to give frequency domain sub-sample data
- The frequency domain sub-sample functions may be averaged in the frequency domain
- The response of a receiver with a specified detector bandwidth may be calculated where the receiver detector frequency response may be described by a gaussian or a rectangular frequency domain response

This processes which allows application of all the above options has been incorporated into the **GGI_TLM_post_process** software, option 58. The software is run with the command

GGI_TLM_post_process

The available post processing options are then presented. The time domain conducted emissions processing option is selected by inputting **58** at the prompt.

The filename for the time domain data must then be supplied by the user. This file should be an ascii (text) file with columns for time and voltage. There may be header lines at the top of the file but following this the data should be continuous. It is assumed that the voltage data is sampled at a uniform sample rate. This assumption is valid for time domain oscilloscope data and for GGI_TLM simulation data however it may not be valid for raw Spice time domain output due to the variable time stepping algorithm used in Spice [3]. In this case the dataset should be resampled to give a dataset which is uniformly sampled in time. The **GGI_TLM_post_process** option 55 (interpolate function) may be used to achieve this.

The user is requested to supply the number of header lines in the file to ignore and then the column for the time data and then the column for the voltage data.

The first process which can be applied is scaling the voltage data. For example if the data is to be presented as dBμV then a scaling factor of '1E-6' should be given. For no scaling, enter '1'.

Once the time domain data has been read and scaled appropriately, low pass and/or high pass filters may be applied. These filters may be used to prevent aliasing if the data is subsequently sub-sampled at a lower data rate for example.

The low and high pass filter functions are Butterworth filters. The filter order is chosen to be between 1 and 4 and the cutoff frequency, f_c, must be specified. The low pass filters of order 1 to 4 are defined using the following functions of s where $s = \frac{j\omega}{w_c}$, $w_c = 2\pi f_c$

$$H_1(\omega) = \frac{1}{1+s}$$

$$H_2(\omega) = \frac{1}{1+1.4142s+s^2}$$

$$H_3(\omega) = \frac{1}{(1+s)(1+s+s^2)}$$

$$H_4(\omega) = \frac{1}{(1+0.7654s+s^2)(1+1.8478s+s^2)}$$

The high pass filter functions are obtained using the same formulae with $s = \frac{w_c}{j\omega}$, $w_c = 2\pi f_c$

The user is asked to enter the number of filters which are required (this may be 0).

For each filter applied the user must enter the type of filter to apply ('LPF' or 'HPF') then the filter order (integer between 1 and 4, inclusive) and the filter cutoff frequency (Hz).

Following the filtering process, the dataset may be re-sampled in order to obtain a number of sub-samples over shorter timespans. The sample rate may also be changed (reduced).

The user is required to specify the number of sub-segments of data required, the time period of each sub-segment, the number of samples in each sub-segment (this should be a power of 2 due to FFT implementation), and the total time period over which the sub-samples are to be distributed (or enter '0' for continuous sub-samples). These parameters are explained in Figure 12 below. Note that the sampling times at this stage will probably be different to the sampling times of the input data. In the re-sampling of the input dataset a linear interpolation is used to calculate the voltage value between input time samples.

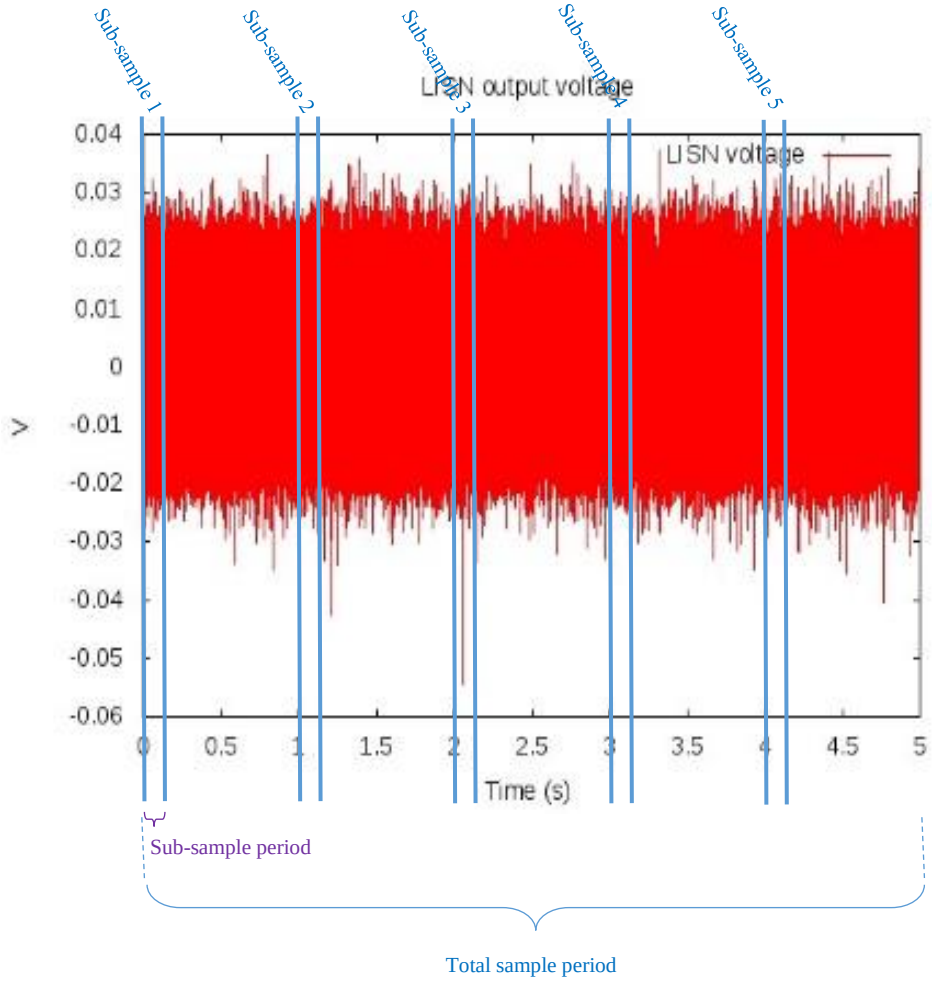

Figure 12. Sub-sampling the input dataset: 5 sub-samples, each of period 0.25s distributed over 5s

The sub-sampled data may be padded with zeros or alternatively a periodic extension of the dataset may be added such that the total number of samples is a power of 2. This has the effect of providing frequency samples from the FFT at a finer resolution that can be obtained with the initial sub-sampled dataset. The user must enter the padding factor (enter '1' for no padding) and then either 'z' to pad with zeros or 'p' to add a periodic extension of the data. Note that the effect of zero padding on the final frequency domain output will be compensated for such that the same frequency domain amplitude will be produced whatever the extent of the zero padding.

After sub-sampling and padding the data, a time domain window function may be applied. The window function which may be applied to the time domain data can be either a rectangular window or a Hann window. The Hann window is given by the function

$$W(i) = \frac{1}{2}\left(1 - cos\left(\frac{2\pi i}{N}\right)\right) \quad 0 \leq i \leq N$$

The Hann window is shown in Figure 13.

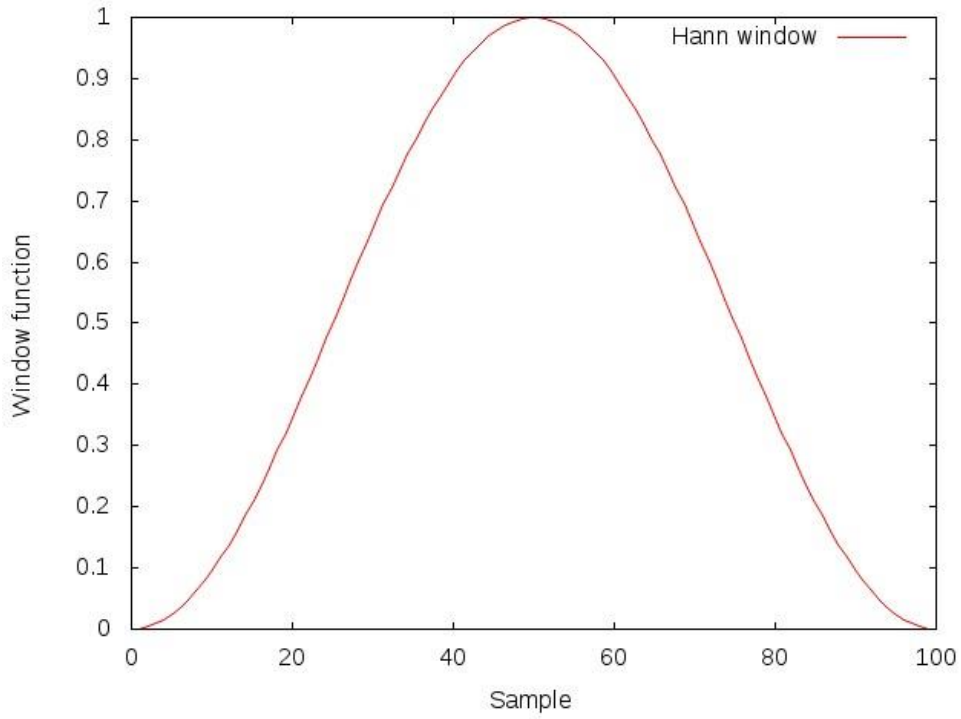

Figure 13. Hann Window function

The user is requested to enter the window type and must respond with either 'r' for a rectangular window (effectively no change to the data as the window covers the full period of the data) or 'h' to apply the Hann window as described above.

Following windowing, the user is asked whether to subtract the d.c. component of the signal. The user must enter 'y' or 'n' to this question.

Each of the sub-samples is transformed into the frequency domain via the FFT algorithm. The magnitude of these frequency domain functions is then averaged. This raw frequency domain data is then written to a file (filename to be provided by the user.)

The format of the raw frequency domain data output is a text file with six columns as follows:

Frequency (Hz), Re{V(f)}, Im{V(f)}, Abs{V(f)}, VdBm(50Ω load), VdB

Note that the real part is the average of the magnitudes and the imaginary part is zero. The data is only written for positive frequencies however the VdBm and VdB data compensates for this by the addition of 3.01dB so it can be compared to measured data etc.

In order to provide frequency domain data which can be compared with that measured using a spectrum analyser or for comparison with emissions limits from standards, frequency domain data with a specified receiver (detector) bandwidth can be produced. Different receiver bandwidths can be specified over different frequency ranges if required (see Post processing example 1 below for an example). The user must specify the number of frequency bands with different receiver bandwidths, the filename for the output and for each band:

Minimum frequency of the band (Hz)

Maximum frequency of the band (Hz)

Number of frequencies to output

'r' for a rectangular receiver transfer function or 'g' for a Gaussian receiver transfer function

The detector bandwidth (Hz)

The detector transfer functions are shown in Figure 14 below.

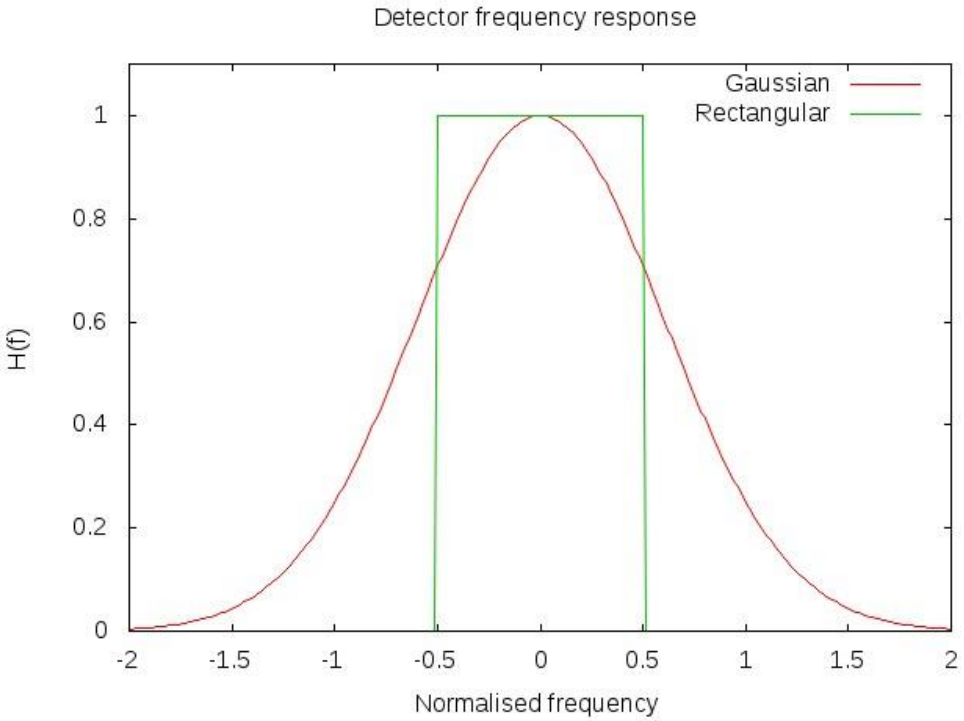

Figure 14. Detector transfer functions.

The format of the frequency domain data output is a text file with three columns as follows:

Frequency (Hz), P_detector, VdBm(50Ω load), VdB

Note that P_detector is the total power in the detector bandwidth at each frequency and that negative frequency data is taken into account in the output.

Post processing test cases

A number of test cases have been produced which exercise the conducted emissions data post processing. These are available in the GGI_TLM directory **GGI_TLM/Test_DATA/POST_PROCESS_CONDUCTED_EMISSIONS/**

The test cases are designed to show the effects of different post processing strategies based on a periodic signal consisting of the first 10 (odd) harmonics of a square wave Fourier series expansion. One period of the waveform is shown in Figure 15. This test set allows the results of different sampling and post processing strategies to be investigated and the results compared with the analytic result for the known input.

In the frequency domain results the raw frequency domain data is shown in red (V.fft), the frequency domain data with a specific detector bandwidth applied are shown in green (V.favg) and the analytic result for the known input signal which is shown as blue stars.

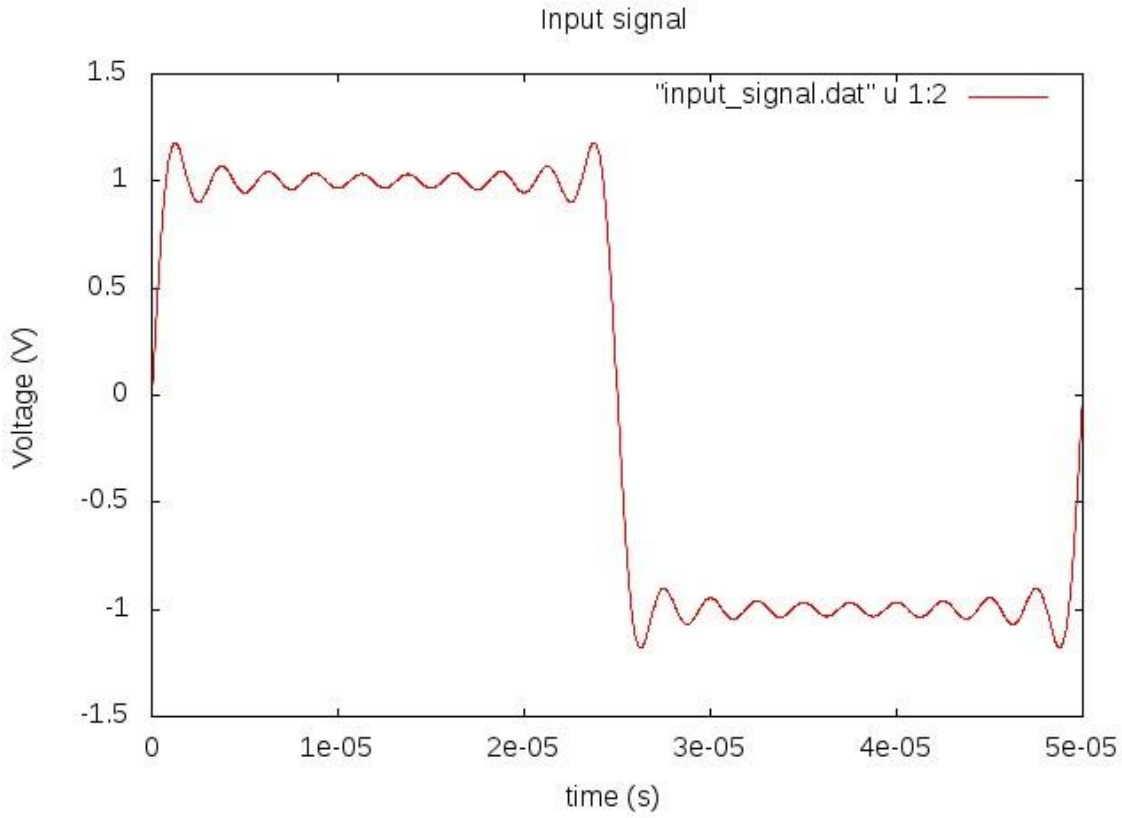

Figure 15. Post processing test 1; time domain voltage.

Test 1:

The input data for test 1 consists of a single cycle of 10 harmonics from a square wave expansion with a fundamental frequency of 20kHz. The dataset consists of 2048 samples (i.e. a power of 2) with the timestep=2.44e-8s. This is a reference dataset which requires no re-sampling or filtering.
The post processing options are shown in the table below and the result is shown in Figure 16. Clearly the post processed dataset reproduces the analytic reference result.

```
          58  POST PROCESSING OPTION: POST PROCESS TIME DOMAIN CONDUCTED EMISSIONS DATA
input_signal.dat
           0   # number of header lines in the file
           1   # time data column
           2   # Voltage data column
   1000000.0000000000        # scale factor for data (eg 1E6 for V to micro V
           0   # number of filters to apply
           1   # Number of sub-data segments of data to process
   5.0000000000000002E-005   # period of each sub-segment
        2048   # number of samples in each sub-segment (this should be a power of 2)
   0.0000000000000000        # time period over which the samples are to be distributed or
enter 0 for continuous subsets
           1   # pad factor (set to 1 for no padding, pad factor should be a power of two)
z # z to pad with zeros or p to create a periodic continuation of the sub-signal
           1   # number of sub-segments to plot (max 9)
r  # time domain window function (r=rectangular, h=Hann)
y  # subtract d.c. (y or n)
V.fft
           2  # number of frequency domain bands with specific detector bandwidths to output
V.favg
   10.000000000000000       # minimum frequency for output
   250000.00000000000       # maximum frequency for output
        1250  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   200.00000000000000       # detector bandwidth
   250000.00000000000       # minimum frequency for output
   2000000.0000000000       # maximum frequency for output
       14875  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   2000.0000000000000       # detector bandwidth
           0  POST PROCESSING OPTION: QUIT
```

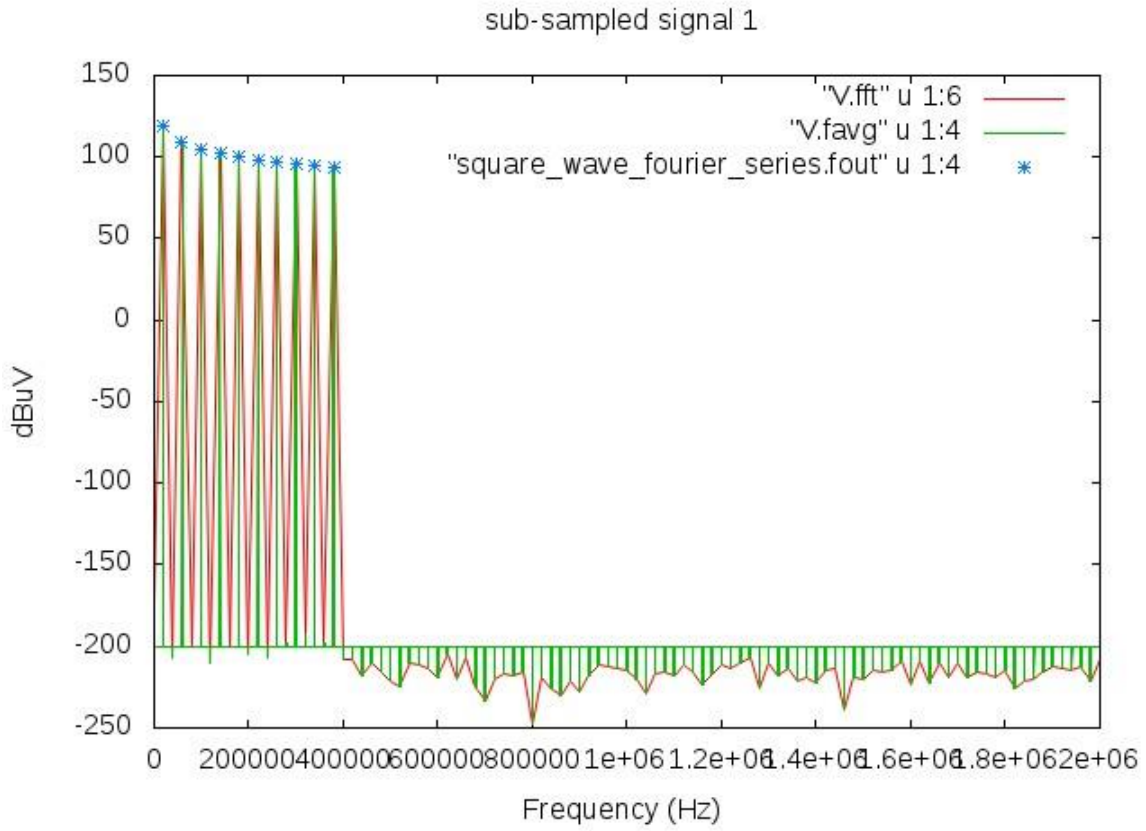

Figure 16. Frequency domain result

Test 1B

The input data for test 1b consists of a single cycle of 10 harmonics from a square wave expansion with a fundamental frequency of 20kHz. The dataset consists of 2000 samples (i.e. NOT a power of 2) with the timestep=2.5e-8s. This is a dataset which requires re-sampling before the FFT is applied.
The post processing options are shown in the table below and the result is shown in Figure 17. Clearly the post processed dataset reproduces the analytic reference result reasonably well but the resampling adds some noise to the frequency domain output however this is of order 80dB below the analytic result.

```
          58  POST PROCESSING OPTION: POST PROCESS TIME DOMAIN CONDUCTED EMISSIONS DATA
input_signal.dat
           0   # number of header lines in the file
           1   # time data column
           2   # Voltage data column
   1000000.0000000000        # scale factor for data (eg 1E6 for V to micro V
           0   # number of filters to apply
           1   # Number of sub-data segments of data to process
   5.0000000000000002E-005   # period of each sub-segment
        2048   # number of samples in each sub-segment (this should be a power of 2)
   0.0000000000000000        # time period over which the samples are to be distributed or
enter 0 for continuous subsets
           1   # pad factor (set to 1 for no padding, pad factor should be a power of two)
z # z to pad with zeros or p to create a periodic continuation of the sub-signal
           1   # number of sub-segments to plot (max 9)
r  # time domain window function (r=rectangular, h=Hann)
y  # subtract d.c. (y or n)
V.fft
           2  # number of frequency domain bands with specific detector bandwidths to output
V.favg
   10.000000000000000       # minimum frequency for output
   250000.00000000000       # maximum frequency for output
```

```
        1250  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   200.00000000000000       # detector bandwidth
   250000.00000000000       # minimum frequency for output
   2000000.0000000000       # maximum frequency for output
       14875  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   2000.0000000000000       # detector bandwidth
          0  POST PROCESSING OPTION: QUIT
```

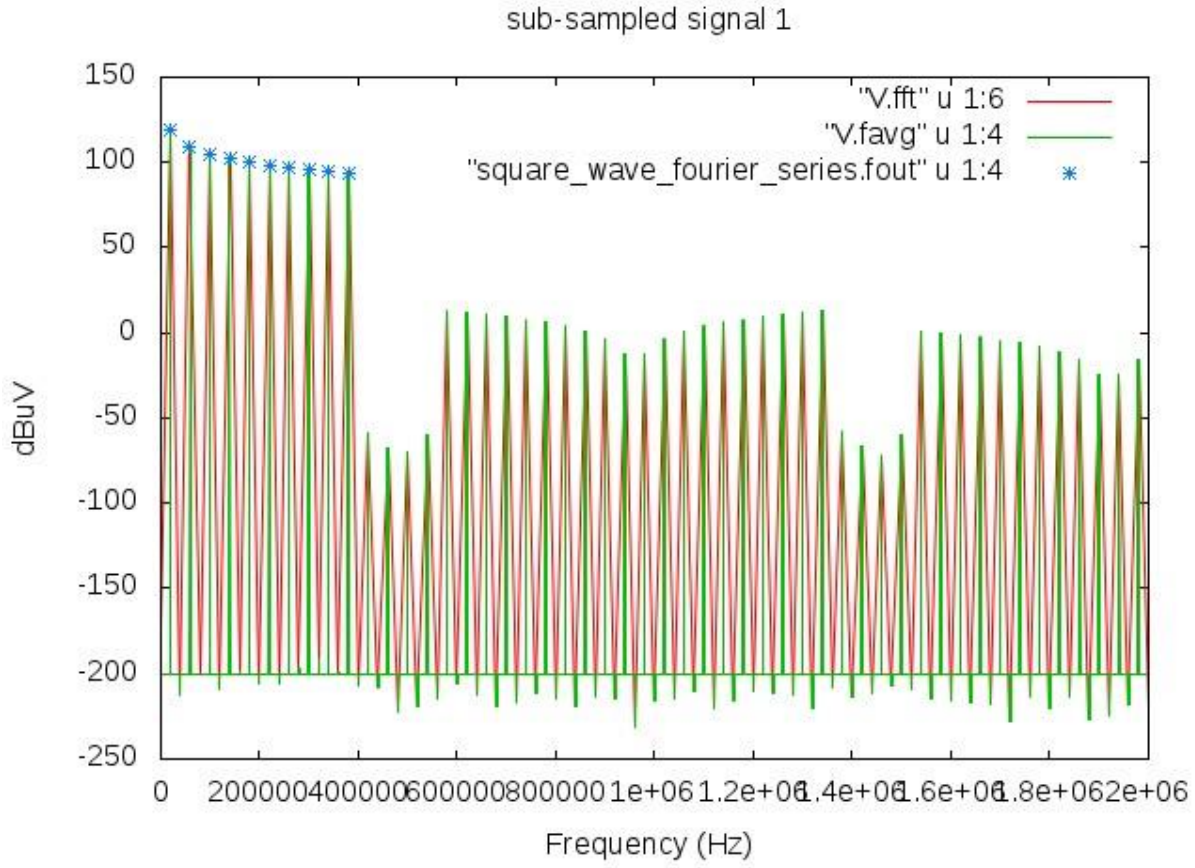

Figure 17. Frequency domain output showing the effect of resampling the input signal.

Test 2

The input data for test 2 consists of 1000 cycles of 10 harmonics from a square wave expansion with a fundamental frequency of 20kHz. The dataset consists of 200 samples per cycle (i.e. NOT a power of 2). In the post processing 100 cycles of the dataset are extracted and resampled to give 262144 samples (Figure 18) before transforming to the frequency domain.
The post processing options are shown in the table below and the result is shown in Figure 19. Clearly the post processed dataset reproduces the analytic reference result reasonably well but the resampling adds some noise to the frequency domain output however this is of order 80dB below the analytic result. The raw FFT dataset has a finer frequency resolution than test 1 due to the longer input time period. The frequency domain data with the detector bandwidth of 2kHz shows a higher 'noise' level than the raw FFT data due to the increased effective receiver bandwidth.

```
          58  POST PROCESSING OPTION: POST PROCESS TIME DOMAIN CONDUCTED EMISSIONS DATA
input_signal.dat
           0   # number of header lines in the file
           1   # time data column
           2   # Voltage data column
   1000000.0000000000        # scale factor for data (eg 1E6 for V to micro V
           0   # number of filters to apply
           1   # Number of sub-data segments of data to process
   5.0000000000000001E-003   # period of each sub-segment
      262144   # number of samples in each sub-segment (this should be a power of 2)
   0.0000000000000000        # time period over which the samples are to be distributed or
enter 0 for continuous subsets
           1   # pad factor (set to 1 for no padding, pad factor should be a power of two)
z # z to pad with zeros or p to create a periodic continuation of the sub-signal
           1   # number of sub-segments to plot (max 9)
r  # time domain window function (r=rectangular, h=Hann)
y  # subtract d.c. (y or n)
V.fft
           2  # number of frequency domain bands with specific detector bandwidths to output
V.favg
   10.000000000000000       # minimum frequency for output
   250000.00000000000       # maximum frequency for output
        1250  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   200.00000000000000       # detector bandwidth
   250000.00000000000       # minimum frequency for output
   2000000.0000000000       # maximum frequency for output
       14875  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   2000.0000000000000       # detector bandwidth
           0  POST PROCESSING OPTION: QUIT
```

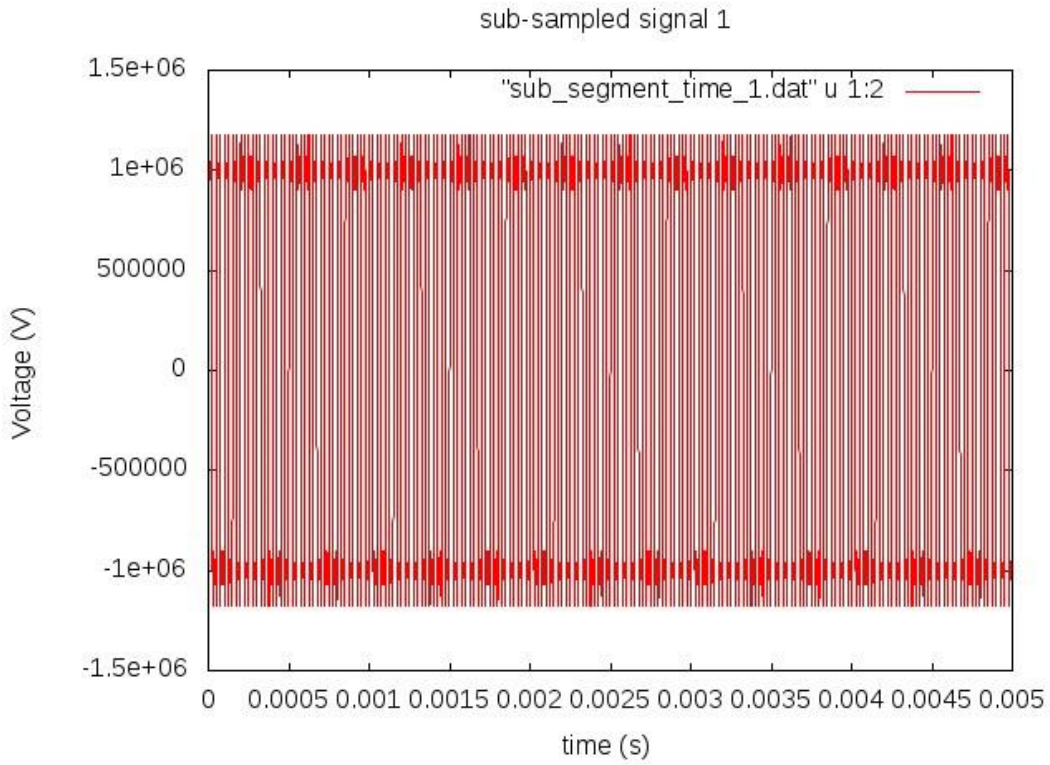

Figure 18. 100 cycles of 10 harmonic 'square wave'

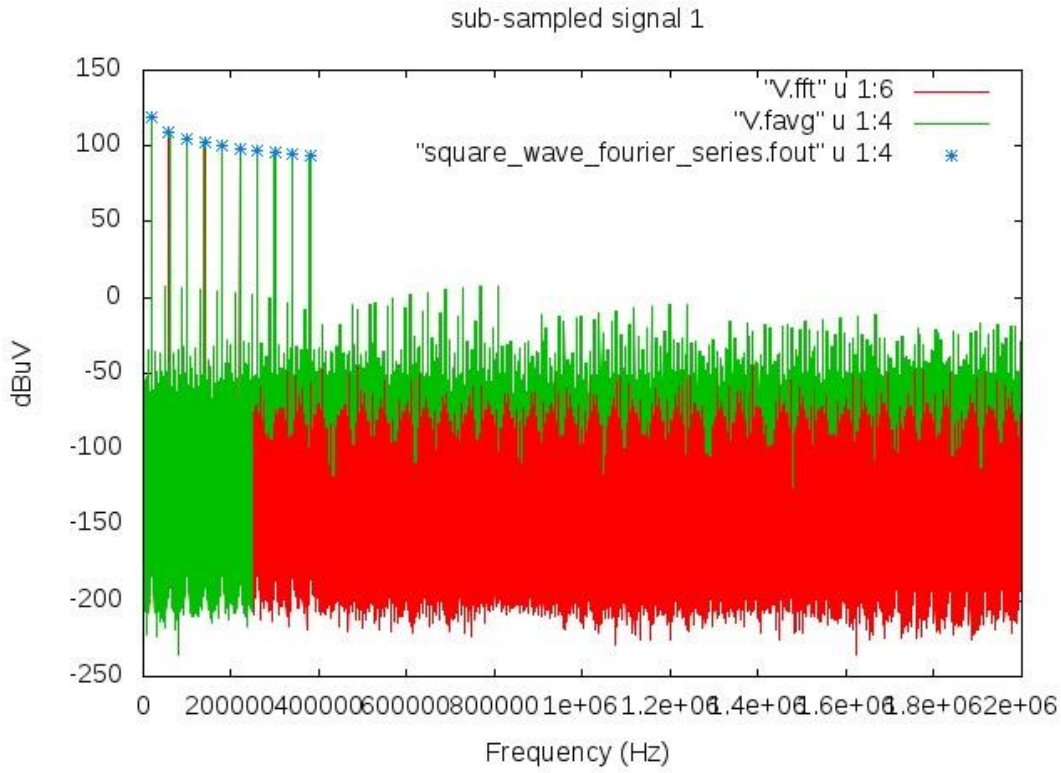

Figure 19 Frequency domain result, test 2.

Test 3

Test 3 shows the application of filters to the input data before re-sampling. The input dataset consists of 1000 cycles of 10 harmonics from a square wave expansion (Figure 20), 20kHz plus a 50Hz signal. The post processing options shown below illustrate the use extracting 100cycles of the 20kHz signal and application of a high pass filter at 100Hz to reduce influence of the 50Hz signal on the resulting signal (Figure 21). In addition, this test shows how two different detector bandwidths can be used over different frequency ranges to produce the frequency domain result as seen in Figure 22 where the higher bandwidth detector applied above 250kHz gives an increased 'noise' level in the result.

```
          58  POST PROCESSING OPTION: POST PROCESS TIME DOMAIN CONDUCTED EMISSIONS DATA
input_signal.dat
           0   # number of header lines in the file
           1   # time data column
           2   # Voltage data column
   1000000.0000000000        # scale factor for data (eg 1E6 for V to micro V
           1   # number of filters to apply
H  # filter type (LPF or HPF)
           4   # filter order
   100.00000000000000        # filter cutoff frequency
           1   # Number of sub-data segments of data to process
   5.0000000000000001E-003   # period of each sub-segment
      262144   # number of samples in each sub-segment (this should be a power of 2)
   0.0000000000000000        # time period over which the samples are to be distributed or
enter 0 for continuous subsets
           1   # pad factor (set to 1 for no padding, pad factor should be a power of two)
z # z to pad with zeros or p to create a periodic continuation of the sub-signal
           1   # number of sub-segments to plot (max 9)
r  # time domain window function (r=rectangular, h=Hann)
y  # subtract d.c. (y or n)
V.fft
           2  # number of frequency domain bands with specific detector bandwidths to output
V.favg
   10.000000000000000       # minimum frequency for output
   250000.00000000000       # maximum frequency for output
        1250  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   200.00000000000000       # detector bandwidth
   250000.00000000000       # minimum frequency for output
   2000000.0000000000       # maximum frequency for output
       14875  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   2000.0000000000000       # detector bandwidth
           0  POST PROCESSING OPTION: QUIT
```

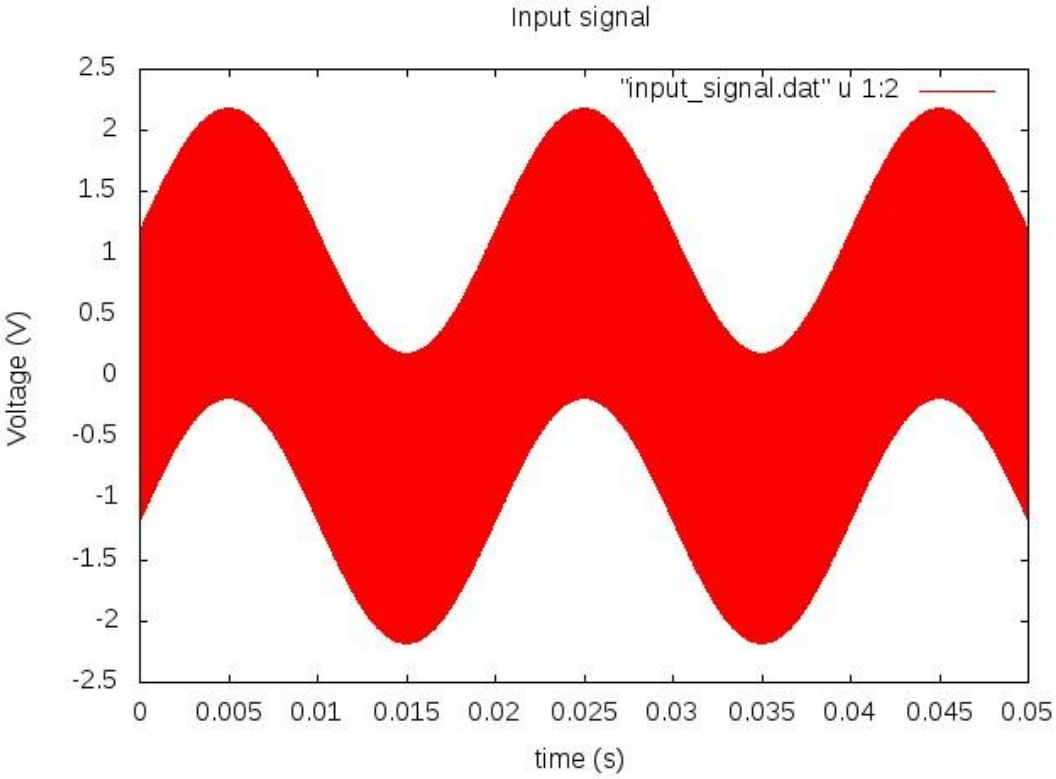

Figure 20. 20kHz 'suqare wave' with 50Hz signal added.

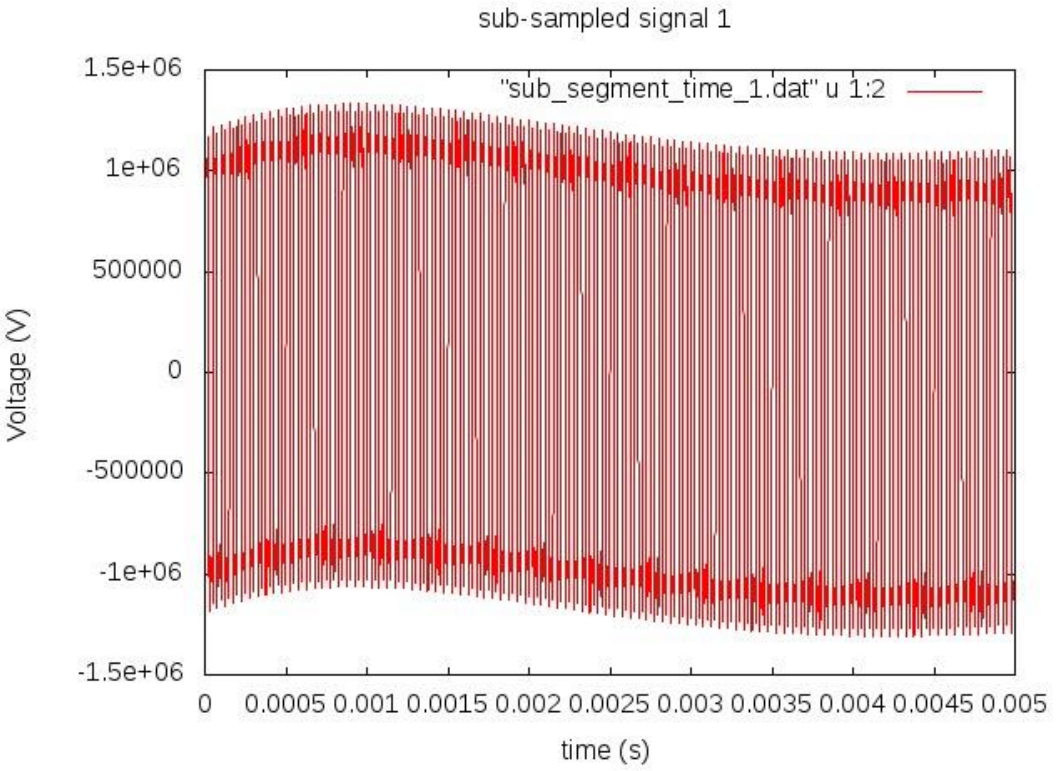

Figure 21. Sub-sampled dataset

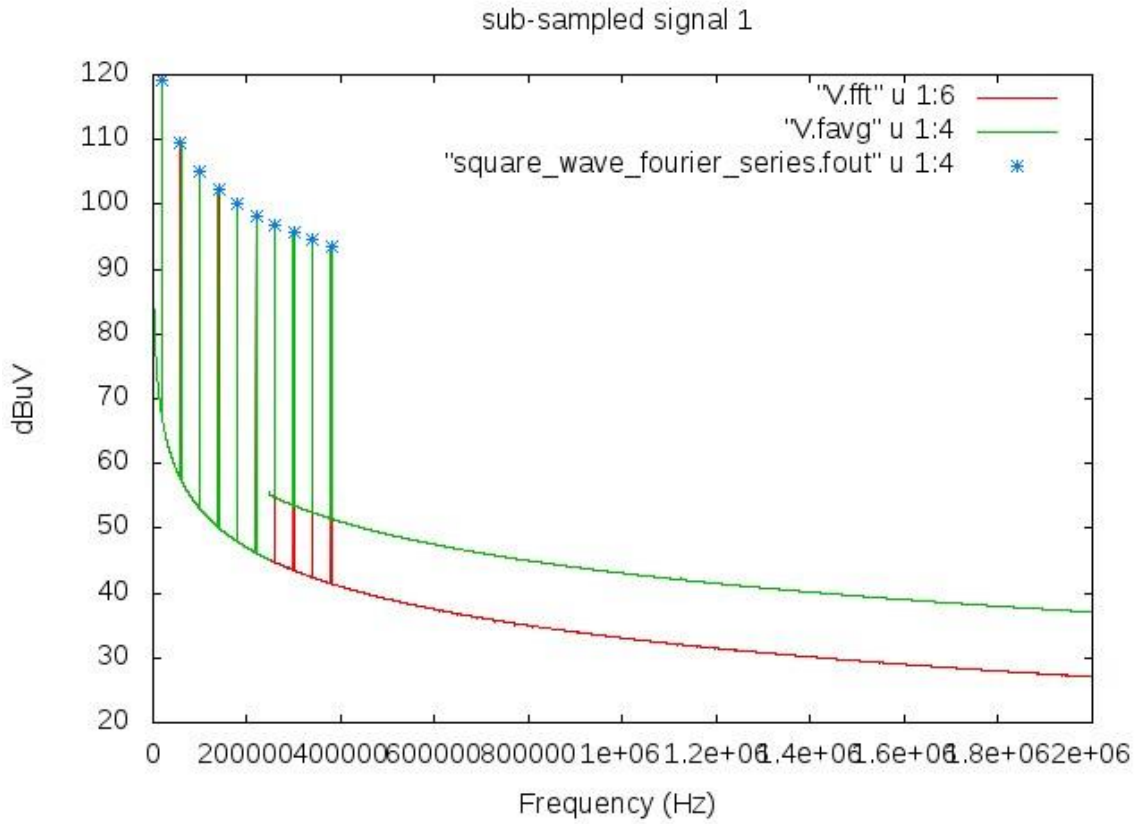

Figure 22. Frequency domain result, test 3.

Test 4

In this test a single cycle of the 10 harmonic square wave is padded with zeros to increase the signal period 128 times. This should increase the frequency resolution by a factor of 128. The post processing automatically scales the raw frequency domain output to compensate the levels for the zero padding so the raw frequency domain data in Figure 23 will agree with the analytic result at the harmonics of the input signal fundamental frequency. There is a problem with the application of a detector with increased bandwidth in this case as seen in the red curve in Figure 23. This curve is not scaled to take account of the zero padding and it is not clear how to do this. For this reason zero padding is not recommended. Increased frequency resolution can be obtained by creating a periodic extension of a single cycle as seen in test 5 below.

```
          58  POST PROCESSING OPTION: POST PROCESS TIME DOMAIN CONDUCTED EMISSIONS DATA
input_signal.dat
           0   # number of header lines in the file
           1   # time data column
           2   # Voltage data column
   1000000.0000000000        # scale factor for data (eg 1E6 for V to micro V
           0   # number of filters to apply
           1   # Number of sub-data segments of data to process
   5.0000000000000002E-005   # period of each sub-segment
        2048   # number of samples in each sub-segment (this should be a power of 2)
   0.0000000000000000        # time period over which the samples are to be distributed or
enter 0 for continuous subsets
         128   # pad factor (set to 1 for no padding, pad factor should be a power of two)
z # z to pad with zeros or p to create a periodic continuation of the sub-signal
           1   # number of sub-segments to plot (max 9)
r  # time domain window function (r=rectangular, h=Hann)
y  # subtract d.c. (y or n)
V.fft
           2  # number of frequency domain bands with specific detector bandwidths to output
V.favg
   10.000000000000000       # minimum frequency for output
   250000.00000000000       # maximum frequency for output
        1250  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   200.00000000000000       # detector bandwidth
   250000.00000000000       # minimum frequency for output
   2000000.0000000000       # maximum frequency for output
       14875  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   2000.0000000000000       # detector bandwidth
           0  POST PROCESSING OPTION: QUIT
```

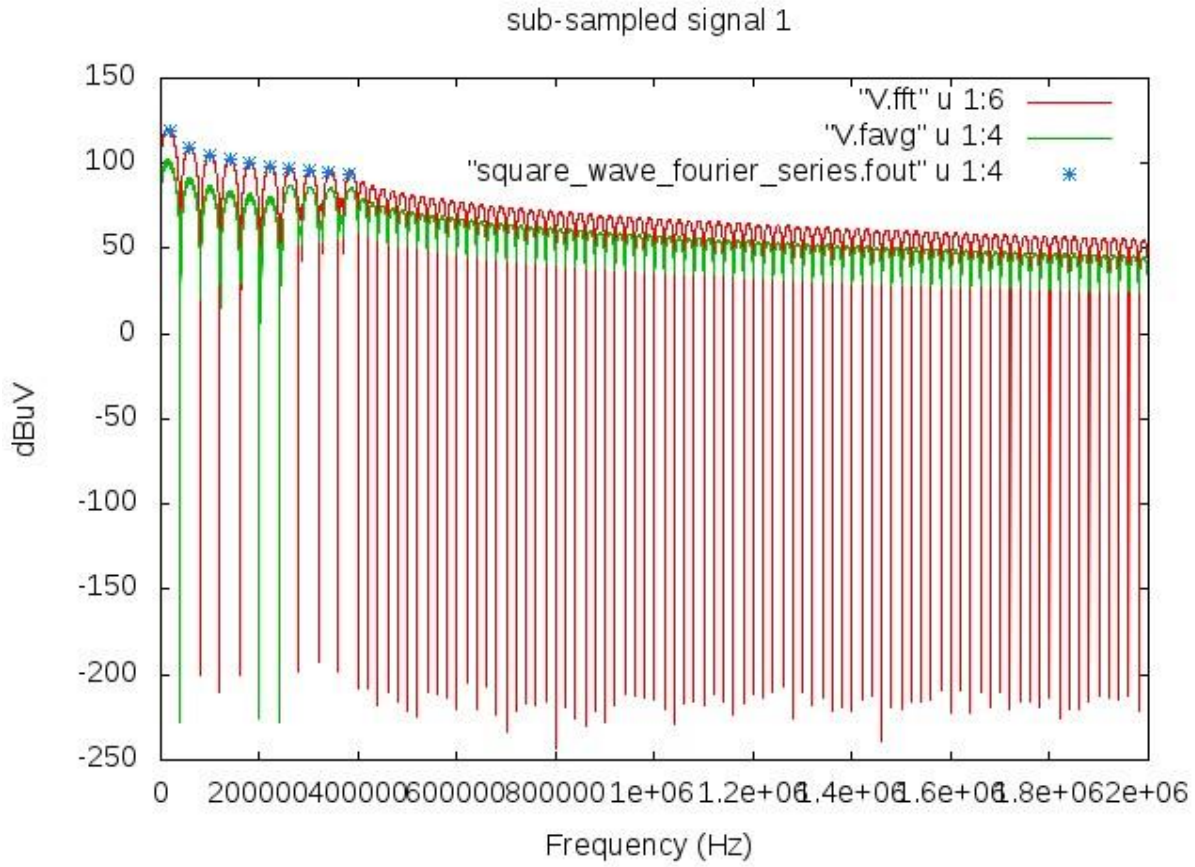

Figure 23. Frequency domain result, test 4.

Test 5

In this test a single cycle of the 10 harmonic square wave is periodically extended to increase the signal period 128 times. This should increase the frequency resolution by a factor of 128. The result in Figure 24 shows the expected results although with finer frequency resolution than obtained by processing a single cycle of the input data. It should be noted that adding a periodic extension does not add any information, it is only useful for example when only a single cycle of the operation of a converter is simulated in GGI_TLM but a large time period of data is required for processing according to a standard, or for comparison with measurement.

```
          58  POST PROCESSING OPTION: POST PROCESS TIME DOMAIN CONDUCTED EMISSIONS DATA
input_signal.dat
           0   # number of header lines in the file
           1   # time data column
           2   # Voltage data column
   1000000.0000000000        # scale factor for data (eg 1E6 for V to micro V
           0   # number of filters to apply
           1   # Number of sub-data segments of data to process
   5.0000000000000002E-005   # period of each sub-segment
        2048   # number of samples in each sub-segment (this should be a power of 2)
   0.0000000000000000        # time period over which the samples are to be distributed or
enter 0 for continuous subsets
         128   # pad factor (set to 1 for no padding, pad factor should be a power of two)
p # z to pad with zeros or p to create a periodic continuation of the sub-signal
           1   # number of sub-segments to plot (max 9)
r  # time domain window function (r=rectangular, h=Hann)
y  # subtract d.c. (y or n)
V.fft
           2  # number of frequency domain bands with specific detector bandwidths to output
V.favg
   10.000000000000000       # minimum frequency for output
   250000.00000000000       # maximum frequency for output
        1250  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   200.00000000000000       # detector bandwidth
   250000.00000000000       # minimum frequency for output
   2000000.0000000000       # maximum frequency for output
       14875  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   2000.0000000000000       # detector bandwidth
           0  POST PROCESSING OPTION: QUIT
```

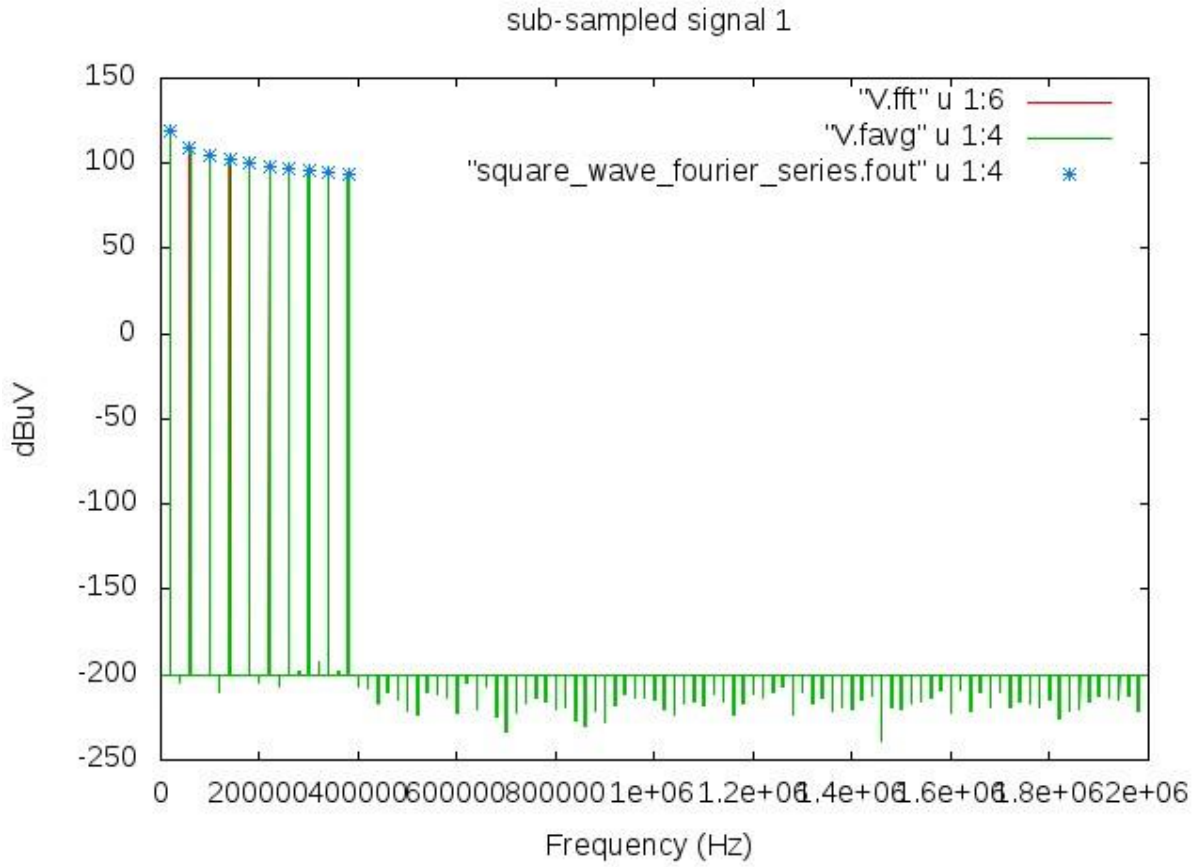

Figure 24. Frequency domain result for test 5

Test 6

Test 6 shows the effect of changing the bandwidth in different frequency bands. The input signal consists of 100 cycles of the 10 harmonic square wave with some added Gaussian noise Figure 25. The post processing options are shown below where a bandwidth of 200Hz is specified from 200Hz to 250kHz and 2kHz bandwidth from 250kHz to 2GHz. Figure 26 shows the frequency domain result. The effect of the change in bandwidth is seen in the step in the noise level at 250kHz in the green curve. Note that the levels of the square wave harmonics are not changed by the bandwidth increase.

```
          58  POST PROCESSING OPTION: POST PROCESS TIME DOMAIN CONDUCTED EMISSIONS DATA
input_signal.dat
           0   # number of header lines in the file
           1   # time data column
           2   # Voltage data column
   1000000.0000000000        # scale factor for data (eg 1E6 for V to micro V
           0   # number of filters to apply
           1   # Number of sub-data segments of data to process
   6.4000000000000003E-003   # period of each sub-segment
      262144   # number of samples in each sub-segment (this should be a power of 2)
   0.0000000000000000        # time period over which the samples are to be distributed or
enter 0 for continuous subsets
           1   # pad factor (set to 1 for no padding, pad factor should be a power of two)
z # z to pad with zeros or p to create a periodic continuation of the sub-signal
           1   # number of sub-segments to plot (max 9)
r  # time domain window function (r=rectangular, h=Hann)
y  # subtract d.c. (y or n)
V.fft
           2  # number of frequency domain bands with specific detector bandwidths to output
V.favg
   10.000000000000000       # minimum frequency for output
   250000.00000000000       # maximum frequency for output
        1250  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   200.00000000000000       # detector bandwidth
   250000.00000000000       # minimum frequency for output
   2000000.0000000000       # maximum frequency for output
       14875  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   2000.0000000000000       # detector bandwidth
           0  POST PROCESSING OPTION: QUIT
```

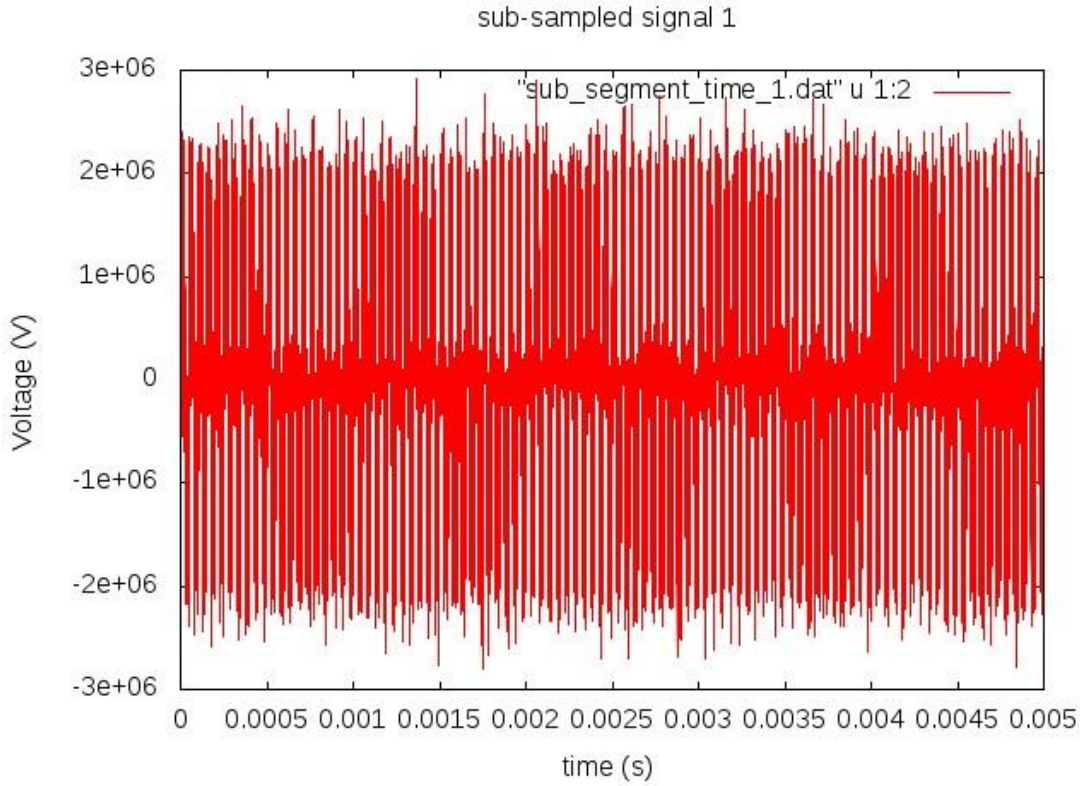

Figure 25. square wave with added noise

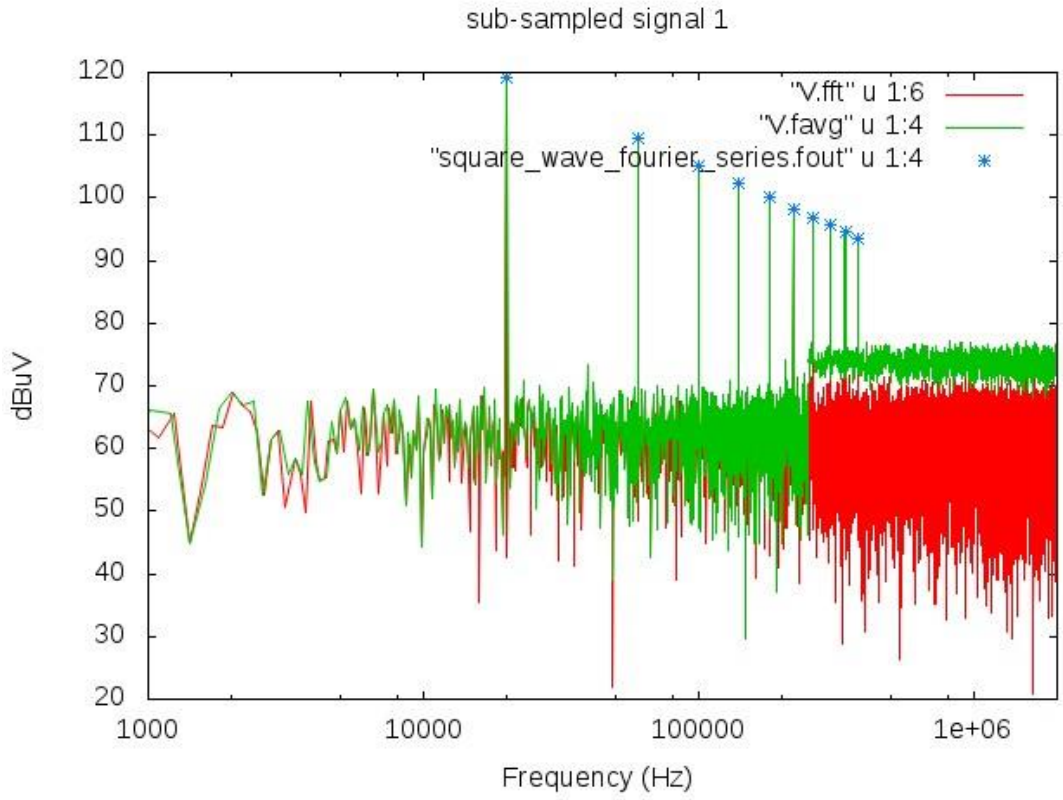

Figure 26. Frequency domain results for test 6.

Test 7

Test 7 shows the processing of 10000 cycles of a 10 harmonic square wave expansion plus 0.5V gaussian noise. The result in Figure 27 shows the result of frequency domain averaging over 100 sub-samples, each of 100 cycles. Two different bandwidths are used over the 200Hz to 2GHz frequency range. The post processing options are shown below.

```
          58  POST PROCESSING OPTION: POST PROCESS TIME DOMAIN CONDUCTED EMISSIONS DATA
input_signal.dat
           0   # number of header lines in the file
           1   # time data column
           2   # Voltage data column
   1000000.0000000000        # scale factor for data (eg 1E6 for V to micro V
           0   # number of filters to apply
         100   # Number of sub-data segments of data to process
   5.0000000000000001E-003   # period of each sub-segment
      262144   # number of samples in each sub-segment (this should be a power of 2)
  0.50000000000000000        # time period over which the samples are to be distributed or
enter 0 for continuous subsets
           1   # pad factor (set to 1 for no padding, pad factor should be a power of two)
z # z to pad with zeros or p to create a periodic continuation of the sub-signal
           4   # number of sub-segments to plot (max 9)
r  # time domain window function (r=rectangular, h=Hann)
y  # subtract d.c. (y or n)
V.fft
           2  # number of frequency domain bands with specific detector bandwidths to output
V.favg
   10.000000000000000       # minimum frequency for output
   250000.00000000000       # maximum frequency for output
        1250  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   200.00000000000000       # detector bandwidth
   250000.00000000000       # minimum frequency for output
   2000000.0000000000       # maximum frequency for output
       14875  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   2000.0000000000000       # detector bandwidth
           0  POST PROCESSING OPTION: QUIT
```

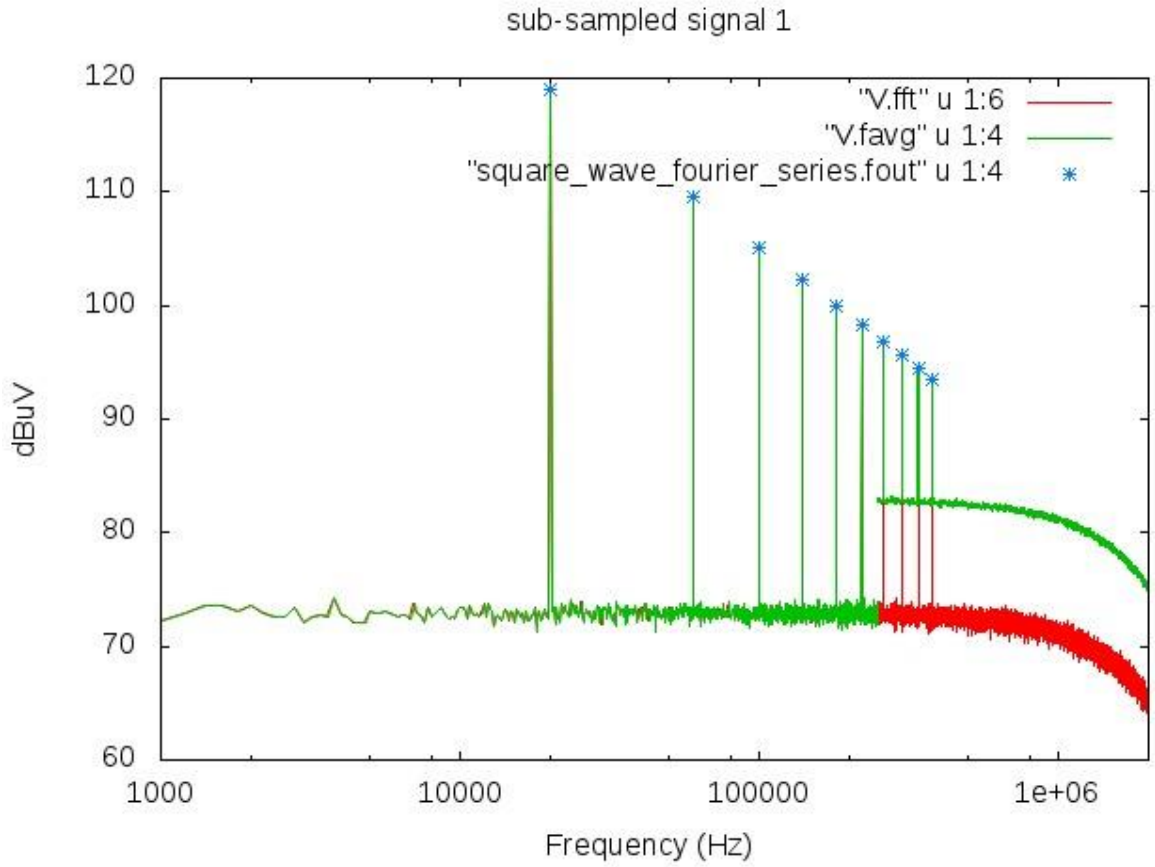

Figure 27. Frequency domain result, Test 7.

Test 8

The post processing of measured conducted emissions data is shown in this test. The time domain LISN voltage data has been measured with a sample rate of 500kHz over a period of 0.1s (Figure 28). 10 sub-segments have been extracted from this dataset and averaged in the frequency domain. The result is shown in Figure 29. The bandwidth of the detector is set to 200Hz over the frequency range 10Hz to 250kHz. This increases the general level of the data (green) over the raw FFT data (red). The post processing options are shown below.

```
          58  POST PROCESSING OPTION: POST PROCESS TIME DOMAIN CONDUCTED EMISSIONS DATA
measured_data.out
           0   # number of header lines in the file
           1   # time data column
           2   # Voltage data column
   1000000.0000000000        # scale factor for data (eg 1E6 for V to micro V
           0   # number of filters to apply
          10   # Number of sub-data segments of data to process
   1.0000000000000000E-002   # period of each sub-segment
      262144   # number of samples in each sub-segment (this should be a power of 2)
  0.10000000000000001        # time period over which the samples are to be distributed or
enter 0 for continuous subsets
           1   # pad factor (set to 1 for no padding, pad factor should be a power of two)
z # z to pad with zeros or p to create a periodic continuation of the sub-signal
           1   # number of sub-segments to plot (max 9)
r  # time domain window function (r=rectangular, h=Hann)
y  # subtract d.c. (y or n)
V.fft
           1  # number of frequency domain bands with specific detector bandwidths to output
V.favg
   10.000000000000000       # minimum frequency for output
   250000.00000000000       # maximum frequency for output
        1250  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   200.00000000000000       # detector bandwidth
           0  POST PROCESSING OPTION: QUIT
```

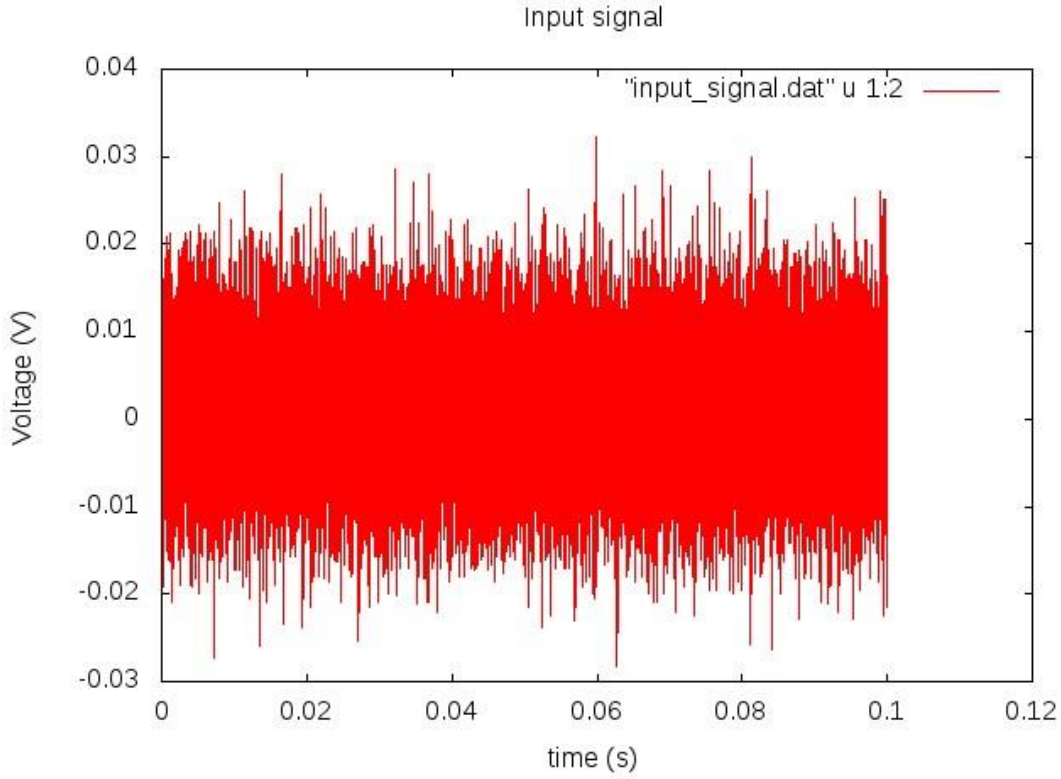

Figure 28. Time domain measured dataset 1.

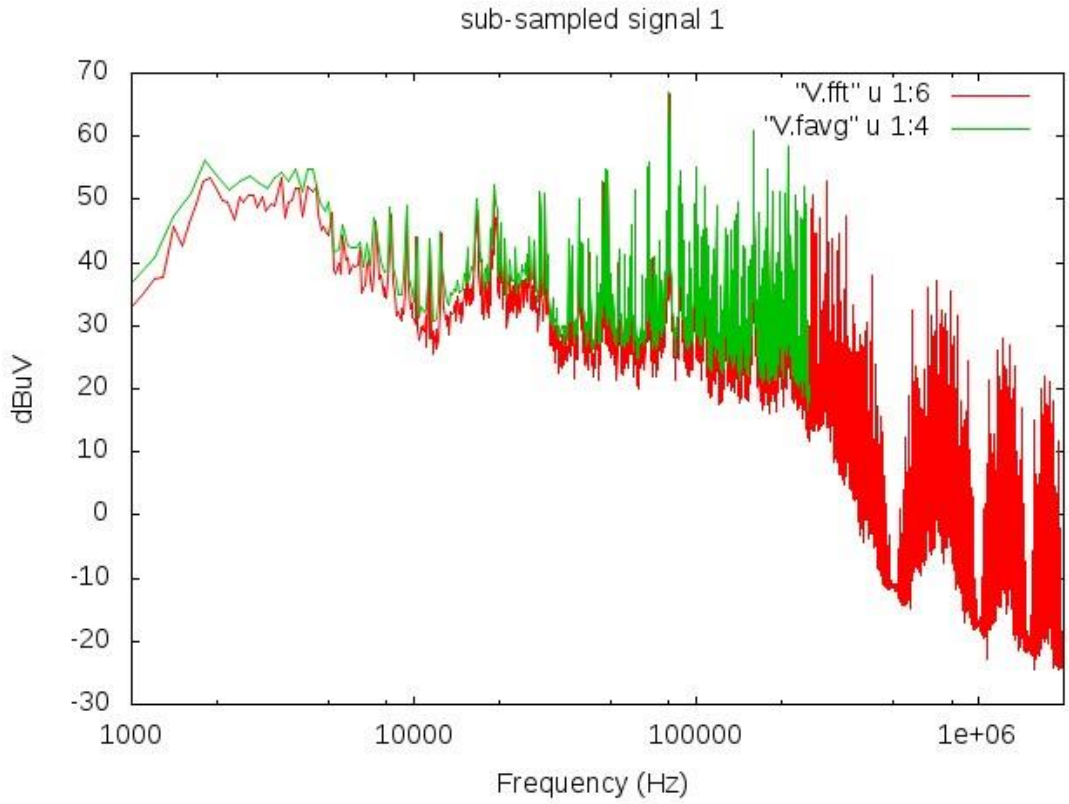

Figure 29. Frequency domain measured data, test 8.

If the complete dataset is processed (post processing options below) then the result in Figure 30 is obtained.

```
          58  POST PROCESSING OPTION: POST PROCESS TIME DOMAIN CONDUCTED EMISSIONS DATA
measured_data.out
           0   # number of header lines in the file
           1   # time data column
           2   # Voltage data column
   1000000.0000000000        # scale factor for data (eg 1E6 for V to micro V
           0   # number of filters to apply
           1   # Number of sub-data segments of data to process
  0.10000000000000001        # period of each sub-segment
      262144   # number of samples in each sub-segment (this should be a power of 2)
  0.10000000000000001        # time period over which the samples are to be distributed or
enter 0 for continuous subsets
           1   # pad factor (set to 1 for no padding, pad factor should be a power of two)
z # z to pad with zeros or p to create a periodic continuation of the sub-signal
           1   # number of sub-segments to plot (max 9)
r  # time domain window function (r=rectangular, h=Hann)
y  # subtract d.c. (y or n)
V.fft
           1  # number of frequency domain bands with specific detector bandwidths to output
V.favg
   10.000000000000000       # minimum frequency for output
   250000.00000000000       # maximum frequency for output
        1250  # number of frequencies for output
r # detector frequency domain function (gaussian or rectangular)
   200.00000000000000       # detector bandwidth
           0  POST PROCESSING OPTION: QUIT
```

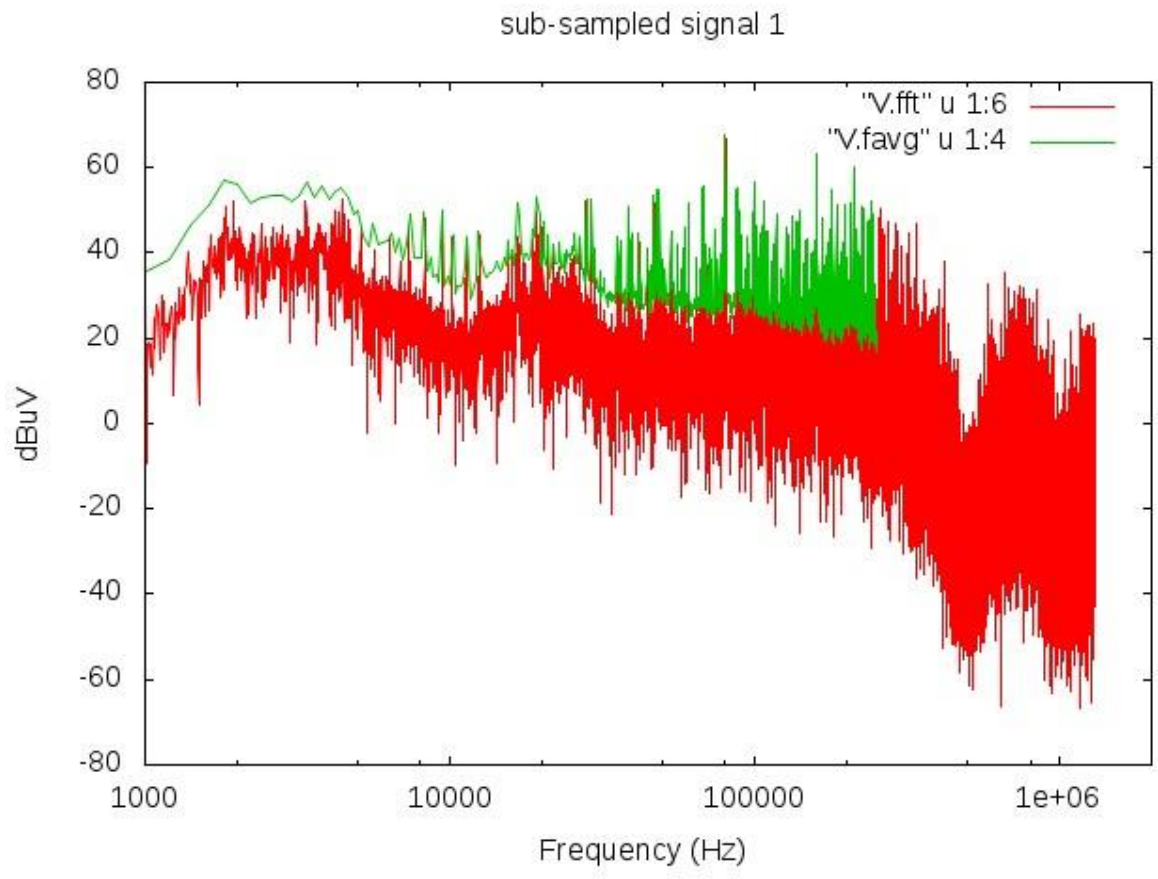

Figure 30. Frequency domain output test 8 when the whole dataset is processed without averaging of sub-samples.

Example1: Transmission line with non-linear load.

In this first example a pulse voltage source drives a diode load via a transmission line. The transmission line is simulated in GGI_TLM and the source and load lumped components are simulated in Ngspice. The configuration is shown in Figure 31.

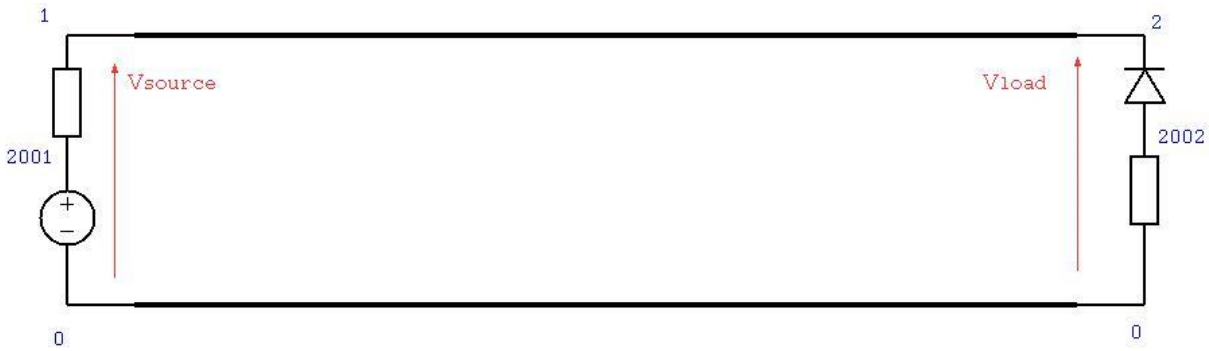

Figure 31 Example 1. Transmission line linking voltage souce and resistance to diode and resistance load

The geometry of the two conductor transmission line is specified as a gerber file (two_track.gbr) and is shown in Figure 32.

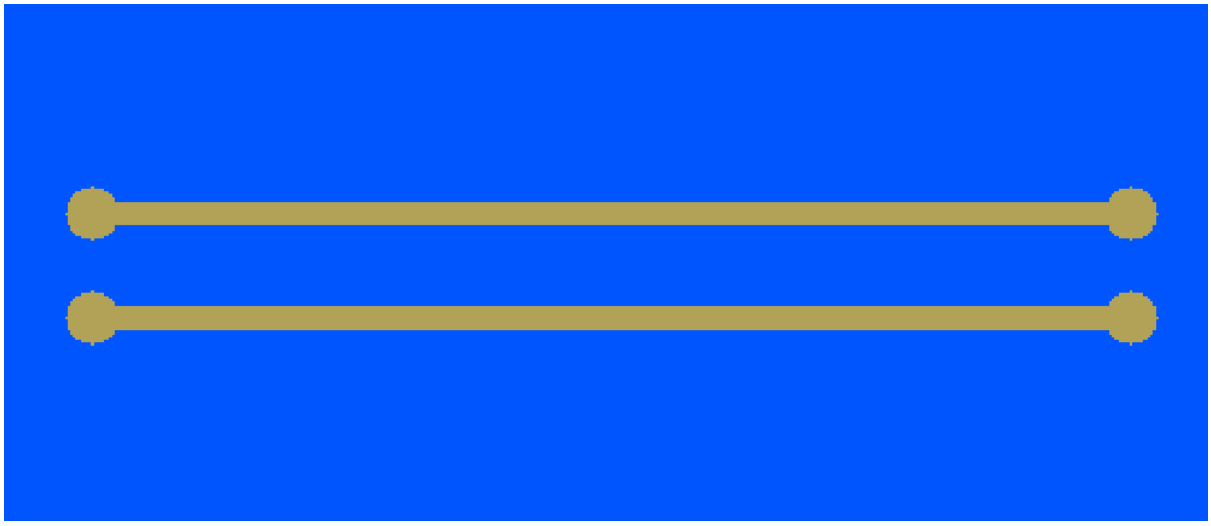

Figure 32 Two track transmission line PCB layout

The Simulation model is built using the process **GGI_TLM_create_PCB_simulation_model**. The input file for this process is shown below.

PCB Simulation Specification file:

```
two_wire_diode_test.inp
0.001  # dl: cell size for the TLM solution
-0.06 -0.02 -0.01   0.06 0.02 0.04  # TLM problem space dimensions:
xmin,ymin,zmin,xmax,ymax,zmax
1      # Number of gerber files to include
two_track.gbr
0.002    # z position for the layer specified in this gerber file
0      # Number of dielectric layers
```

```
0      # Number of vias
2      # Number of lumped components
1      # COMPONENT NUMBER
one_port_model  # Component type for component 1
1      # Number of ports
1      # Port number
1      # Ngspice node number for port 1
-0.05  0.005 0.002  # connection point coordinates for connection number 1, port 1
-0.05 -0.005 0.002  # connection point coordinates for connection number 2, reference for port
1
0.004     # z offset for component
none      # package type
2      # COMPONENT NUMBER
one_port_model  # Component type for component 2
1      # Number of ports
2      # Port number
2      # Ngspice node number for port 1
0.05  0.005 0.002  # connection point coordinates for connection number 1, port 1
0.05 -0.005 0.002  # connection point coordinates for connection number 2, reference for port
1
0.004     # z position for component
none      # package type
0        # Number of additional components (heatsinks etc)

* START of GGI_TLM input file text *

ngspice_node_output_list
2   # number of ngspice output nodes
1   # NGSPICE OUTPUT NUMBER
1   # ngspice node number
2   # NGSPICE OUTPUT NUMBER
2   # ngspice node number

ngspice_timestep_factor
4

Simulation_time
1e-8

* END of GGI_TLM input file text *

* START of Ngspice input file text *

* Model to be included in the GGI_TLM simulation for port 1, connected to node 1
* in this case a voltage source with series resistance
Rs1     2001   1   100.0
Vs1     2001   0   EXP( 0.0     1.0    0.000000E+00    2.000000E-010    1.200000E-09
2.000000E-010 )
*
* Model to be included in the GGI_TLM simulation for port 2, connected to node 2
* in this case a diode with series resistance
Diode2   2002   2   Dmod
Rl2      2002   0   100.0
.model DMOD D ( is=1e-12 )
*

* END of Ngspice input file text *
```

The GGI_TLM_create_PCB_simulation_model process creates a GGI_TLM input file (two_wire_diode_test.inp) and a template circuit file for the Ngspice model (Spice_circuit_TEMPLATE.cir) file. The Ngspice circuit for the simulation is shown in Figure 33 below in which the source and load end components are driven by separate Thevenin equivalent circuits which form the links to GGI_TLM. The 3D Simulation model is shown in Figure 34.

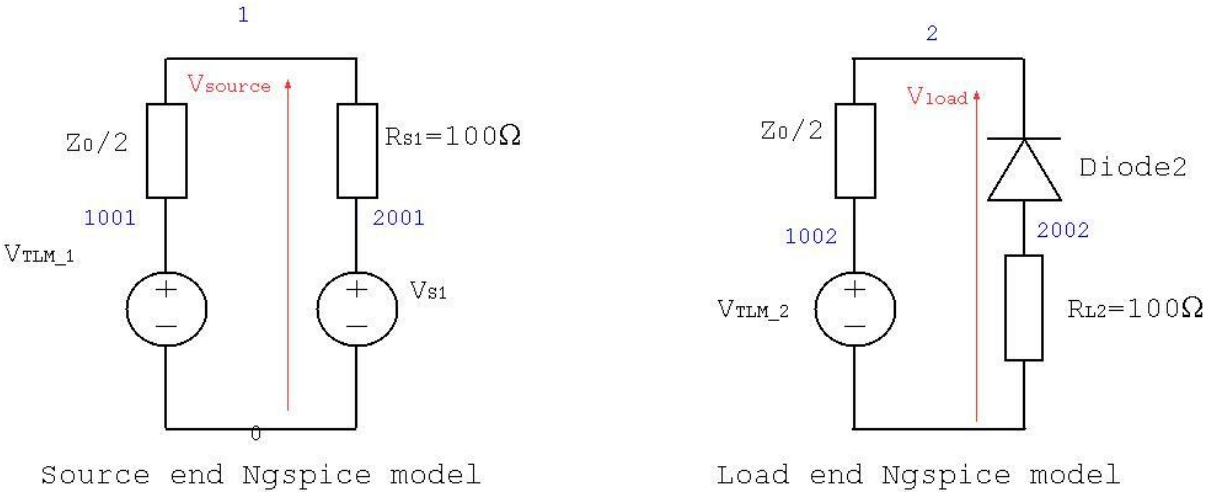

Figure 33 Thevenin equivalent circuit models of source and load end components. Ngspice node numbers are shown in blue.

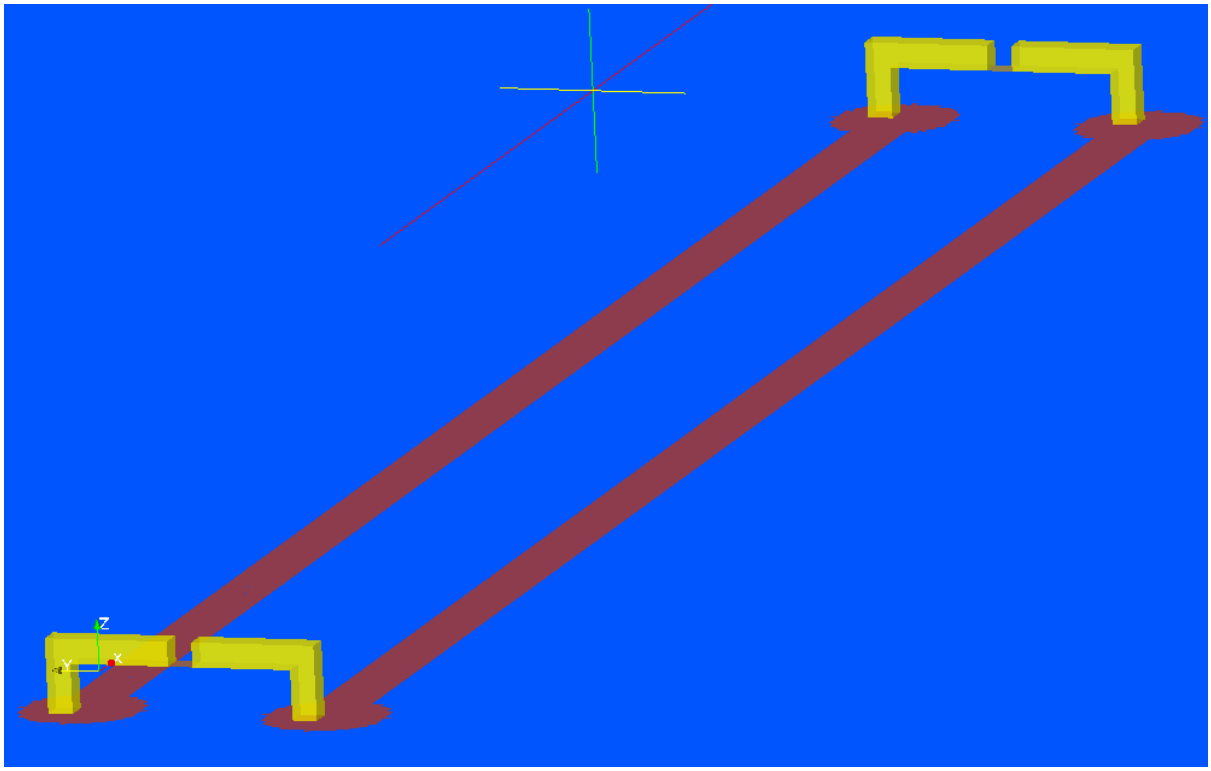

Figure 34. 3D GGI_TLM Model of the two wire transmission line configuration with component models in place

GGI_TLM input file:

```
Mesh_outer_boundary_dimension
   -6.000000E-02    6.000000E-02   -2.000000E-02    2.000000E-02   -1.000000E-02    4.000000E-
02

Mesh_cell_dimension
    1.000000E-03

Outer_boundary_reflection_coefficient
0.0 0.0 0.0 0.0 0.0 0.0
```

```
Surface_list
     4  # Number of surfaces
     1  # SURFACE NUMBER
stl_triangulated_surface
two_track.gbr.stl
1.0 0.0
    0.000000E+00    0.000000E+00    0.000000E+00
    0.000000E+00    0.000000E+00    2.000000E-03
     2  # SURFACE NUMBER
zplane
   -5.100000E-02    8.673617E-19    4.000000E-03   -5.000000E-02    1.000000E-03    4.000000E-
03
    0.000000E+00    0.000000E+00    0.000000E+00
    0.000000E+00    0.000000E+00    0.000000E+00
     3  # SURFACE NUMBER
zplane
    5.000000E-02    8.673617E-19    4.000000E-03    5.100000E-02    1.000000E-03    4.000000E-
03
    0.000000E+00    0.000000E+00    0.000000E+00
    0.000000E+00    0.000000E+00    0.000000E+00
     4  # SURFACE NUMBER
stl_triangulated_surface
component_terminal_connections.stl
1.0 0.0
    0.000000E+00    0.000000E+00    0.000000E+00
    0.000000E+00    0.000000E+00    0.000000E+00

Surface_material_list
     4  # Number of surface materials
     1  # SURFACE MATERIAL NUMBER
PEC
     1  # Number of surfaces
    1 # Surface list
 1 # Surface orientation list
     2  # SURFACE MATERIAL NUMBER
SPICE
     1  # Ngspice port number
     1     0  # Ngspice node numbers
-y  # Port direction
     1  # Number of surfaces
     2  # Surface number
     1  # Surface orientation list
     3  # SURFACE MATERIAL NUMBER
SPICE
     2  # Ngspice port number
     2     0  # Ngspice node numbers
-y  # Port direction
     1  # Number of surfaces
     3  # Surface number
     1  # Surface orientation list
     4  # SURFACE MATERIAL NUMBER
PEC
     1  # Number of surfaces
     4  # Surface number
     1  # Surface orientation list

volume_list
     0   # Number of volumes

volume_material_list
     0   # Number of volume materials

ngspice_node_output_list
2   # number of ngspice output nodes
1   # NGSPICE OUTPUT NUMBER
1   # ngspice node number
2   # NGSPICE OUTPUT NUMBER
2   # ngspice node number

ngspice_timestep_factor
4

Simulation_time
1e-8
```

Spice template input file:

Following is the template Spice circuit file. Note that this is a template file as the TLM Thevenin equivalent source impedance needs to be substituted along with the timestep information in the .TRAN line. This is done automatically by the GGI_TLM software. The voltage source values (Vtlm*) are set as the simulation runs.

```
Ngspice template file for GGI_TLM - ngspice linked simulation
*
* GGI_TLM link port    1 using nodes    1 and   0
*
* Voltage source with series resistance: equivalent circuit of TLM link
 Vtlm1        1001           0  DC  0.0
 Rtlm1        1001           1  #Z0_TLM
*
* GGI_TLM link port    2 using nodes    2 and   0
*
* Voltage source with series resistance: equivalent circuit of TLM link
 Vtlm2        1002           0  DC  0.0
 Rtlm2        1002           2  #Z0_TLM
*
*
* Voltage source required for the voltage source controlling the breakpoint time
Vbreak time_node 0 DC 0.0
*

* Model to be included in the GGI_TLM simulation for port 1, connected to node 1
* in this case a voltage source with series resistance
Rs1     2001   1   100.0
Vs1     2001   0   EXP( 0.0     1.0    0.000000E+00    2.000000E-010    1.200000E-09
2.000000E-010 )
*
* Model to be included in the GGI_TLM simulation for port 2, connected to node 2
* in this case a diode with series resistance
Diode2   2002   2   Dmod
R12      2002   0   100.0
.model DMOD D ( is=1e-12 )
*
*
* Control for transient simulation
.TRAN #dt_out  #tmax_ngspice 0.0 #dt_ngspice  UIC
*
.END
```

The model is validated by simulating the whole circuit, including the transmission line in Ngspice. The transmission line is included in the Ngspice circuit as a delay line with an appropriate characteristic impedance and time delay. The comparison of the time domain voltage at the source and load ends is shown in Figure 35. The Spice circuit file for the validation circuit including the delay line is included below.

Ngspice validation model:

```
Ngspice validation circuit for GGI_TLM - ngspice linked simulation demonstration

* Source end voltage source with series resistance
Rs1     2001   1   100.0
Vs1     2001   0   EXP( 0.0     1.0    0.000000E+00    2.000000E-010    1.200000E-09
2.000000E-010 )
*
* Delay line used to link source and load ends
*
T1 1 0 2 0 Z0=320 TD=4.4e-10
*
* Load end Diode and series resistance
Diode2   2002   2   Dmod
R12      2002   0   100.0
.model DMOD D ( is=1e-12 )
*
*
```

```
* Control for transient simulation
.TRAN    4.169551E-13    1.000267E-08 0.0    4.169551E-13
.PRINT TRAN V(1) V(2)
*
.END
```

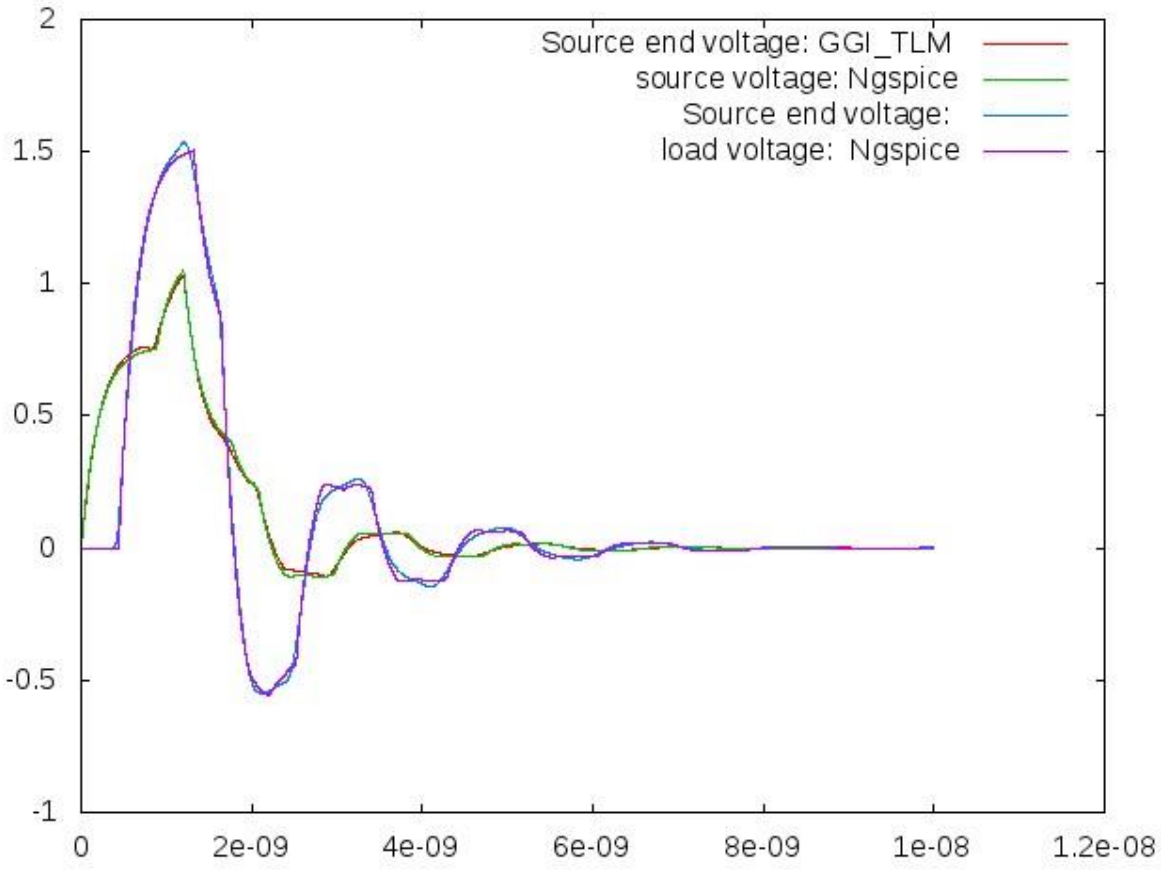

Figure 35. Comparison between GGI_TLM – Ngspice simulation and Ngspice simulation with delay line.

Example 2: Simple buck converter model

Figure 36 shows the circuit for a simple dc-dc buck converter designed to convert the 20V input to 5V at the load.

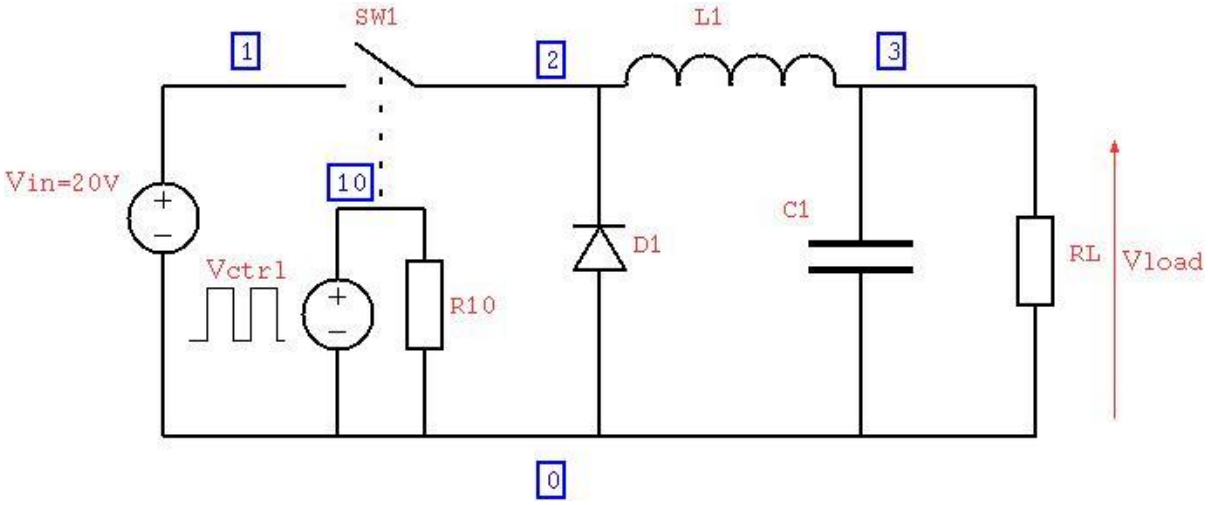

Figure 36 dc-dc converter circuit diagram

An initial ngspice model has been created in order to test the correct operation of the idealised circuit and to calculate the initial conditions for the GGI_TLM- Ngspice model. The calculation of initial conditions for the inductor current and the large output capacitor is due to the very large time constants of the turn on transient. The load voltage is shown as a function of time in Figure 37 and the inductor current in Figure 38. A simulation time of 200μs has been used to allow the converter to reach a steady state. The detail of the steady state inductor current is shown in Figure 39. Note that the converter is switched at very high frequency. This is due to the known memory leak problem with GGI_TLM with Ngspice which currently limits the number of timesteps which can be simulated before the memory required exceeds that available. The high switching frequency is chosen such that a number of complete cycles of the converter can be simulated on a normal PC.

```
BASIC BUCK CONVERTER
*
* SWITCH DRIVER
VCTRL   10 0    PULSE(0V 6V 0 0.0001US 0.0001US 0.05US 0.2US)
R10     10 0    1MEG
*
* INPUT VOLTAGE
VIN     1       0       DC      20
*
* CONVERTER
SW1     1 2     10 0    SWitch1
D1      0       2       DSCH
L1      2       30      50UH
VL1     30       3       0.0
C1      3       0       2UF
*
* LOAD
RL      3       0       5.0
*
.MODEL  SWitch1         SW      Vt=5V Vh=0.2V RON=0.01 ROFF=1MEG
.MODEL DSCH D( IS=0.0002  )
*
.OPTIONS NOPAGE
* ANALYSIS
.TRAN   0.01US          200US
```

```
*
* VIEW RESULTS
.PRINT tran V(3) I(VL1)

.END
```

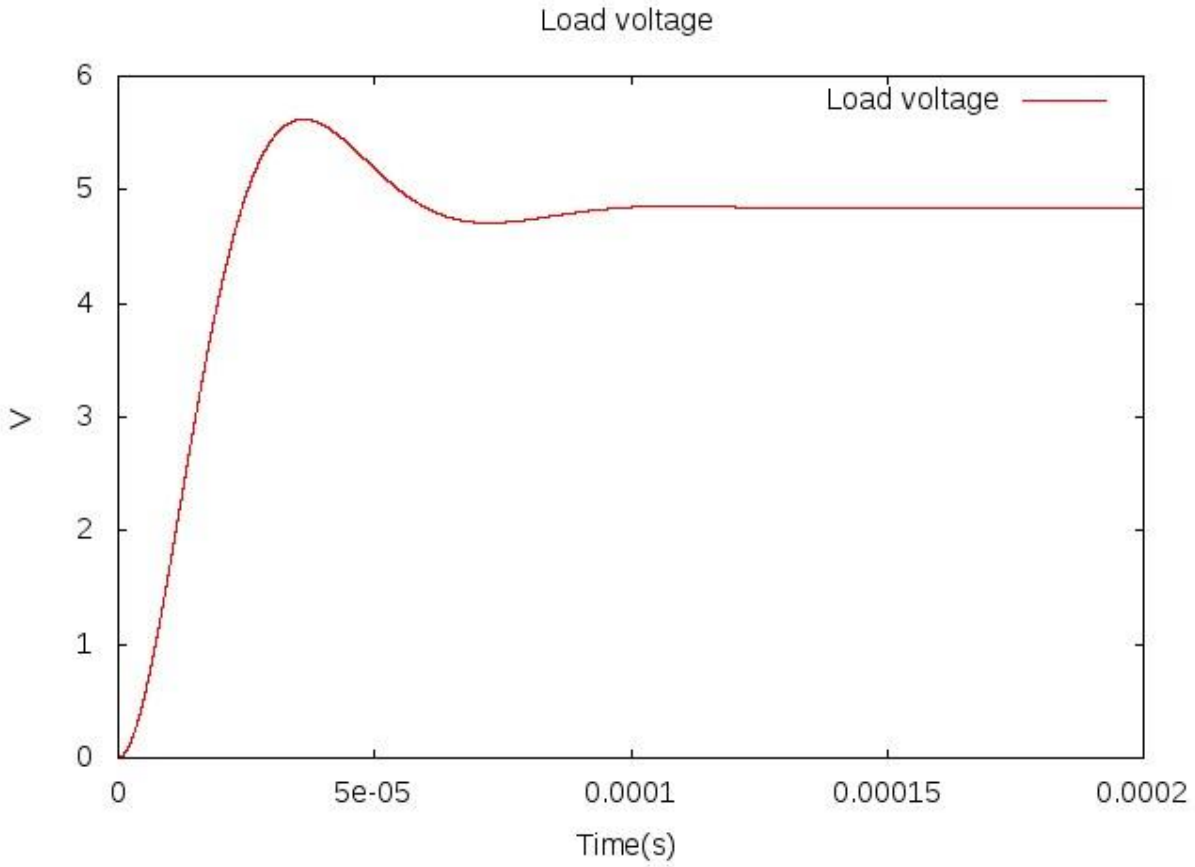

Figure 37 Switch on transient converter output voltage

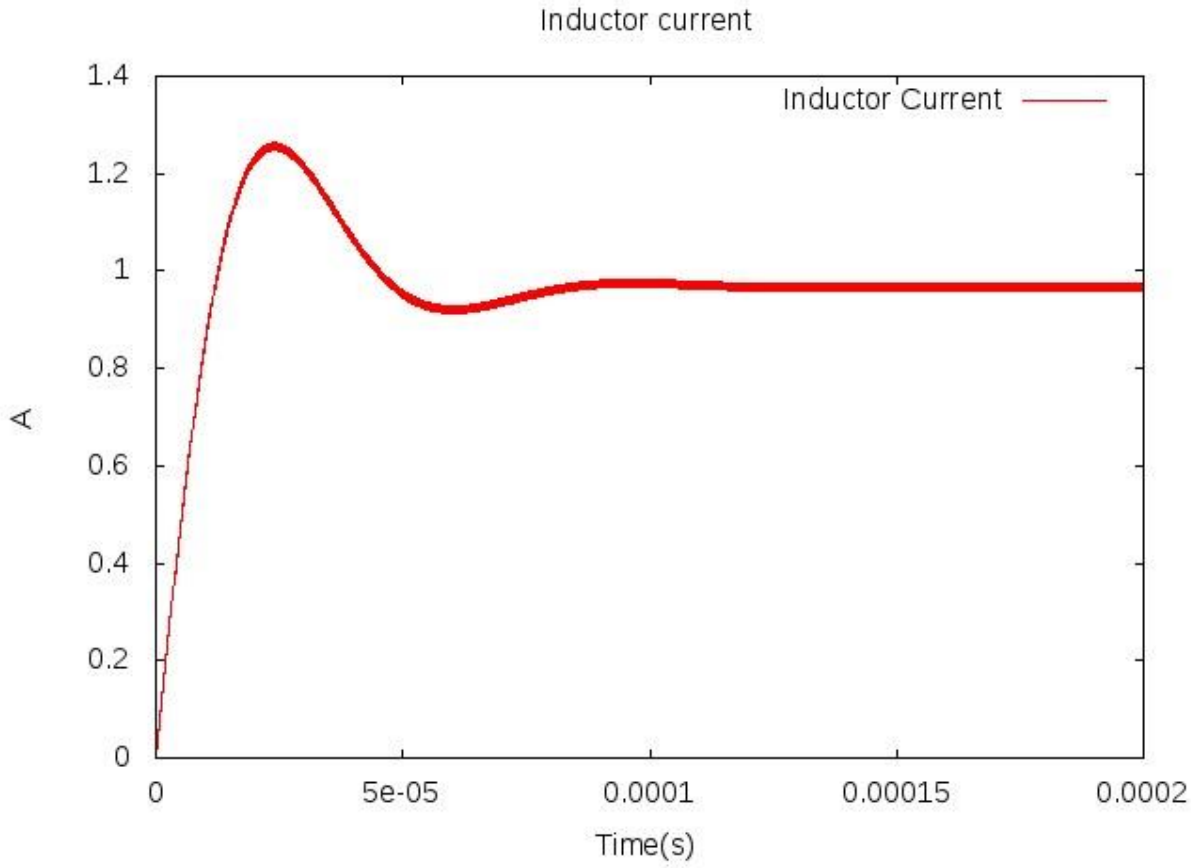

Figure 38 Switch on transient inductor current

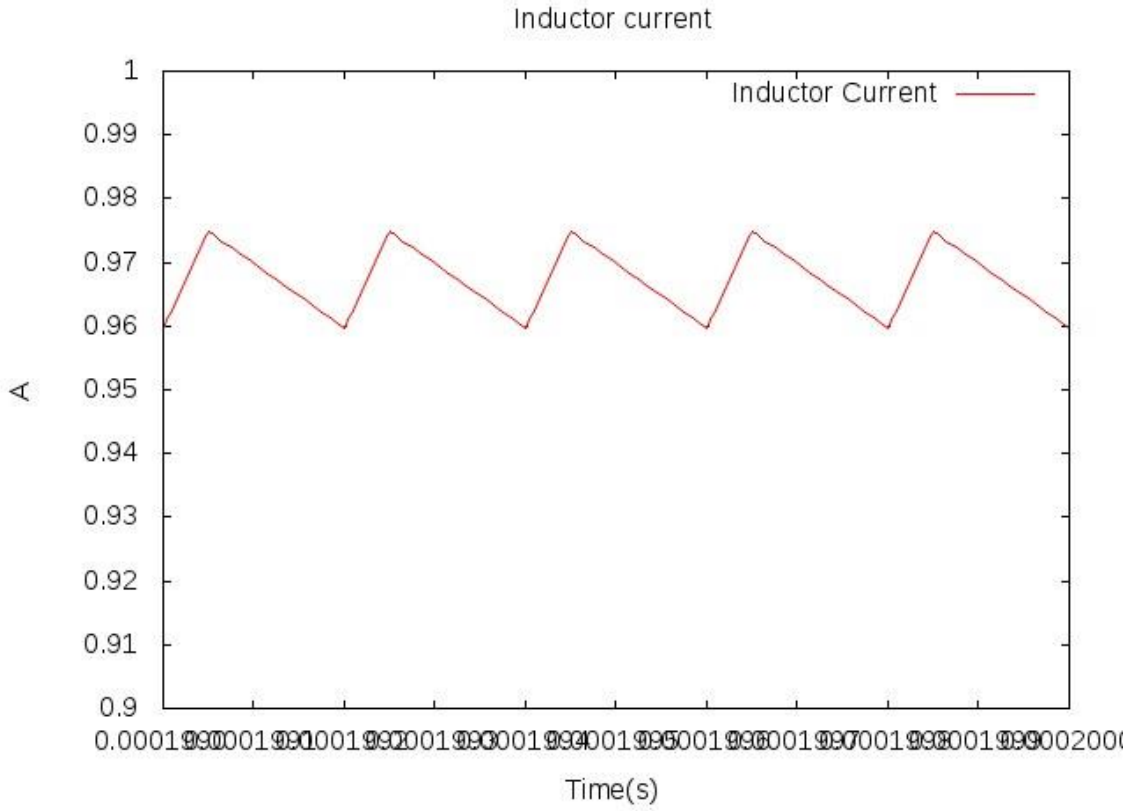

Figure 39 Detail of switch on transient inductor current approaching steady state

A second Ngspice simulation has been performed in which the steady state output capacitor voltage and inductor current have been set as initial conditions as seen in the Ngspice circuit file below.

```
BASIC BUCK CONVERTER using initial conditions
*
* SWITCH DRIVER
VCTRL    10 0   PULSE(0V 6V 0 0.0001US 0.0001US 0.05US 0.2US)
R10      10 0   1MEG
*
* INPUT VOLTAGE
VIN      1       0       DC      20
*
* CONVERTER
SW1      1 2     10 0    SWitch1
D1       0       2       DSCH
L1       2       30      50UH   IC=0.967A
VL1     30        3       0.0
C1       3       0       2UF    IC=4.835V
*
* LOAD
RL       3       0       5.0
*
.MODEL   SWitch1         SW      Vt=5V Vh=0.2V RON=0.01 ROFF=1MEG
.MODEL DSCH D( IS=0.0002  )
*
.OPTIONS NOPAGE
* ANALYSIS
.TRAN   0.01US          1US    UIC
*
* VIEW RESULTS
.PRINT tran V(3) I(VL1)
.END
```

Figure 40 and Figure 41 show the load voltage and inductor current as a function of time with the initial conditions applied where it is seen that the steady state operation has been reached from the start of the simulation. The approach of using initial conditions to eliminate the switch on transient must be used in the GGI_TLM simulation as it is not feasible to run GGI_TLM for the length of time required to calculate the initial transient behaviour of the circuit.

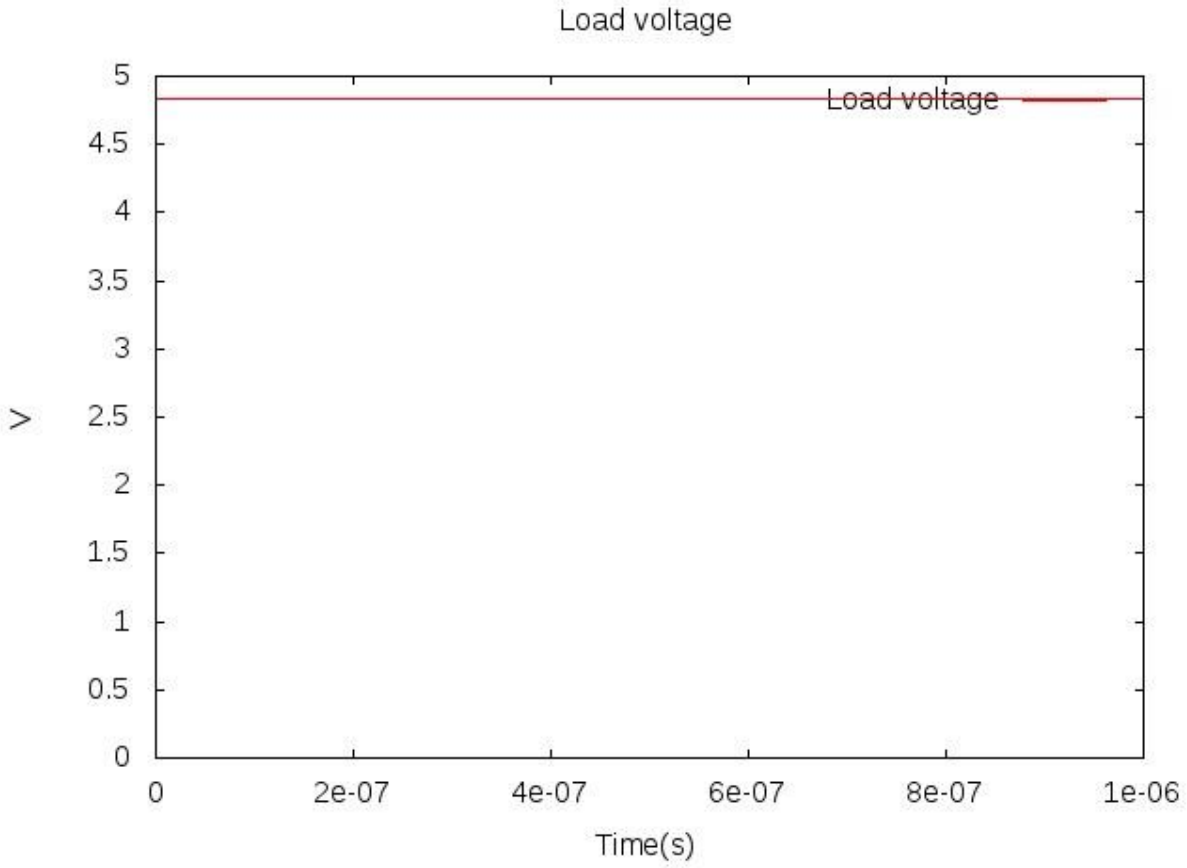

Figure 40 Load voltage using initial conditions

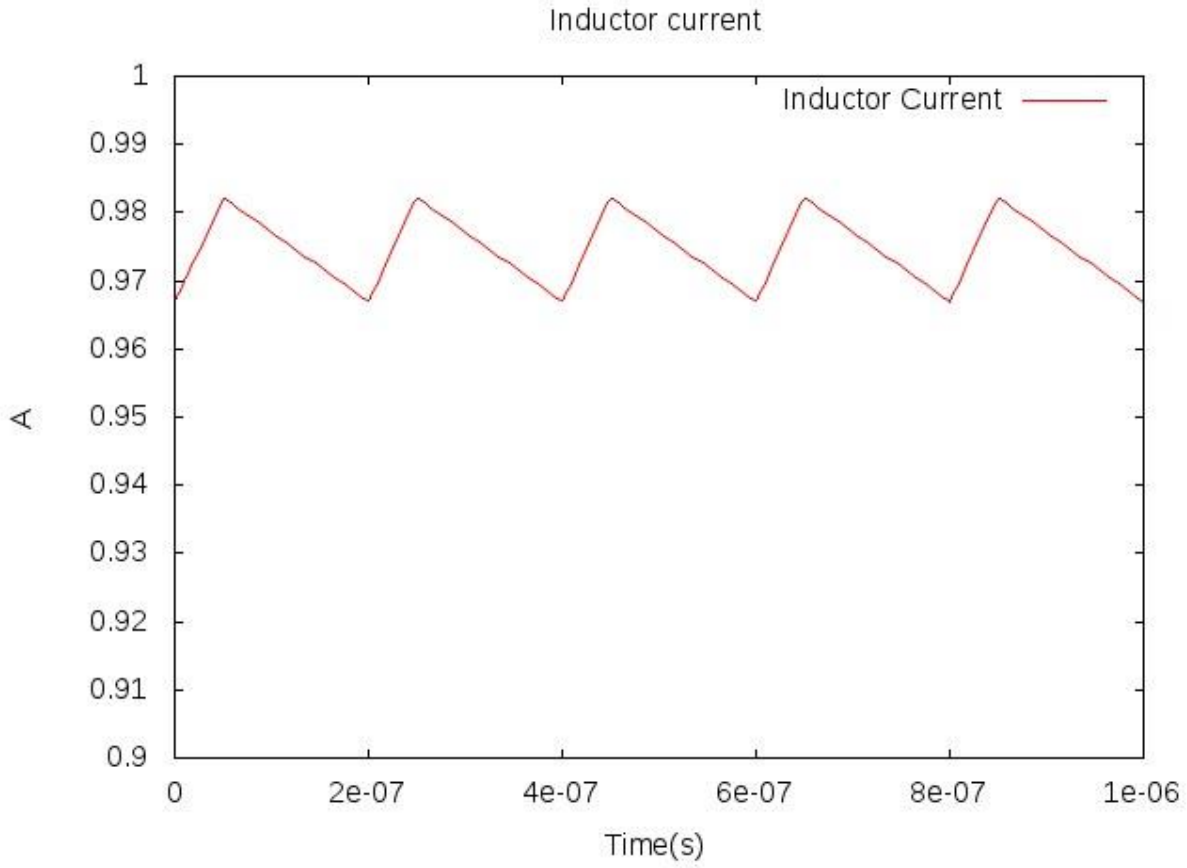

Figure 41 Inductor current using initial conditions

The GGI_TLM-Ngspice model development starts with the PCB layout which is specified in a gerber file. The PCB geometry is shown in Figure 42.

Figure 42 PCB layout for the converter model

The model is created using the **GGI_TLM_create_PCB_simulation_model** process. The input file to the process is shown below:

```
converter.inp
0.001  # dl: cell size for the TLM solution
-0.01 -0.01 -0.03   0.07 0.04 0.01  # TLM problem space dimensions:
xmin,ymin,zmin,xmax,ymax,zmax
1      # Number of gerber files to include
top_layer.gbr
0.002    # z position for the layer specified in this gerber file
2      # Number of dielectric layers. Here the first layer idefines an output volume for H
field output, the second is the FR4 substrate
-0.01 -0.01 0.006   0.07 0.04 0.007  # outer dimensions of the dielectric layer. Material
filename follows (without .vmat extension)
air
0.0 0.0 0.0    0.06 0.03 0.002  # outer dimensions of the dielectric layer. Material filename
follows (without .vmat extension)
FR4
0      # Number of vias
6      # Number of lumped components
1      # COMPONENT NUMBER
one_port_model  # Component type for component 1
1      # Number of ports
1      # Port number(s)
1  0   # Ngspice node numbers for port 1
0.005  0.020 0.002  # connection point coordinates for connection number 1, port
0.005  0.015 0.002  # connection point coordinates for connection number 2, reference for port
0.003     # z position for component
none      # package type
2      # COMPONENT NUMBER
two_port_model  # Component type for component 2
1         # Number of ports
2         # Port number(s)
2  0        # Ngspice node number for port 2
0.020  0.020 0.002  # connection point coordinates for connection number 1, port
0.025  0.020 0.002  # connection point coordinates for connection number 2, reference for port
-0.003              # z position for component
rectangular         # package type
-0.0055  0.0005  -0.0015     0.0055  -0.0095   0.0015      # package parameters. Material
filename follows (without .vmat extension)
component_case
1          # Number of PEC surfaces
zmin   2   # PEC surface and the terminal to electrically connect to
3      # COMPONENT NUMBER
one_port_model  # Component type for component 3
1       # Number of ports
3       # Port number(s)
3  0    # Ngspice node number for port
0.025  0.015 0.002  # connection point coordinates for connection number 1, port
0.025  0.005 0.002  # connection point coordinates for connection number 2, reference for port
0.003     # z position for component
none      # package type
4      # COMPONENT NUMBER
one_port_model  # Component type for component 4
1       # Number of ports
4       # Port number(s)
4  0    # Ngspice node number for port
0.030  0.020 0.002  # connection point coordinates for connection number 1, port
0.040  0.020 0.002  # connection point coordinates for connection number 2, reference for port
0.003     # z position for component
none      # package type
5      # COMPONENT NUMBER
one_port_model  # Component type for component 5
1       # Number of ports
5       # Port number(s)
5  0    # Ngspice node number for port
0.045  0.015 0.002  # connection point coordinates for connection number 1, port
0.045  0.005 0.002  # connection point coordinates for connection number 2, reference for port
0.003     # z position for component
none      # package type
6      # COMPONENT NUMBER
one_port_model  # Component type for component 6
1       # Number of ports
6       # Port number(s)
6  0    # Ngspice node number for port
0.055  0.015 0.002  # connection point coordinates for connection number 1, port 1
0.055  0.005 0.002  # connection point coordinates for connection number 2, reference for port
0.003     # z position for component
none      # package type
```

```
2         # Number of additional components (heatsinks etc)
heatsink
-0.002  0.0 -0.020    0.062 0.03 -0.005  # outer dimensions of the heatsink
10                                       # number of slots
y                                        # slot width direction
zmin                                     # slot depth direction (face from which slots are
cut)
0.004                                    # width of slots
0.010                                    # depth of slots
dielectric
0.015  0.010  -0.005     0.026  0.020   -0.004 # outer dimensions of the dielectric. Material
name follows (without .vmat extension)
ceramic_tile
* START of GGI_TLM input file text *

ngspice_node_output_list
8   # number of ngspice output nodes
1   # NGSPICE OUTPUT NUMBER
1   # ngspice node number      # Supply voltage
2   # NGSPICE OUTPUT NUMBER
2   # ngspice node number      # Switch  voltage
3   # NGSPICE OUTPUT NUMBER
3   # ngspice node number      # Diode voltage
4   # NGSPICE OUTPUT NUMBER
4   # ngspice node number      # Inductor voltage
5   # NGSPICE OUTPUT NUMBER
5   # ngspice node number      # Capacitor voltage
6   # NGSPICE OUTPUT NUMBER
6   # ngspice node number      # load voltage
7   # NGSPICE OUTPUT NUMBER
7   # ngspice node number      # control voltage 1
8   # NGSPICE OUTPUT NUMBER
8   # ngspice node number      # control voltage 2

ngspice_timestep_factor
8                  (real)

ngspice_lpf_alpha
0.1                  (real)

Output_surface_list
1               ! number of output surfaces
1               ! OUTPUT SURFACE NUMBER
1                  surface number
0               ! side of surface for output (+1 or -1)
Jm
output_time_information
0.5e-6  ! first output time
1.0e-6  ! last output time
0.025e-6       ! output time interval

Output_volume_list
2 ! number of output volumes
1 ! OUTPUT NUMBER
1 ! volume number for output
Hx
output_time_information
0.5e-6  ! first output time
1.0e-6  ! last output time
0.025e-6       ! output time interval
2 ! OUTPUT NUMBER
1 ! volume number for output
Hy
output_time_information
0.5e-6  ! first output time
1.0e-6  ! last output time
0.025e-6       ! output time interval

Simulation_time
1e-6

* END of GGI_TLM input file text *

* START of Ngspice input file text *

* Model to be included in the GGI_TLM simulation connected to node 1
* INPUT VOLTAGE
```

```
VIN     1       0       DC      20.0

* Model to be included in the GGI_TLM simulation connected to node 2
* Voltage controlled switch model
SW1     2       0       8  0    SWitch1
** Switch snubber
CSW1 2 100 22e-12
RSW1 100 0 10.0

* Model to be included in the GGI_TLM simulation connected to node 3
* Diode model
D1      0       3       DSCH  IC=0.967A
** Diode snubber
CD1 0 101 22e-12
RD1 101 3 10.0

* Model to be included in the GGI_TLM simulation connected to node 4
* Inductor model
L1      4       0       500UH  IC=0.967A
*
* Model to be included in the GGI_TLM simulation connected to node 5
* Capacitor model
C1      5       0       4UF   IC=4.835V

* Model to be included in the GGI_TLM simulation connected to node 6
* LOAD model
RL      6       0       5.0
*
* SWITCH CONTROL SIGNAL
VCTRL   8       0       PULSE(0V 6V 0 0.0001US 0.0001US 0.125US 0.5US)
Rctrl   8       0       1MEG
*
.MODEL  SWitch1         SW      Vt=5V Vh=0.2V RON=0.01 ROFF=1MEG
*
.MODEL DSCH D( IS=0.0002 )

*
* END of Ngspice input file text *
```

This process creates a GGI_TLM input file and a template Ngspice circuit file (with #Z0_TLM and simulation time still to be substituted) as shown below. A diagram of this Ngspice circuit is shown in Figure 43. Note that initial conditions have been set for the output capacitor voltage and the inductor current. (Note that this GGI_TLM model includes diode and switch snubber circuits, also the switching frequency has been reduced).

```
Ngspice template file for GGI_TLM - ngspice linked simulation
*
* GGI_TLM link port    1 using nodes    1 and   0
*
* Voltage source with series resistance: equivalent circuit of TLM link
 Vtlm1        1001           0  DC  0.0
 Rtlm1        1001           1  #Z0_TLM
*
* GGI_TLM link port    2 using nodes    2 and   0
*
* Voltage source with series resistance: equivalent circuit of TLM link
 Vtlm2        1002           0  DC  0.0
 Rtlm2        1002           2  #Z0_TLM
*
* GGI_TLM link port    3 using nodes    3 and   0
*
* Voltage source with series resistance: equivalent circuit of TLM link
 Vtlm3        1003           0  DC  0.0
 Rtlm3        1003           3  #Z0_TLM
*
* GGI_TLM link port    4 using nodes    4 and   0
*
* Voltage source with series resistance: equivalent circuit of TLM link
 Vtlm4        1004           0  DC  0.0
 Rtlm4        1004           4  #Z0_TLM
*
* GGI_TLM link port    5 using nodes    5 and   0
*
* Voltage source with series resistance: equivalent circuit of TLM link
```

```
 Vtlm5        1005           0  DC  0.0
 Rtlm5        1005           5  #Z0_TLM
*
* GGI_TLM link port    6 using nodes    6 and   0
*
* Voltage source with series resistance: equivalent circuit of TLM link
 Vtlm6        1006           0  DC  0.0
 Rtlm6        1006           6  #Z0_TLM
*
*
* Voltage source required for the voltage source controlling the breakpoint time
Vbreak time_node 0 DC 0.0
*

* Model to be included in the GGI_TLM simulation connected to node 1
* INPUT VOLTAGE
VIN     1       0       DC      20.0

* Model to be included in the GGI_TLM simulation connected to node 2
* Voltage controlled switch model
SW1     2       0       8  0    SWitch1
** Switch snubber
CSW1 2 100 22e-12
RSW1 100 0 10.0

* Model to be included in the GGI_TLM simulation connected to node 3
* Diode model
D1      0       3       DSCH  IC=0.967A
** Diode snubber
CD1 0 101 22e-12
RD1 101 3 10.0

* Model to be included in the GGI_TLM simulation connected to node 4
* Inductor model
L1      4       0       500UH  IC=0.967A
*
* Model to be included in the GGI_TLM simulation connected to node 5
* Capacitor model
C1      5       0       4UF   IC=4.835V

* Model to be included in the GGI_TLM simulation connected to node 6
* LOAD model
RL      6       0       5.0
*
* SWITCH CONTROL SIGNAL
VCTRL   8       0       PULSE(0V 6V 0 0.0001US 0.0001US 0.125US 0.5US)
Rctrl   8       0       1MEG
*
.MODEL  SWitch1        SW      Vt=5V Vh=0.2V RON=0.01 ROFF=1MEG
*
.MODEL DSCH D( IS=0.0002 )

*
*
* Control for transient simulation
.TRAN #dt_out  #tmax_ngspice 0.0 #dt_ngspice  UIC
*
.END
```

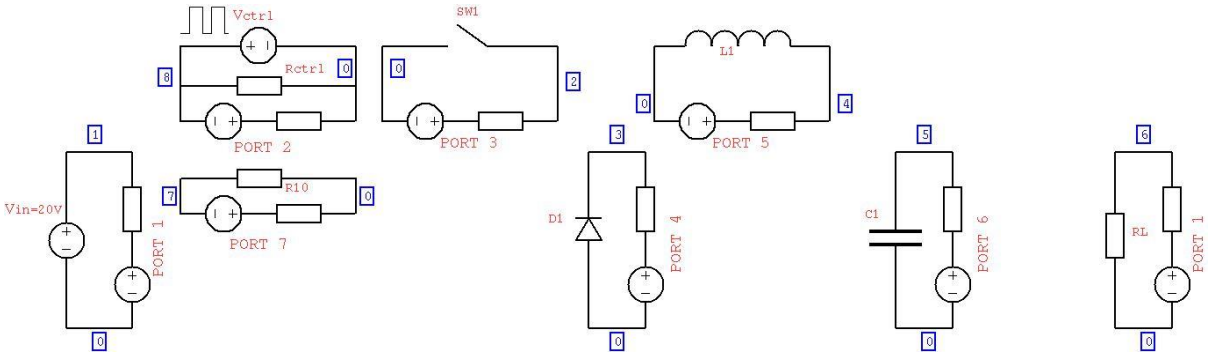

Figure 43 Ngspice model for the converter with the GGI_TLM linking components added (switch and diode snubber components have been removed for clarity)..

The 3D GGI_TLM simulation model is shown in Figure 44 and Figure 45.

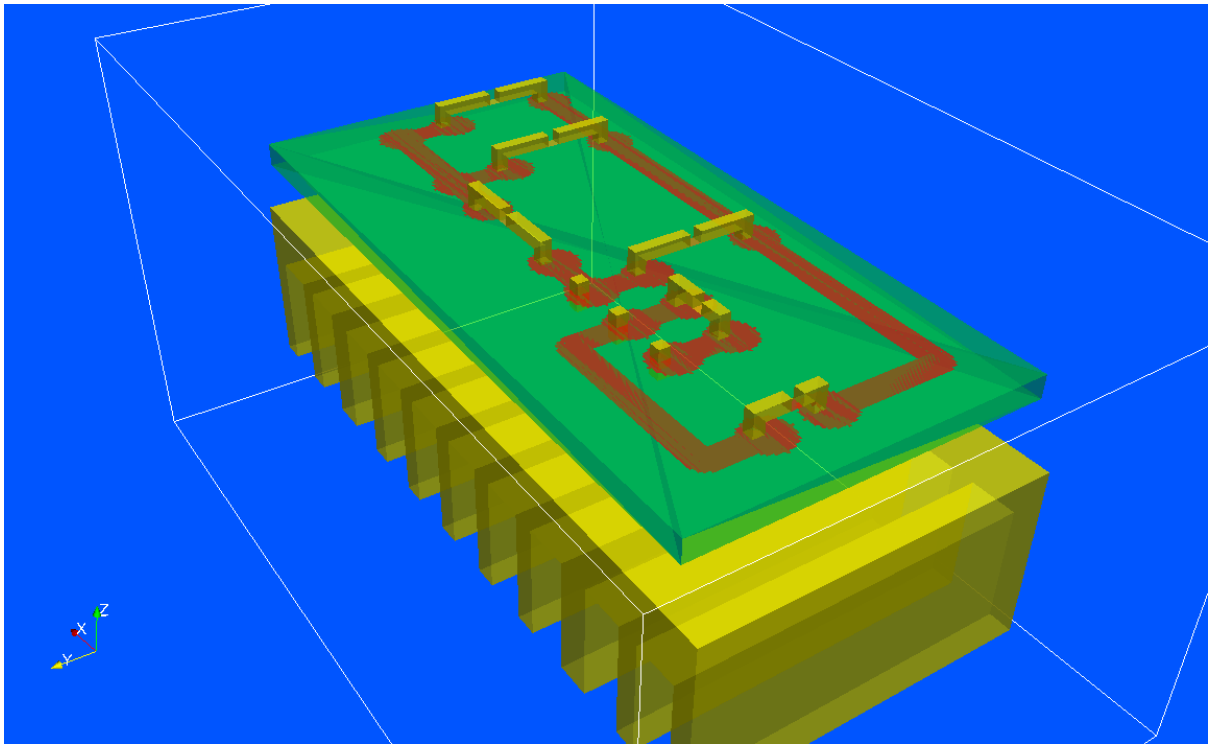

Figure 44 3D GGI_TLM converter model including heatsink

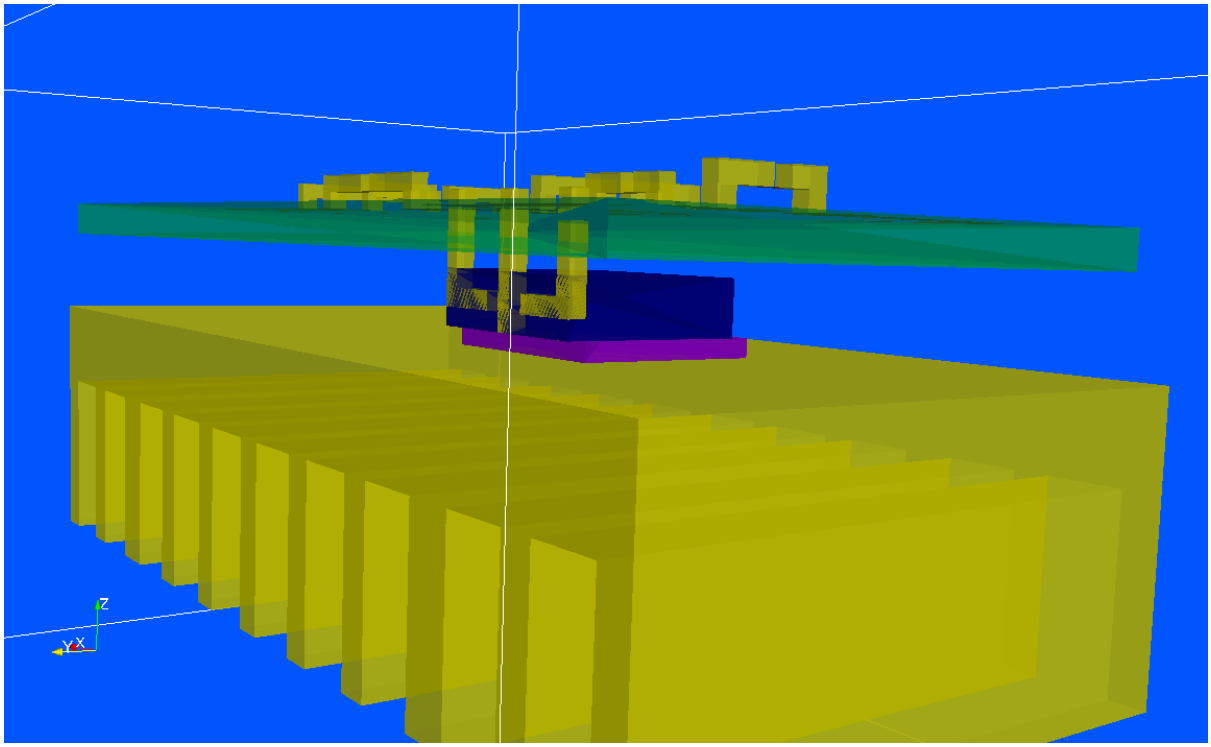

Figure 45 3D GGI_TLM converter model showing the MOSFET package model and ceramic tile between it and the heatsink

Figure 46 shows the converter switch, diode, inductor and load voltages predicted by the GGI_TLM model. The current density on the PCB tracks with the switch closed and open are shown in Figure 47 and Figure 48 respectively. The magnetic field with the switch closed and open are shown in Figure 49 and Figure 50 respectively.

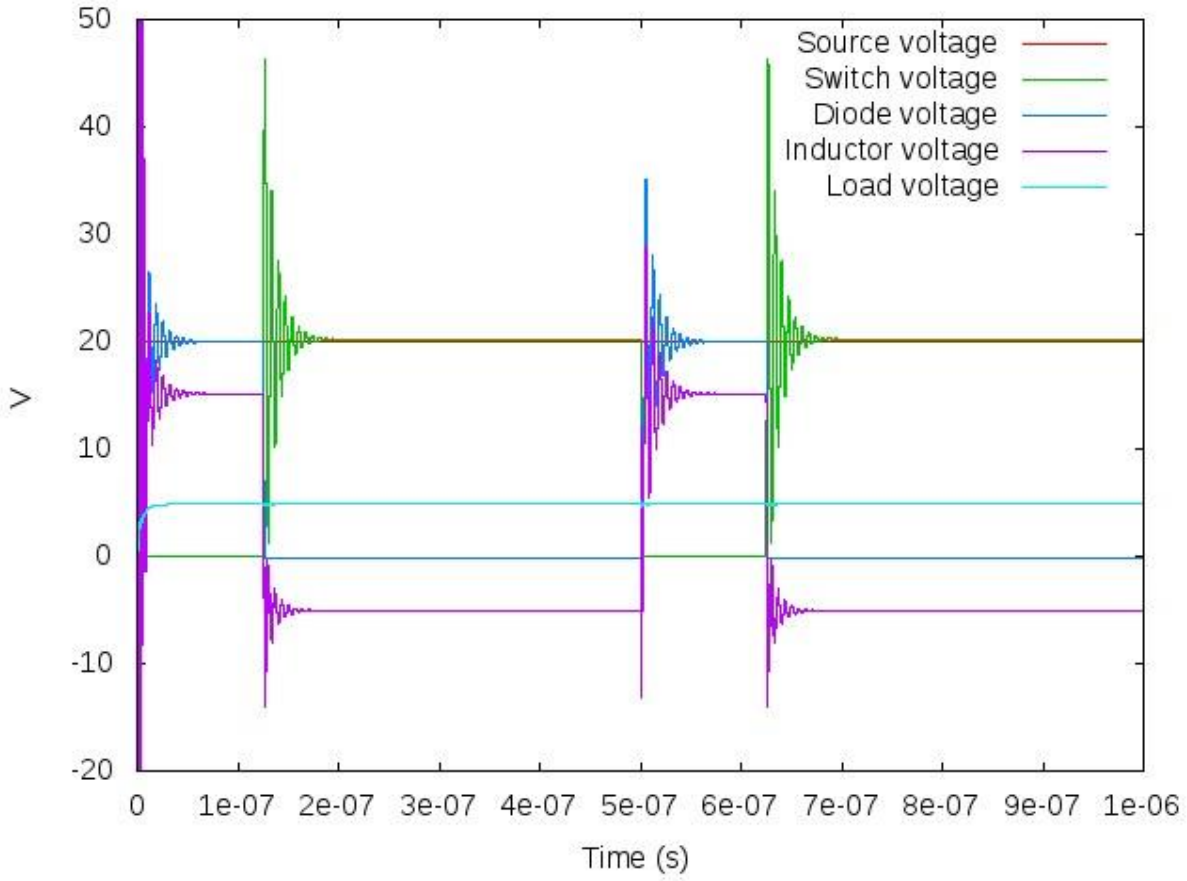

Figure 46. Converter voltages (model includes switch and diode snubbers, operating at 2GHz)

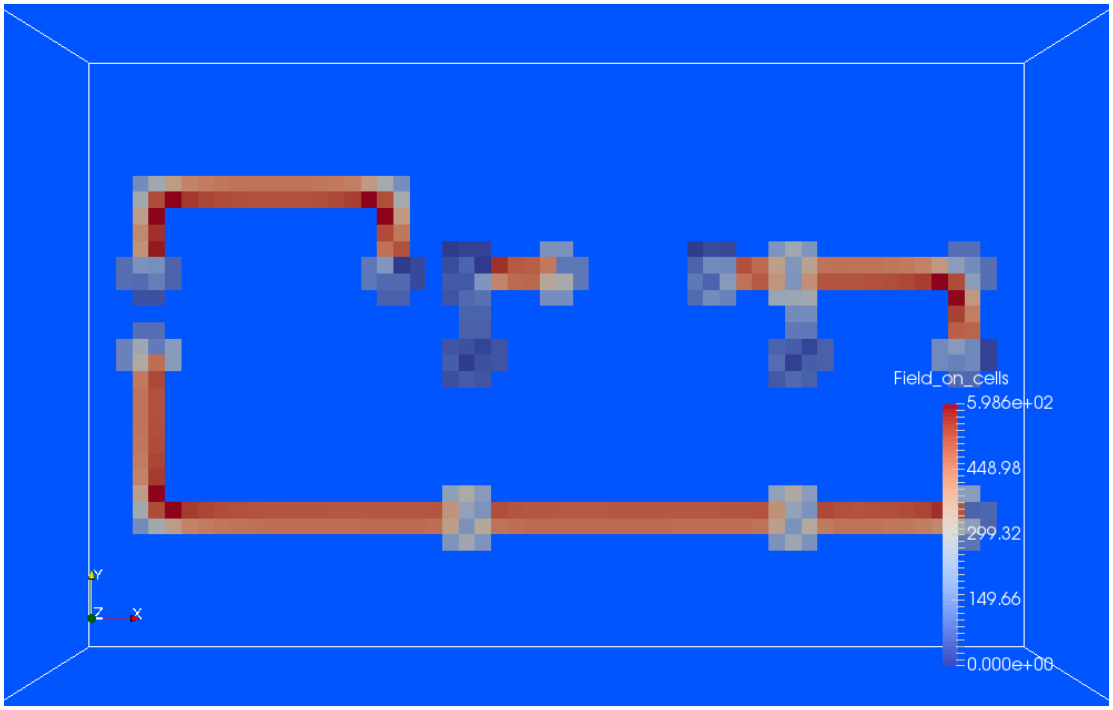

Figure 47. Current density with switch closed

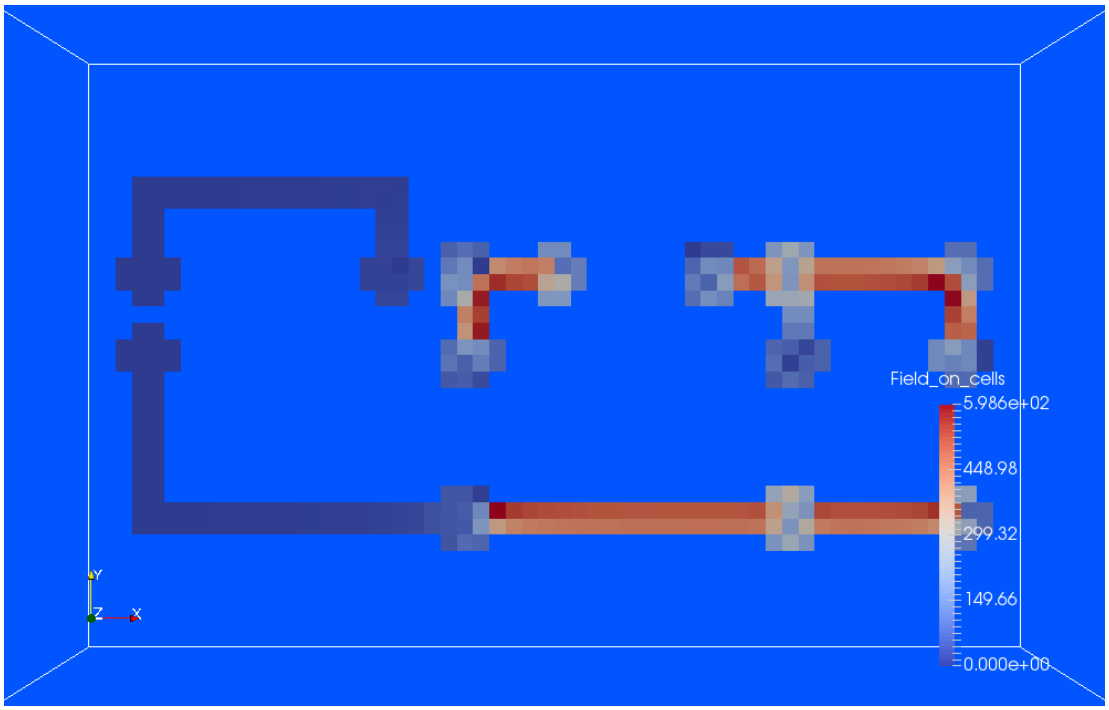

Figure 48. Current density with switch open

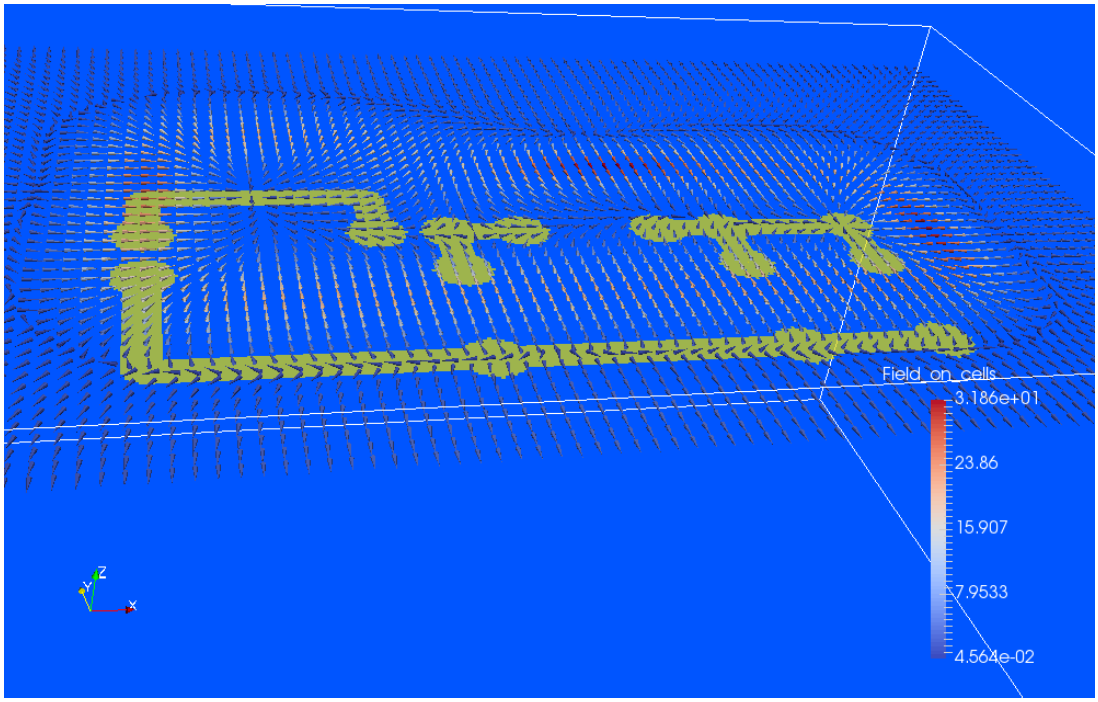

Figure 49. Magnetic field, switch closed

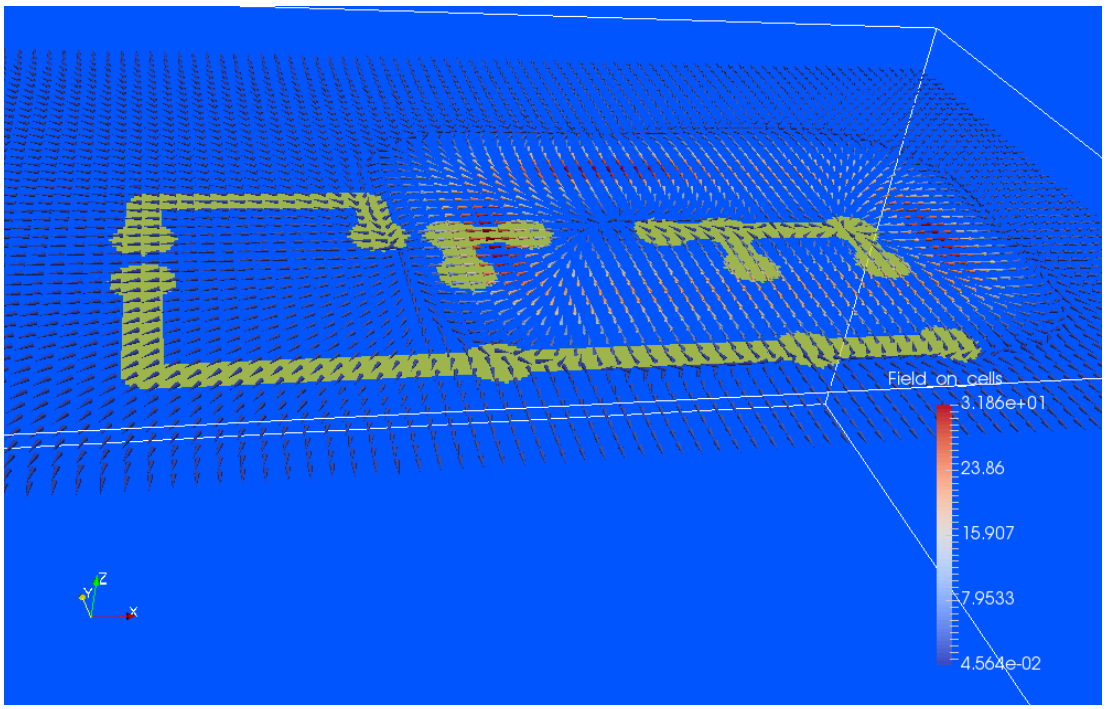

Figure 50. H field, switch open

References

[1] GGI_TLM Open source project, www.github.com/ggiemr/GGI_TLM
[2] C. Christopoulos, "The Transmission-Line Modeling Method: TLM," Piscataway, IEEE Press, 1995
[3] Ngspice, ngspice.sourceforge.net
[4] Gerber format, www.ucamco.com/en/gerber
[5] J.W. Park, P.P.M. So, W. J. R. Hoefer, "Lumped and distributed Device Embedding Techniques in Time Domain TLM Field Models," IEEE MTT
[6] P. P. M. So, W. J. R. Hoefer, "A TLM-SPICE Interconnection Framework for Coupled Field and Circuit Analysis in the Time Domain," IEEE Trans MTT, Vol 50, No 12, December 2002
[7] P.P.M. So, W. J. R. Hoefer, "A General Framework for SPICE-TLM Interconnection," IEEE MTT
[8] H. Du, P. P. M. So, W. J. R. Hoefer, "Embedding Current-Coupled Lumped Networks in TLM Models," IEEE MTT
[9] M. N. de Sousa, L. R. A. X. Menezes, P. Russer, "Hybrid TLM-SPICE Method in Electromagnetics Problems,"