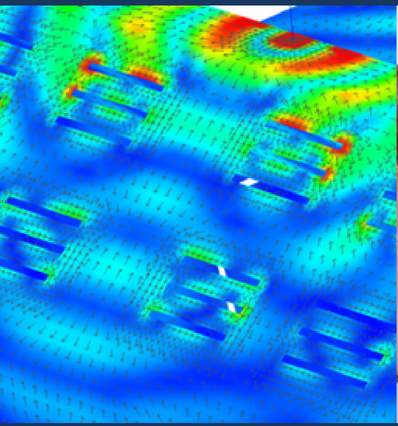
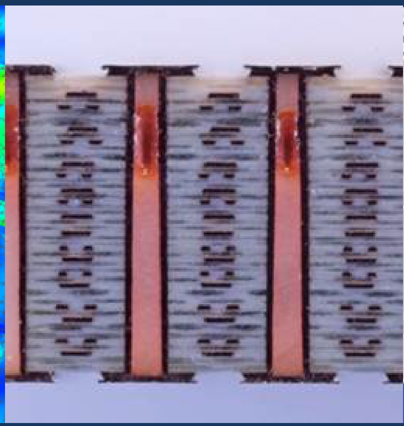
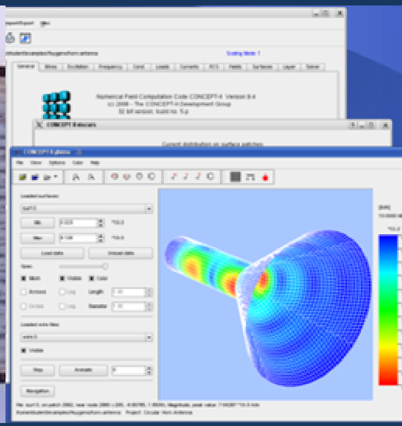
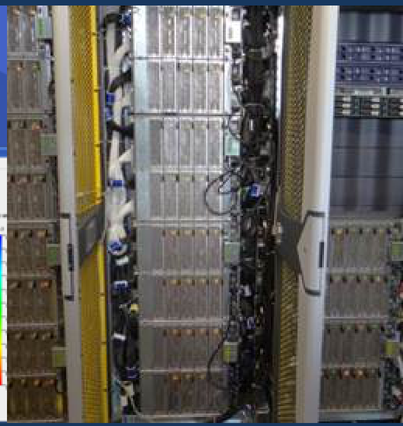





CONMLS

Institut für Theoretische Elektrotechnik

Technische Universität Hamburg (TUHH)

TET

TUHH

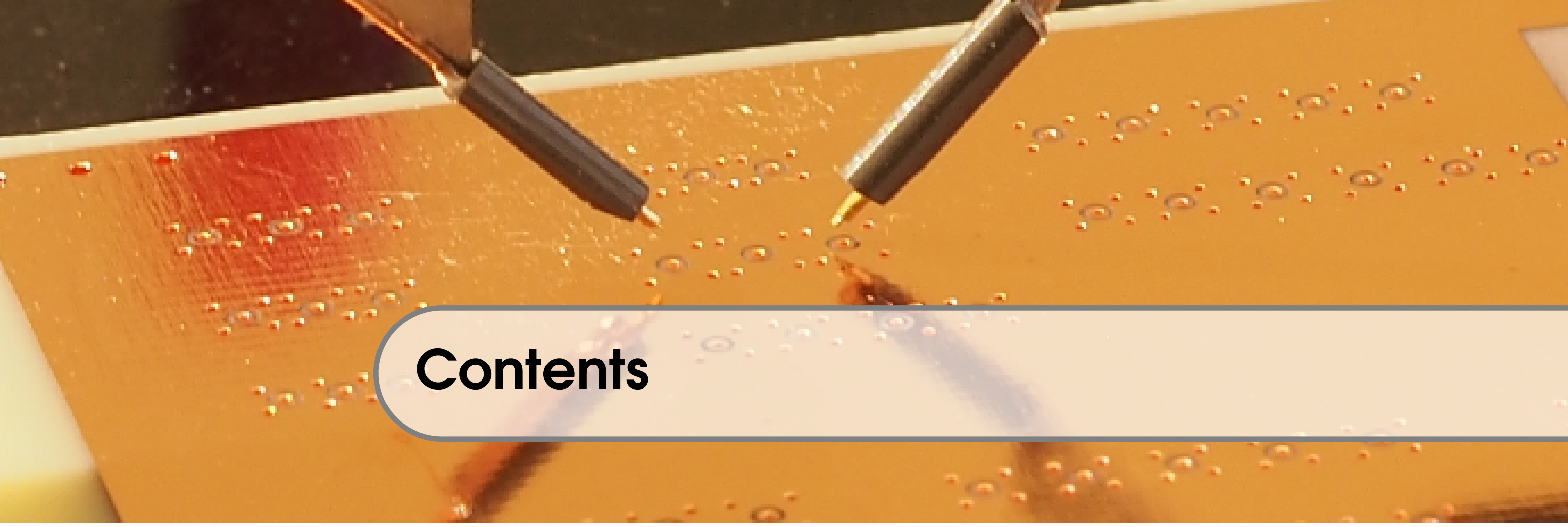
Technische Universität Hamburg

For question and concerns regarding this document, please consult Torsten Reuschel (email: torsten.reuschel@tuhh.de).

WWW.TET.TUHH.DE

This book refers to CONMLS rev1096 with TETlib rev651. Features available in other versions may not be covered. Likewise, not all of the described features may be available in former revisions. The following authors contributed to this documentation: David Dahl, Sebastian Müller, Torsten Reuschel, and Renato Rimolo-Donadio.

Hamburg, Germany, September 6, 2017



Contents

1 — Introduction

The CONMLS code is the improved Fortran [1] implementation of the Matlab [2] Via Pin Simulator (VPF) code [3]. It makes use of the models discussed in [4, 5, 6, 7, 8] to simulate multilayer substrates enclosed by solid reference planes. This document explains the syntax, structure and functionality.

Figure 1.1 illustrates the program functionality and main building blocks. An interpreter reads the input files, which are a high-level description of the structure to be simulated. These files are then decoded. A second code component gets the variables created by the interpreter and identifies the cavities and their related interconnect elements. Then, the calculator computes the parallel-plate impedance per cavity, generates or imports the transmission line models for traces, and calculates or reads the via-to-plane capacitances to approximate the near fields in the antipad region. The calculator combines the plane and trace model by applying modal decomposition, and creates the interconnection matrices for via capacitances and lumped elements. Finally, the partial results are concatenated, for instance, using segmentation techniques. Post-processing functions and utilities are available to store (e.g. as lydite- or touchstone® files [9]) and visualize the results. This version of the program is based on previous code developments [3, 10, 11, 12]. It has been interfaced with a revised CIM package [13] which allows the computation of arbitrary-shaped planes by applying the contour integral method. Another previously external algorithm for the computation of MTL systems with an hybrid 2D approach for arbitrary cross sections [14, 15] is included in the CONMLS code.

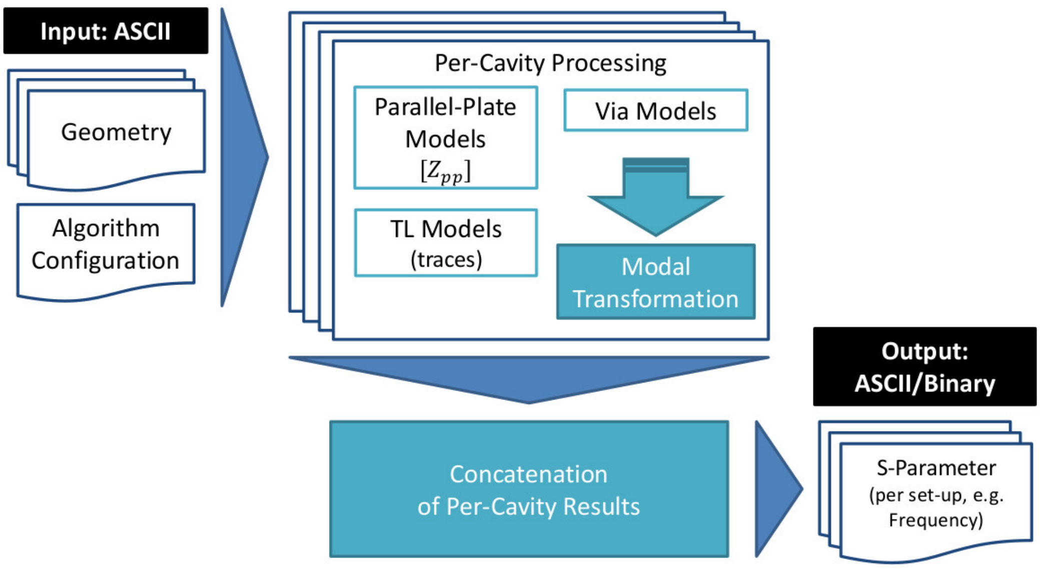


Figure 1.1: Flow chart of input/output and code functionalities in CONMLS.

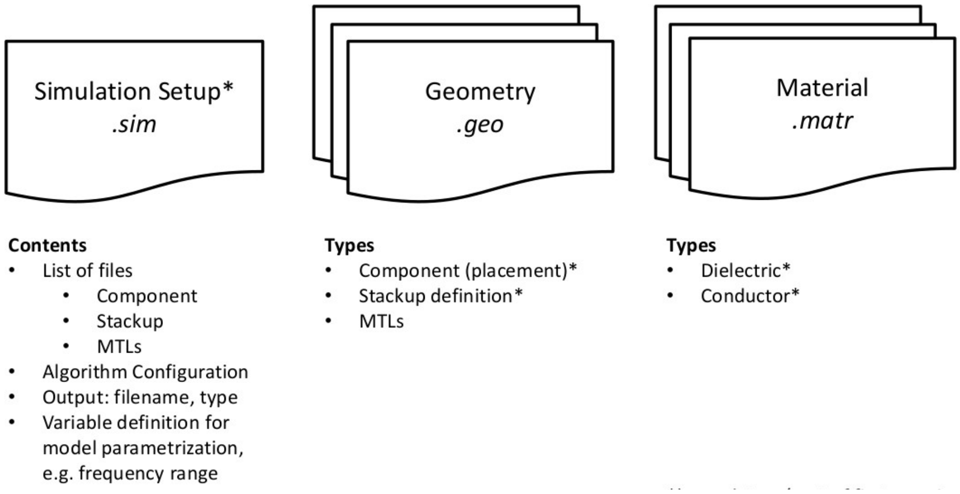


Figure 1.2: Input files for the high-level description of multilayer substrates in CONMLS are based on the widely used INI-format. Asterisks mark mandatory files. Other files may be required depending on the configuration of the algorithm.

Figure 1.2 shows the required input files and their dependencies. The fundamental configuration is provided through the simulation setup (.sim-file). The choice of algorithms and their respective parameters is contained in this file. Moreover, models may be freely parameterized with regard to variables and parameter sweeps, all of which are to be defined in the simulation setup. Additional files such as the geometry specification are linked. Geometry definition of PCB layout, stackup, vias, and multi-conductor transmission lines (MTLs) are to be provided through .geo-files. These files may utilize physical material definitions that are located in .matr-files.

Notation

Parts of this user guide are enriched by one or more of the following elements:

R Remarks provide additional information for advanced users.

! Notes give hint to specifics in configuration and behavior of CONMLS.

Short examples of code and syntax are provided in the following way:

```
mlss.2.exe --help
```

2 — Installation

The CONMLS code is available for Linux and Windows operating systems.

2.1 Windows

The following guide describes the installation process of CONMLS under MS Windows 64-bit and the steps required to get started:

2.1.1 Installation of the CONCEPT-II MLS module under Windows operating systems

1. Run *conmls-[xx]-setup.exe* to install the CONMLS binaries, where *[xx]* denotes the version.
2. Extend the system path by `C:\CONMLS-1.0\bin` or wherever the package has been installed, here referred to as $CONMLS.

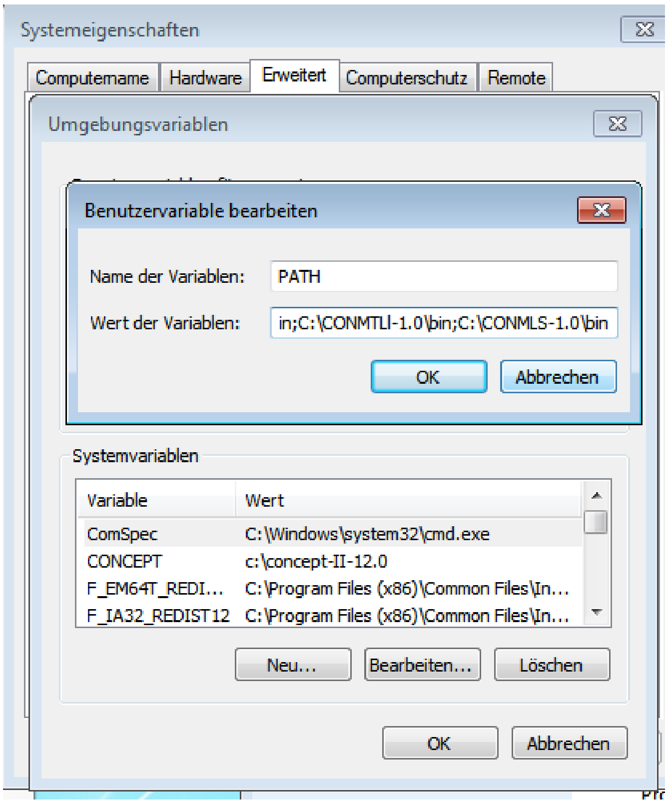


Figure 2.1: Setting the path to the CONMLS executables (System variables/Name of variables).

To do so, open the *System* applet in the *Control panel* (Systemsteuerung). Choose the register *Advanced* (Erweitert). Click on *Environment variables* (Umgebungsvariable).

Select the entry *Path* of *System variables* (Systemvariablen) and press *Edit* (Bearbeiten). A window according to Fig. 2.1 opens where *Path* can be completed. Note: *Path* has to be extended by a semicolon followed by the CONMLS path. For details see Fig. 2.1.

The installation works only properly if all programs that are contained in the directory $CONMLS/bin can be called without specifying the complete path. Please do not execute programs under $CONMLS/bin. Copying of executables is not necessary and should not be done!

2.1.2 Installation of auxiliary software

Gnuplot is necessary for the graphical representation of curves. To install it, download the binaries from `http://www.gnuplot.info`.

CONMLS uses the program gnuplot.exe. Please extend the variable PATH by the corresponding path, e.g., *C:\Programs\gnuplot\bin* or wherever gnuplot.exe is installed. The program must be callable from any working directory.

2.1.3 Test of the installation

Using a command line window, the following two steps can be used to validate that the path variables and binaries are configured and installed properly.

(a) Type conmls.exe <Enter>. The GUI of CONMLS appears. If not, the Path has not been specified properly.

(b) Type gnuplot.exe <Enter> at the command line. The gnuplot window appears. If not, the Path has not been specified properly.

3 — Getting Started

This chapter gives some aid to those who are new to the CONMLS code.

3.1 Graphical User Interface

Starting the graphical user interface CONMLS for the first time, the various design views and monitors are empty as shown in Fig. 3.1.

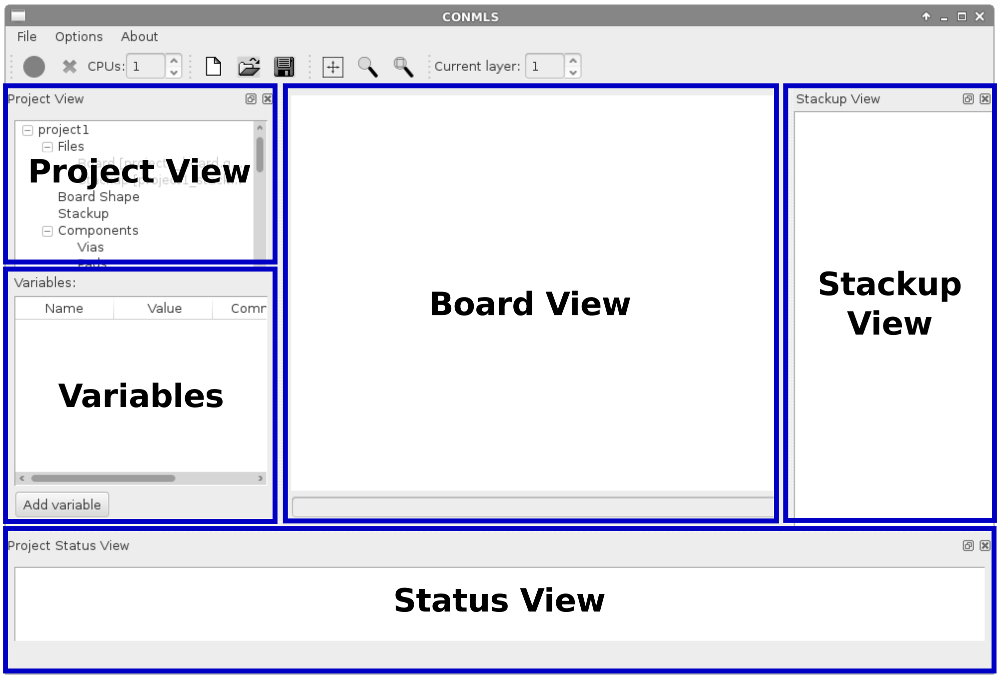


Figure 3.1: Graphical user interface CONMLS after first startup.

The depicted views and monitors are:

Project View A tree-structured list of all components, models, and configurations.

Board View A bird's view of the PCB.

Stackup View The stackup of the PCB.

Variables A list of variables used in the current project. Variables can be utilized for parameter sweeping.

Status View The output of the backend is piped to this monitor.

Toolbar Direct access to repeatedly used functions such as loading and saving a project as well as running a simulation.

The following sections provide details on functionality and usage of the graphical user interface based on specific examples.

3.1.1 Example 1: Via-to-Via Link

The first example features two vias that are interconnected by a transmission line on layer *signal1*. One of the vias ranges through the entire stackup and features a via stub. The second via is back-drilled and ends on layer *signal1*.

Loading a previously stored project can be done using the *File* menu or using the *Open file..* button in the toolbar, cf. Fig. 3.2.

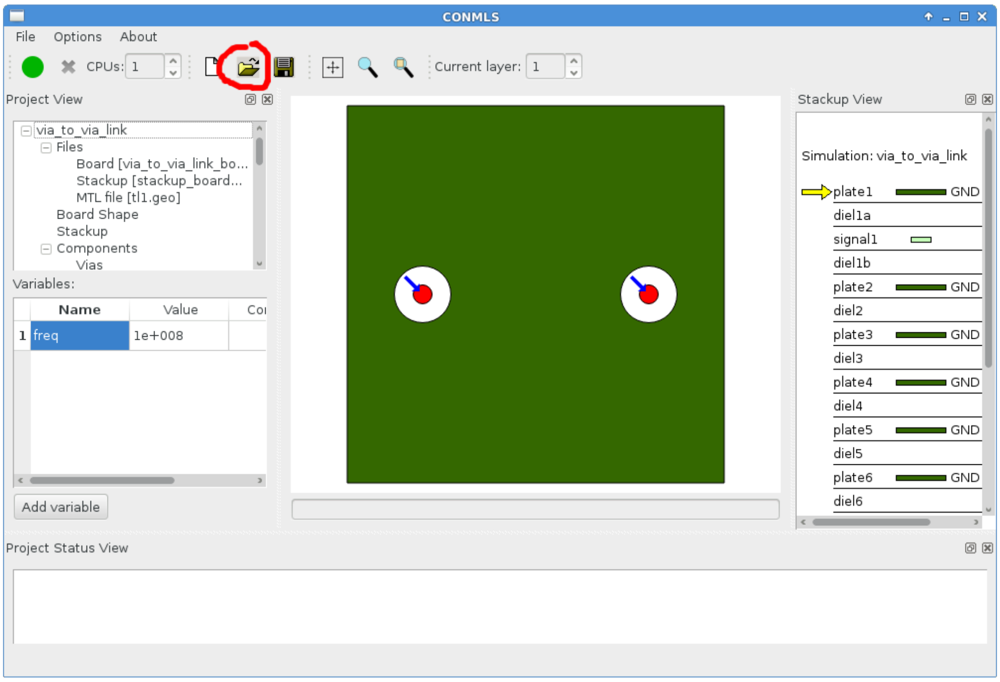


Figure 3.2: Graphical user interface CONMLS after loading example *Via-to-Via Link*.

All properties and settings of the project are accessible through the project tree on the left of the screen. Typically, materials need imported or defined prior to setting up the stackup, cf. Fig. 3.3. The shape of the board and the layers of the stackup can be configured as depicted in Figs. 3.4 and 3.5. Next, the via models can be defined as shown in Fig. 3.6. Once defined, vias can be placed as shown in Fig. 3.7.

Note how the via *via1* ranges from top to bottom of the board. The graphical preview of the geometry offers two more perspectives. Clicking on the white antipad area of a via opens the cross-sectional view as shown in Fig. 3.8. Second, the currently displayed layer can be selected in the stackup view on the right, cf. Fig. 3.9. Selecting the layer plate2 reveals how only *via1* pinches through this metallic layer.

Ideal transmission lines as well as generic multiconductor transmission lines can be used to interconnect vias. Figures 3.10 and 3.11 depict the latter. Again, selecting the corresponding layer, here: *signal2*, reveals the routing of transmission lines, cf. Fig. 3.12.

Once the geometry is set up, the algorithms, models, and boundary conditions that are to be used for the simulation can be configured as depicted in Fig. 3.13. Sweeping parameters such as the frequency can be selected from the project tree, cf. Fig. 3.14. Finally, ports should be placed as shown in Fig. 3.15 in order to probe the geometry which is to be simulated.

Finally, the simulation can be started after selecting the number of processors to be used.

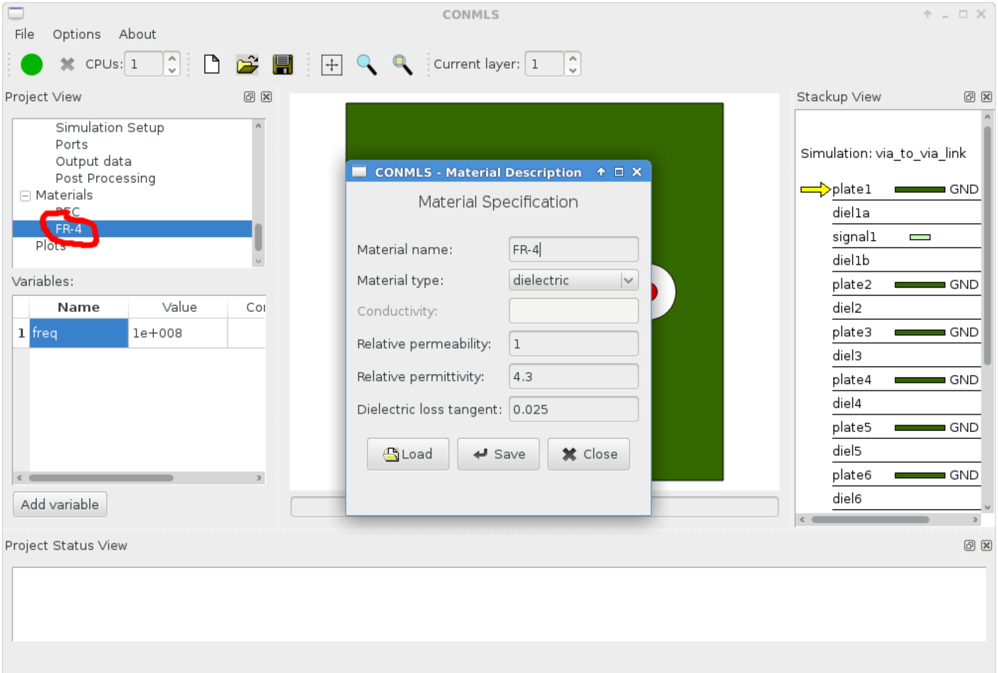


Figure 3.3: Editing a material definition.

CONMLS offers parallelization based on OpenMP (shared memory parallelization). Completion of the simulation is indicated by a blue *Program finished!* at the bottom of the project status view. For quick sanity checks of the results, the graphical user interface offers selecting input and output ports and viewing the magnitude of their respective transfer function, cf. Figs. 3.17 through 3.19.

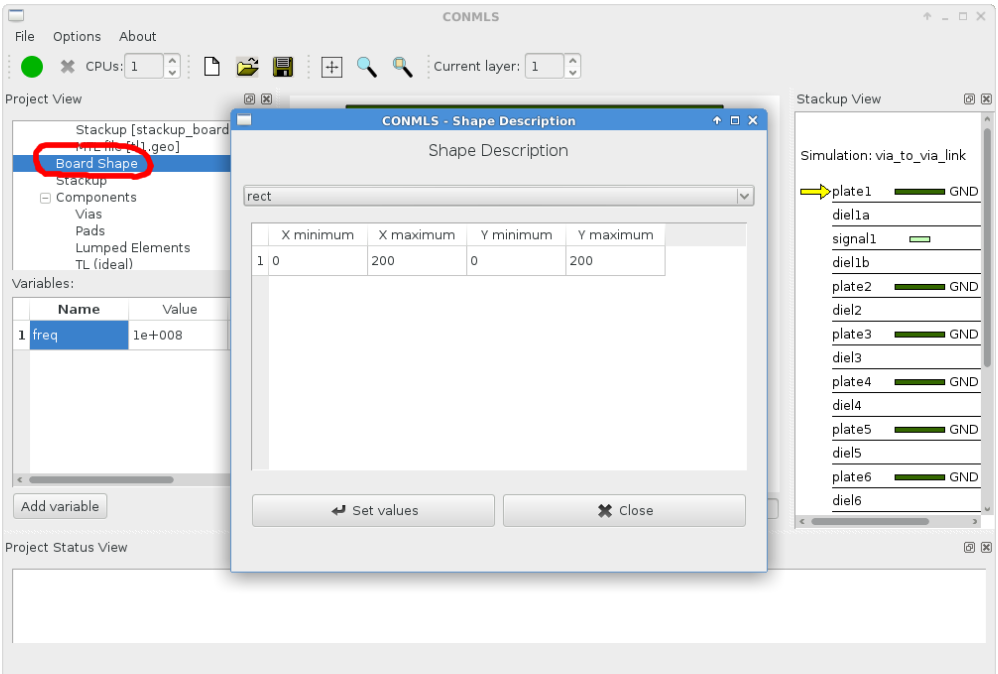


Figure 3.4: Editing the shape of the printed circuit board.

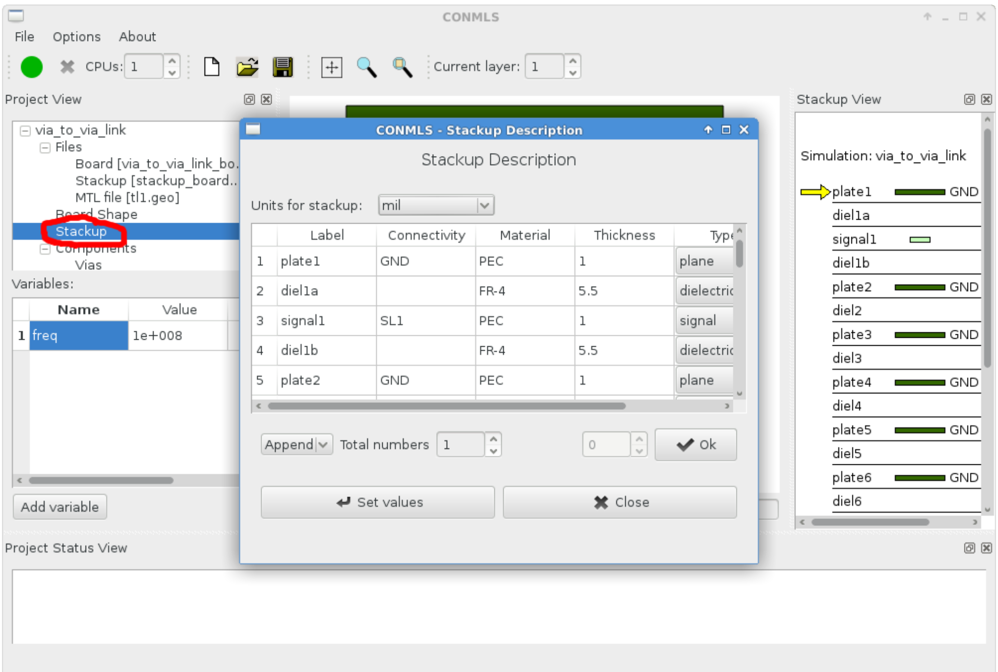


Figure 3.5: Editing the layers of the stackup.

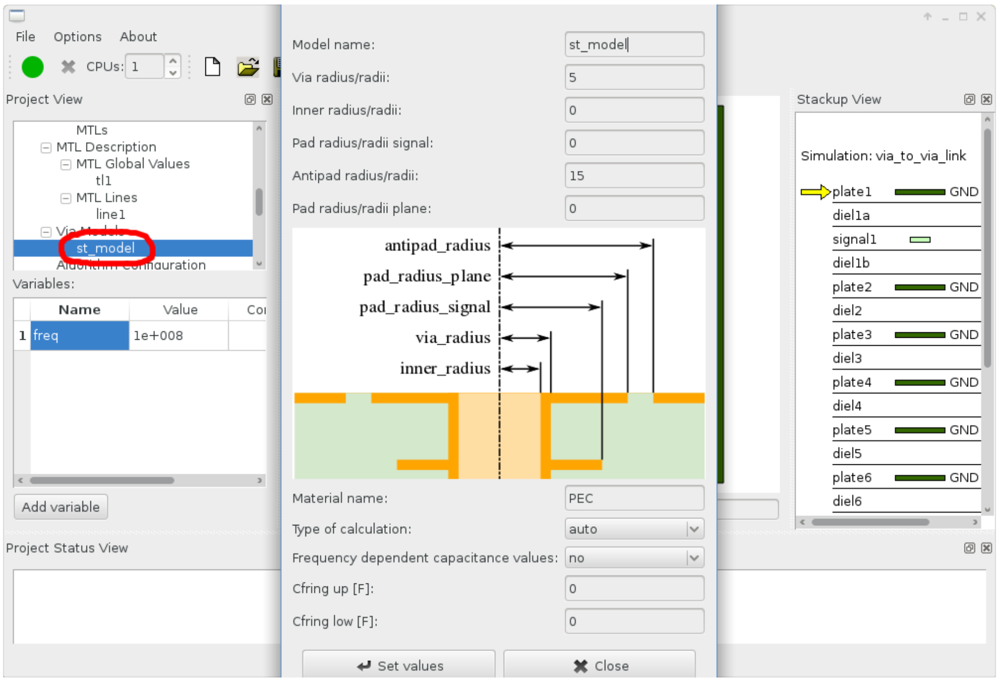


Figure 3.6: Definition of a via model.

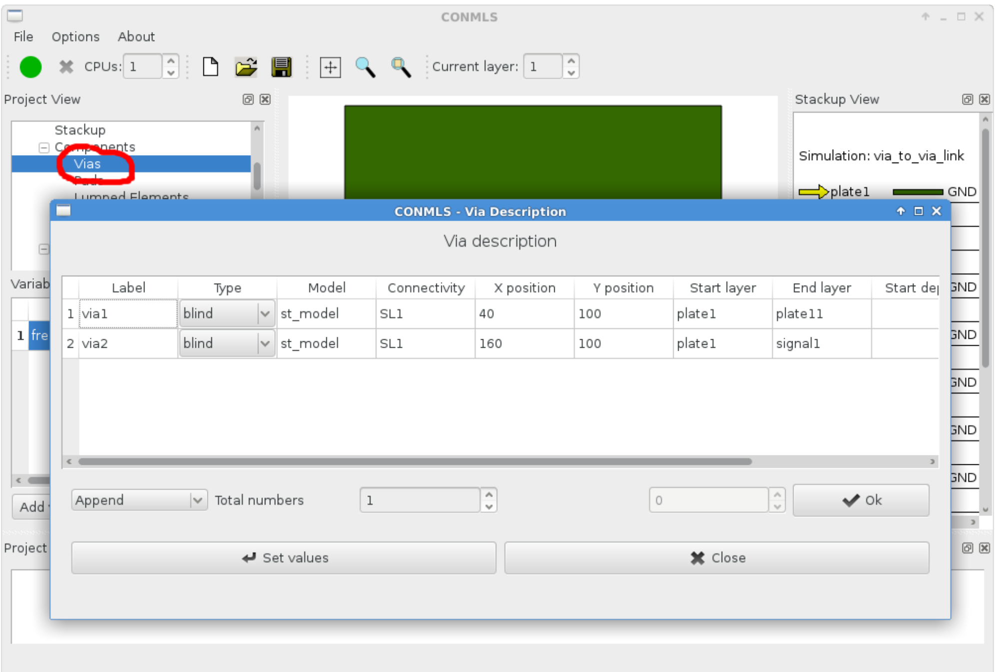


Figure 3.7: Placement of vias.

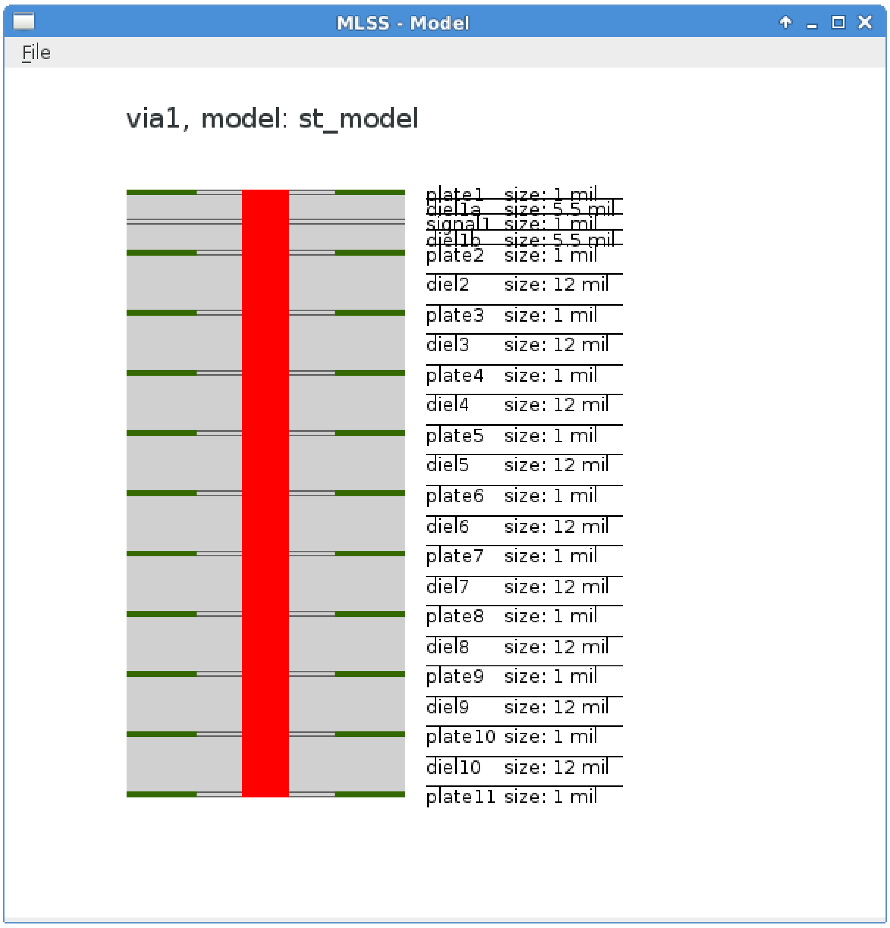


Figure 3.8: Cross-sectional view of via *via1*.

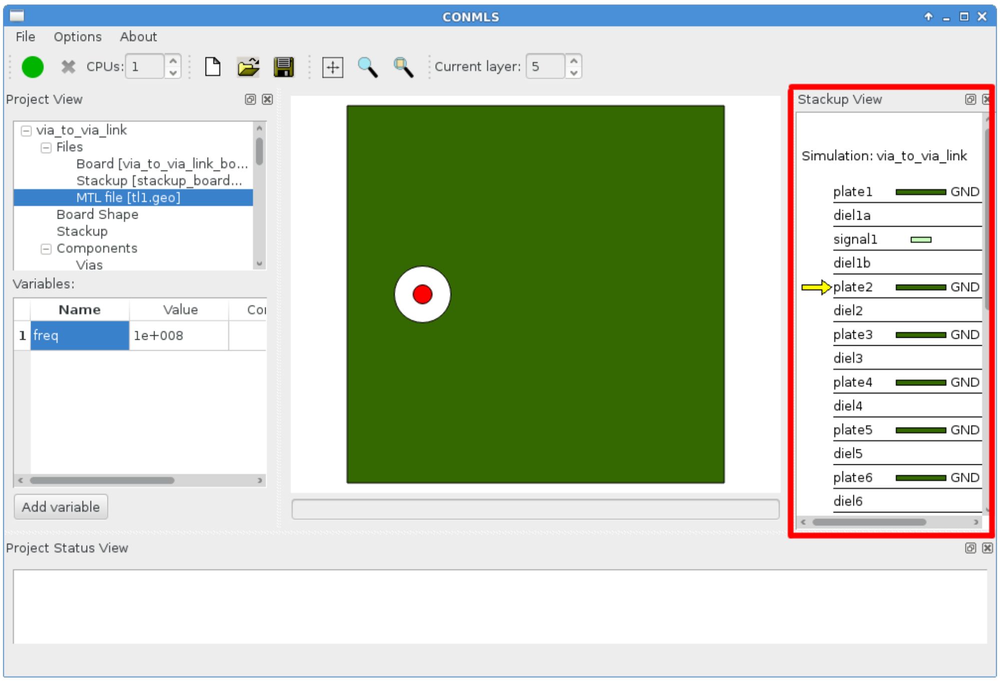


Figure 3.9: Selecting and viewing the layout of layers in the stackup.

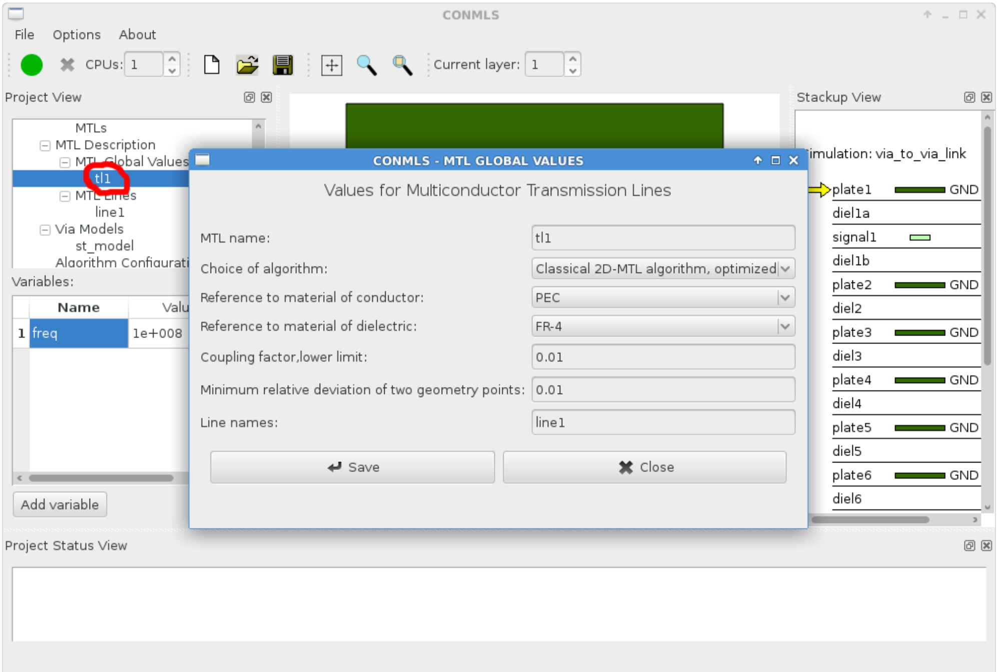


Figure 3.10: Definition of a multiconductor transmission line (MTL) object.

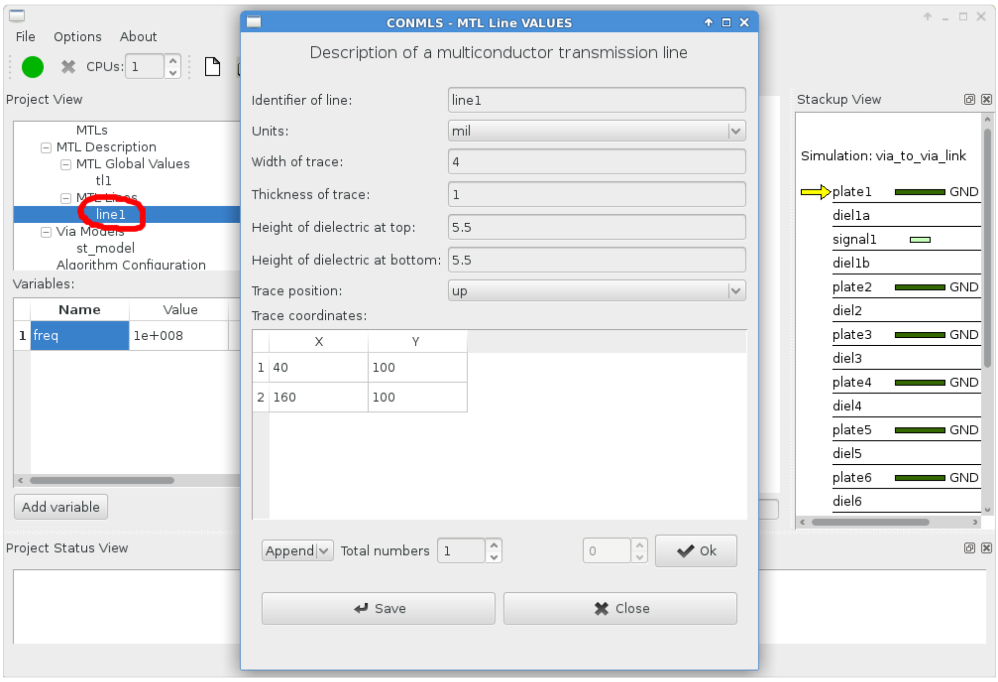


Figure 3.11: Editing the traces ('lines') of a multiconductor transmission line object.

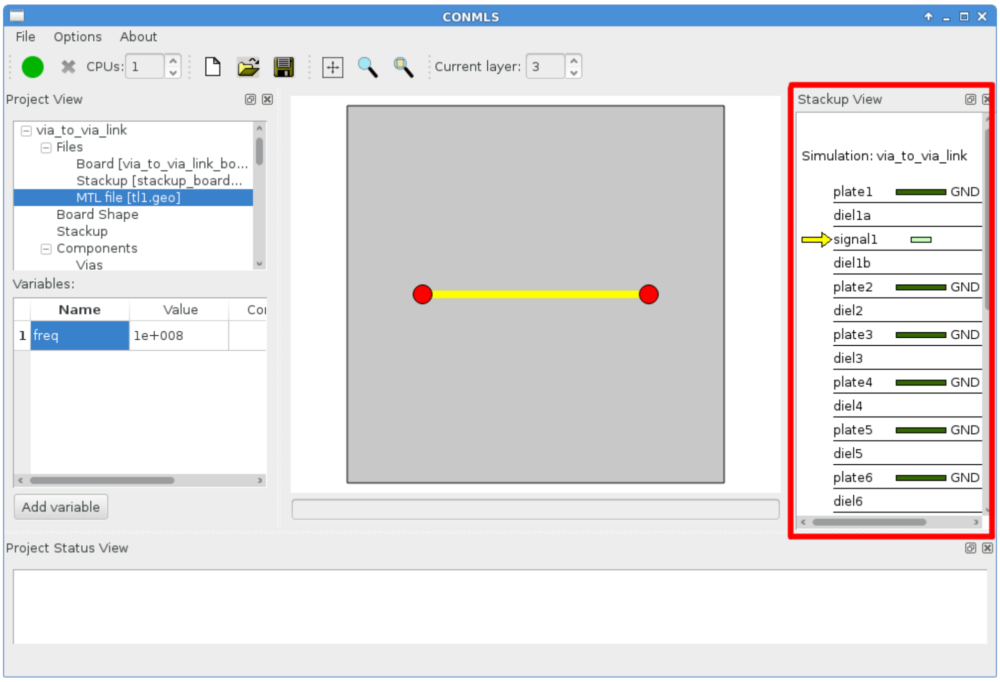


Figure 3.12: Signal layers can be selected from the stackup view to view the routing of traces.

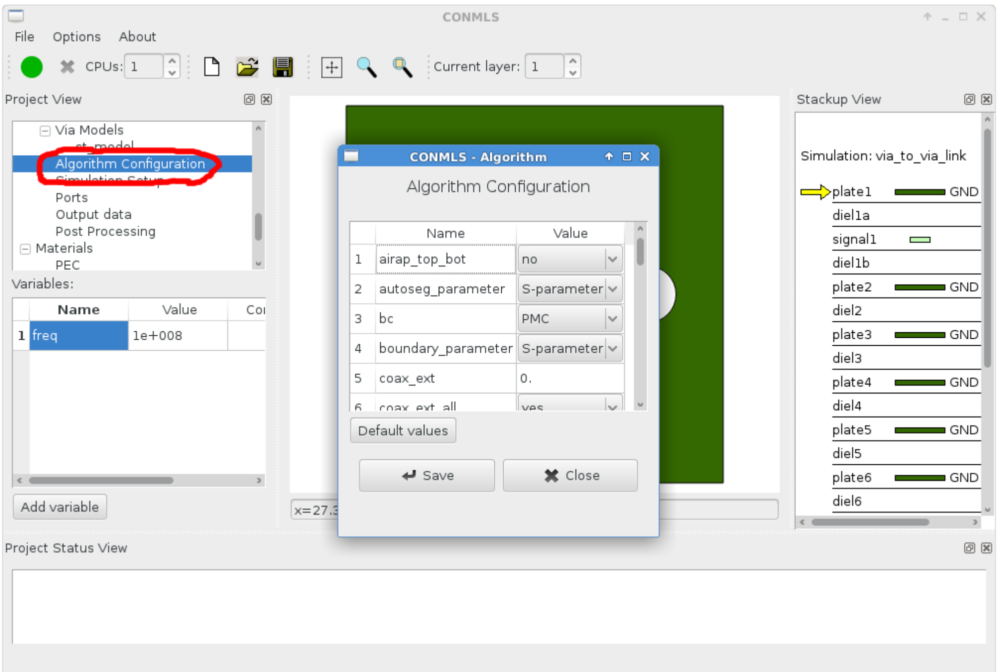


Figure 3.13: Selecting and configuring algorithms, e.g., semi-analytical models.

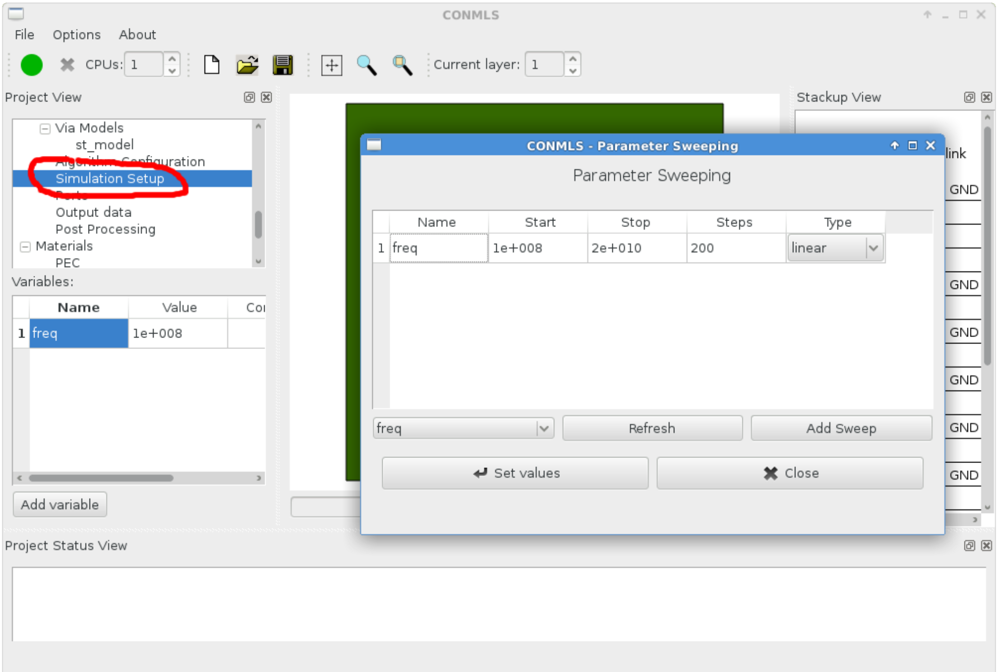


Figure 3.14: Definition of parameter sweeps.

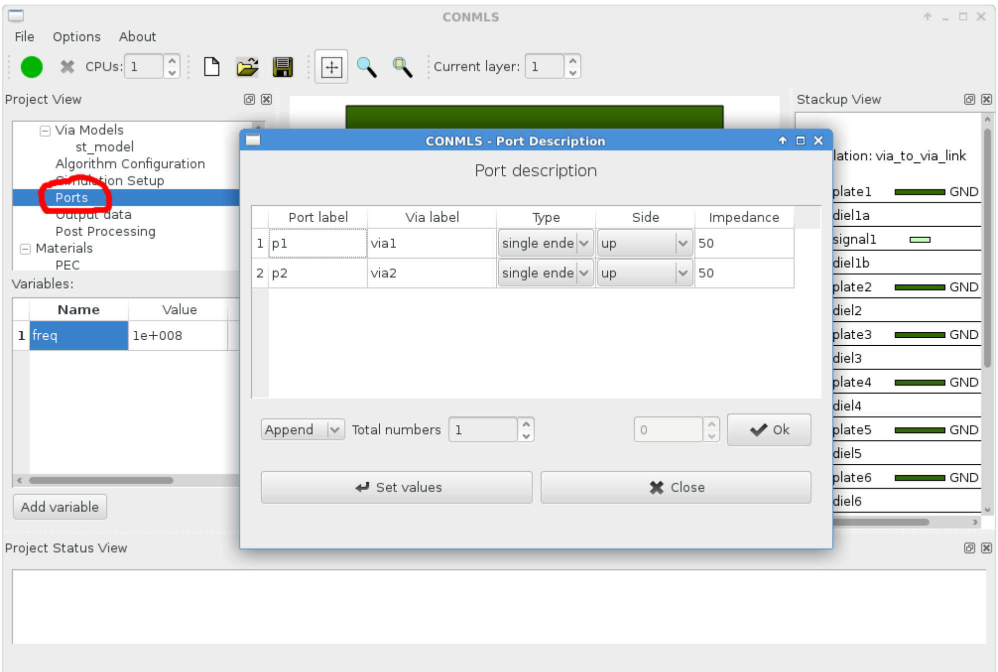


Figure 3.15: Definition of ports, e.g., via ports and stripline ports.

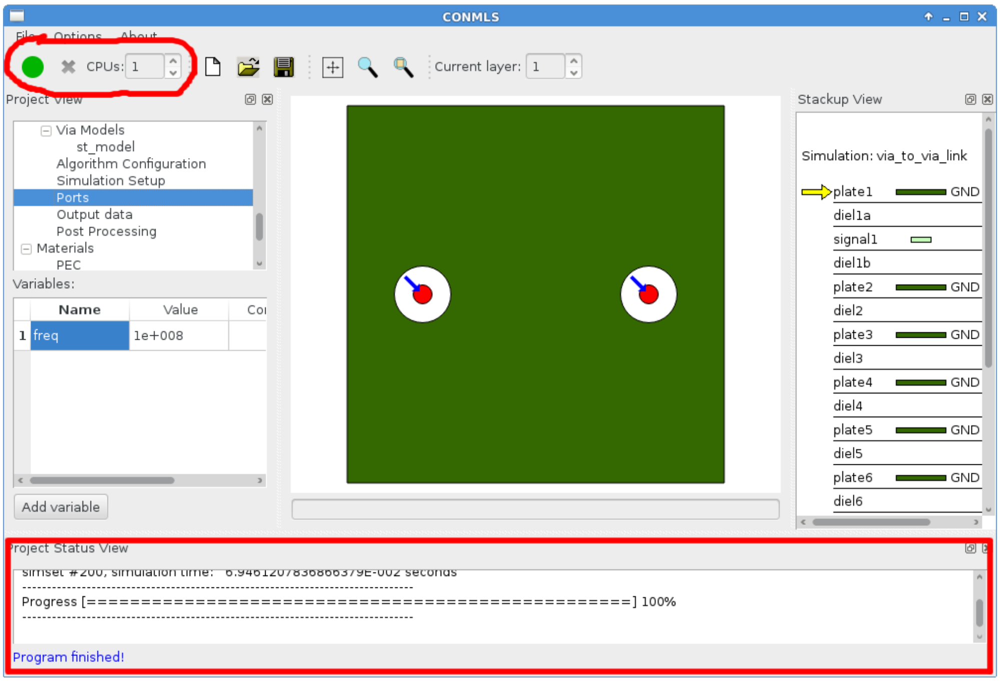


Figure 3.16: Setting the number of processors to be used (shared memory parallelization).

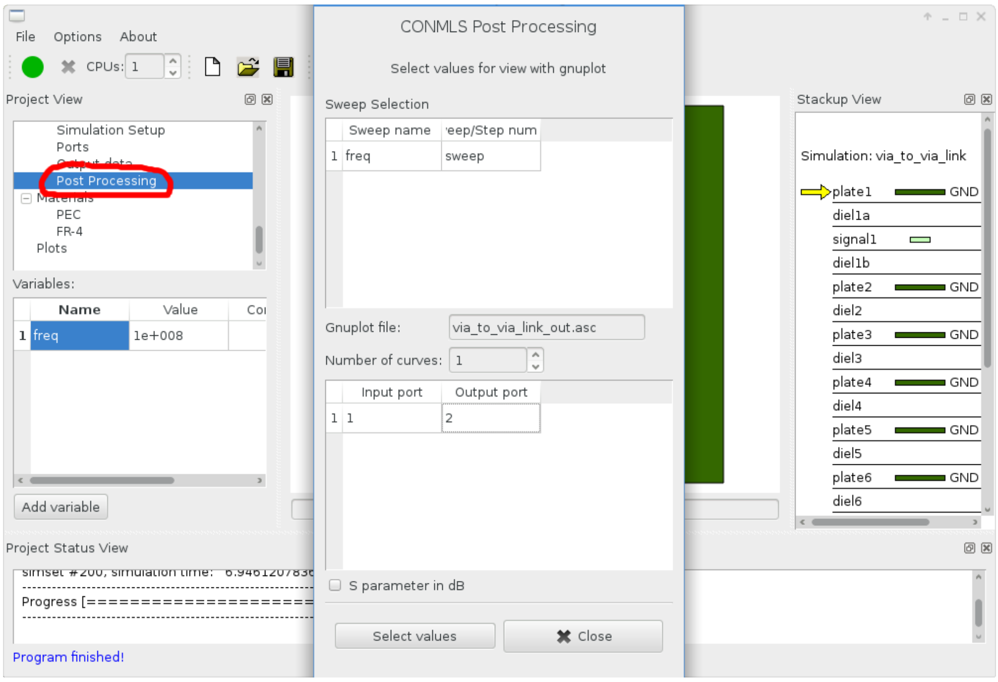


Figure 3.17: Selection of input and output ports for plotting.

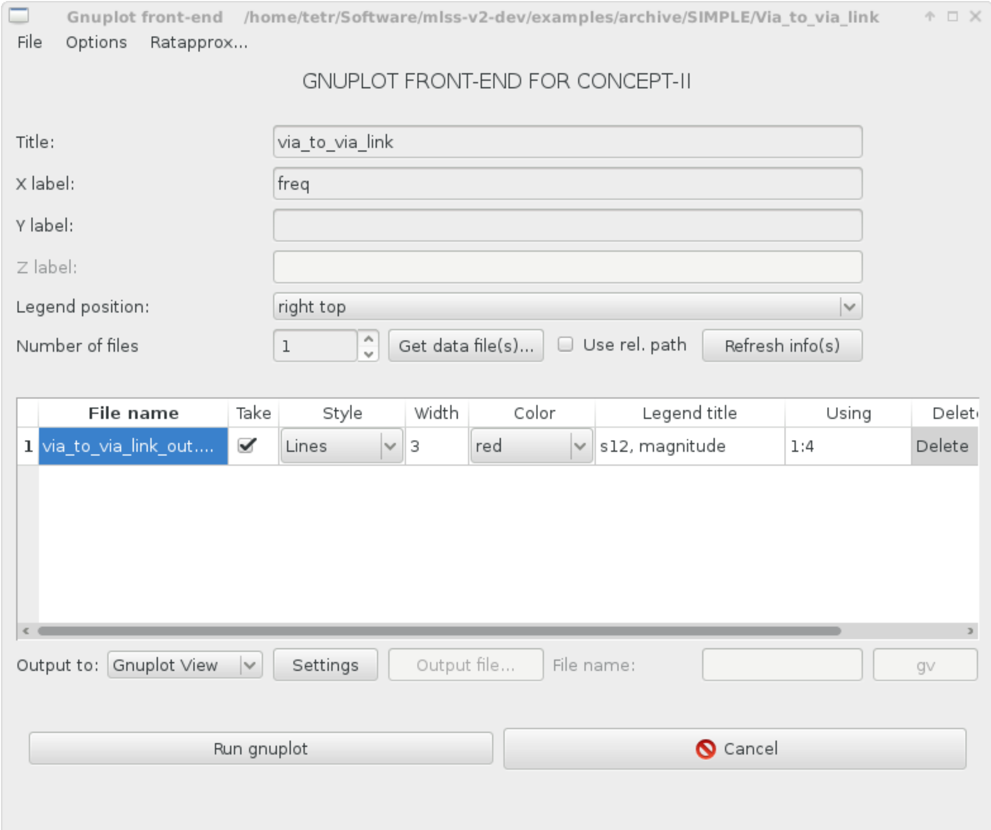


Figure 3.18: The plotting interface. Data may be imported from CONMLS as well as external files. Plots such as Fig. 3.19 can be displayed by clicking on *Run gnuplot*.

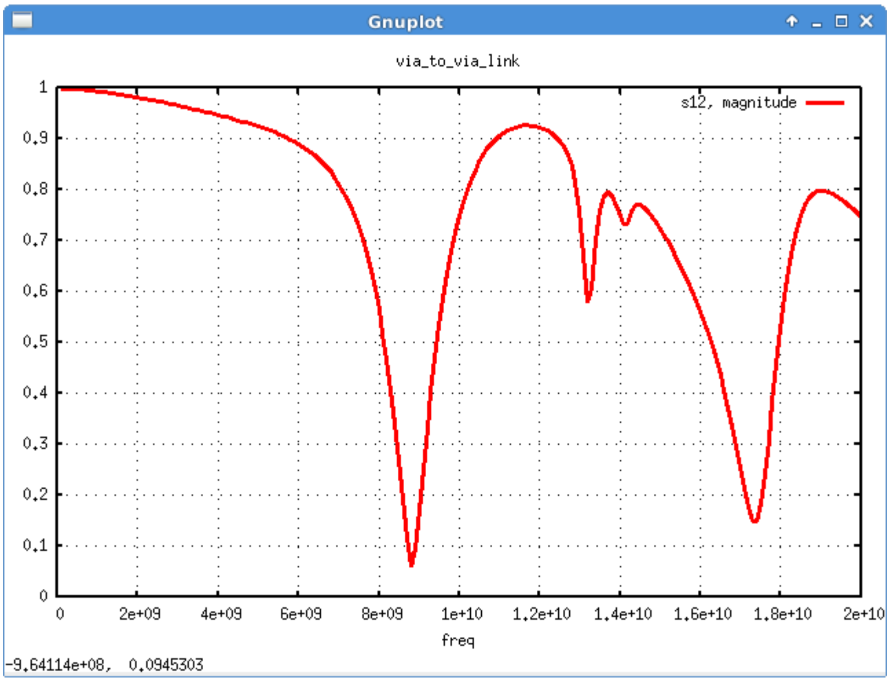


Figure 3.19: Transmission of the via-to-via link. The stub causes resonances near 9 GHz and 18 GHz.

3.1.2 Example 2: Multilayer Thru Vias

This example features two adjacent vias and can be used to investigate the crosstalk behavior in a multilayer board environment.

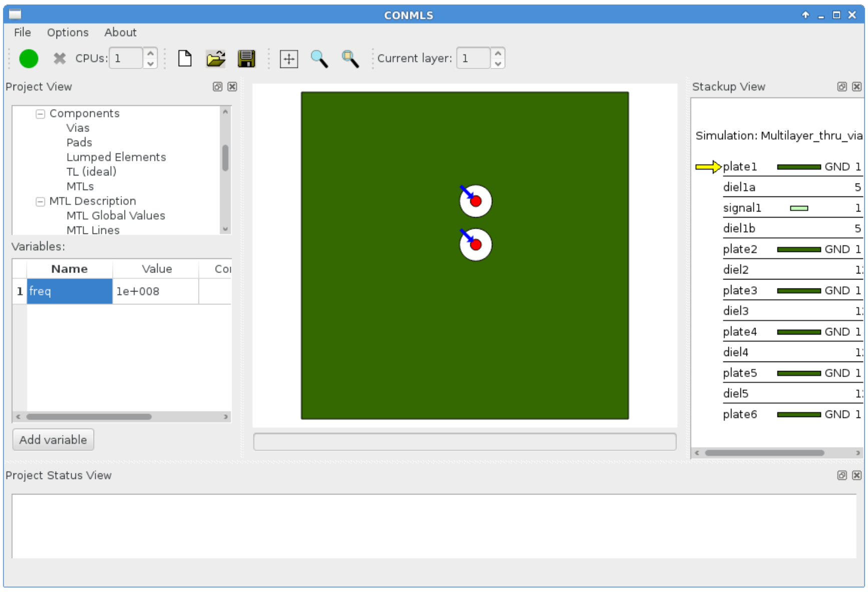


Figure 3.20: Main window view of demo scenario 2: two adjacent vias in a multilayer environment.

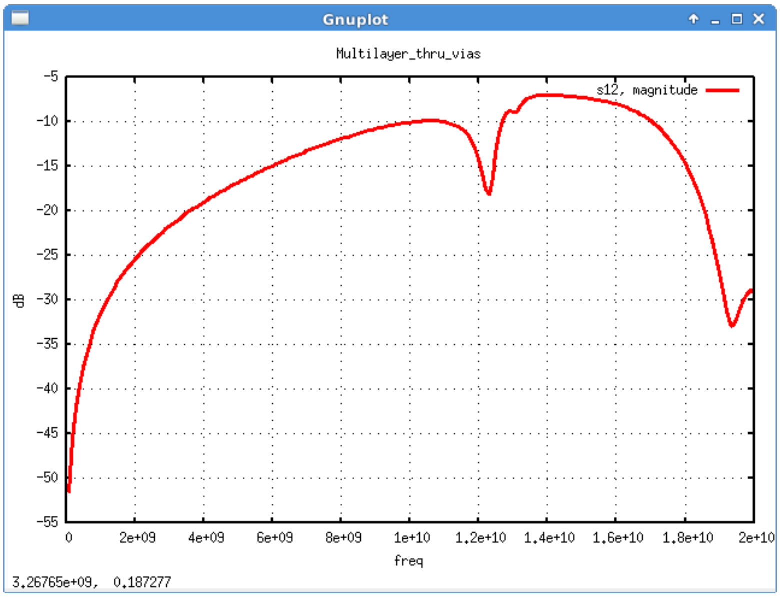


Figure 3.21: Crosstalk of two adjacent vias in a multilayer environment.

3.1.3 Example 3: Pin Field with 64 Vias

To demonstrate the efficiency of the CONMLS code, this example features a via pin field with 64 vias. On a standard office computer, this

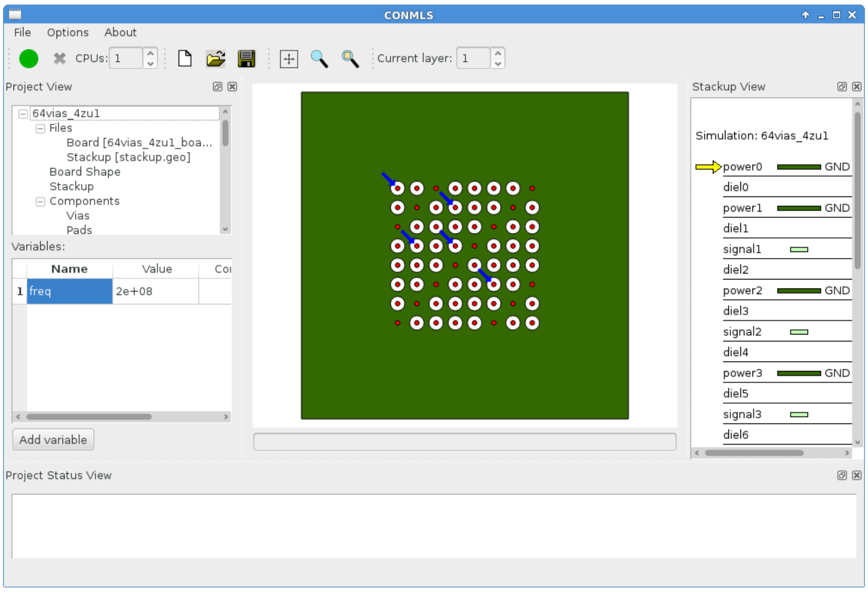


Figure 3.22: Main window view of demo scenario 3: a pin field with 64 vias.

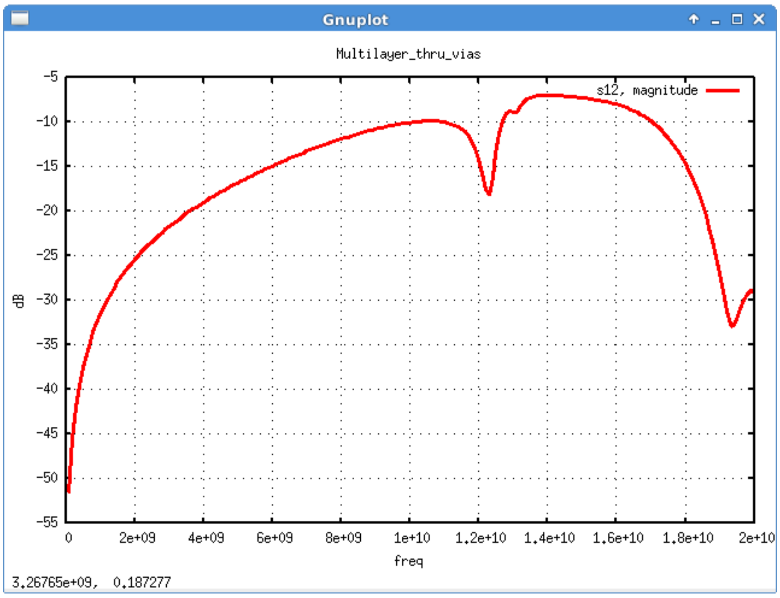


Figure 3.23: Crosstalk between the edge via and a central via of the pin field.

3.2 Running from Command Line

CONMLS allows for parallelization of computations based on *OpenMP*, a shared memory parallelization scheme. To utilize it, run CONMLS using the command

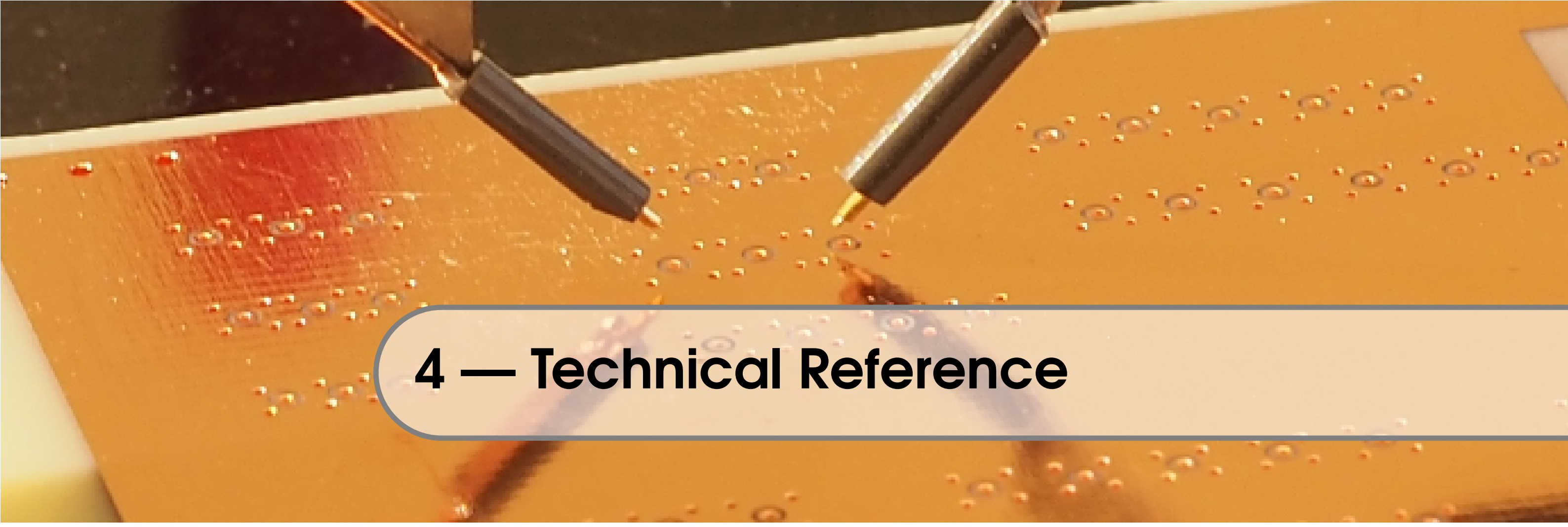
```
OMP_NUM_THREADS=[num] mlss.2 path/to/config.sim
```

where *num* sets the number of parallel threads during runtime.
A corresponding input argument is available to set the desired number of threads:

```
mlss.2 -n [NUM_THREADS] path/to/config.sim
```

Note that this argument will override the conventional setting through the environment variable *OMP_NUM_THREADS*.

> If parallelization is to be disabled, CONMLS needs to be explicitly configured to use only one process, e.g.
>
> ```
> OMP_NUM_THREADS=1 mlss.2 path/to/config.sim
> ```
>
> or
>
> ```
> mlss.2 -n 1 path/to/config.sim
> ```



4 — Technical Reference

This chapter introduces the syntax used to configure CONMLS and build models. All files are based on the INI-format implementation available from *TETlib*. The INI-format arranges *attributes* and their respective *values* by means of *sections*:

```
[section]
#comment
attribute = value
```

While the maximum line length is limited to 1024 characters as of TETlib rev589, whitespaces are automatically discarded and can be used for formatting purposes. Nevertheless, tabs are known to cause issues during processing by the Fortran code and should be avoided.

The input of CONMLS is structured into a algorithm configuration (.sim) and a physical description of the problem consisting of geometry (.geo) and material (.matr) definition, cf. Fig. 1.1. The details and contents of these files are discussed in the following.

4.1 Simulation Setup

The simulation setup file (file-extension .sim) is the principal file which is processed by CONMLS. Selection and configuration of algorithms as well as other global runtime options is achieved through this file. If an option is not defined, the program will used a default value. A warning is be issued and printed in the simulation log regarding the selection of a default value. The general structure of a simulation setup is given in Listing 4.1.

The identifier in the *global* section is required for ease of clarity as this type of file may be processed by other software packages that are being developed at the Institute of Electromagnetic Theory. The other sections are (in alphabetical order):

cim_config This section is specific to the CIM [16] algorithm.

files Referencing of other project files, e.g. geometry definitions. One section only.

global System commands, to be used for control of software environment only. One section.

output Configuration of desired output, e.g. output filename and numerical precision of results. One section only.

parameter Selection and configuration of algorithm options. One section only.

sweep Definition of parameter sweep. Multiple instances of this section are allowed.

variables Definition of variables. One section only.

The actual positioning and order of results and attribute definitions in the file is irrelevant to the output of CONMLS and can be chosen ad libitum. However, note that the relative order of

Listing 4.1: Generic Simulation Setup File (.sim)

```
[global]
# simulation configuration file for mlss.2
identifier              = mlss.2

[variables]
myVar                    = 30e-9 * sin(pi/5)
my_2nd_Var               = .5 * $myVar$

[sweep]
# name of the variable to be swept
name                    = freq
start                   = 1e8
stop                    = 4e10
steps                   = 400
# possible sweep types: linear, log
type                    = linear

[files]
geometry                = 2d_board_definition.geo
stackup                 = stackup_definition.geo
mtl                     = mtl_definition.geo

[parameter]
autoseg_parameter       = 1
boundary_parameter      = 1
bc                      = PMC
frequency               = $freq$
method_zpp              = crm_single
modesm                  = 300
modesn                  = 300

[output]
append                  = 1
format                  = lydite
# Output filename. If append==0, this is used as prefix and "_###"
# is added to the individual filenames where "###" denotes the
# simset index.
filename                = output_basename
precision               = double
```

multiple parameter sweep definitions will impact their delineation through CONMLS and as such will impact the automatic indexing on output.

! All input will be turned to lower-case letters. This is particularly worth noting, because actual filenames need to be spelled using lower-case letters.

4.1.1 Model Parameterization

CONMLS allows for model parameterization by means of variables that can be used as placeholder for constant values as well as parameter sweeps, cf. subsequent section. Declaration and initialization of variables is achieved by means of simple attribute-value pairs in the *variables* section: Variables are available and may be used for the assignment of values throughout

```
[variables]
# declare a frequency value of 20GHz
freq        = 20e9
# yet another arbitrary value
pitch_new   = 10
```

the entire simulation setup as well as geometry and material definitions. They are evaluated as complex-valued mathematical expressions, which are not limited to basic operations such as addition, subtraction, multiplication and division. Expressions may contain mathematical functions such as polynomials, exponentials with arbitrary base (a^b), exp, sin, cos, tan, and log.

The following example is to demonstrate the utilization of variables by means of frequency-dependent material characteristics. Consider the linearly interpolated permittivity

$$\varepsilon_r(f) = 3.6 + 0.2\frac{40\,\text{GHz} - f}{40\,\text{GHz} - 1\,\text{GHz}}$$

where the frequency f is given in Hertz. The following listing shows the corresponding notation for input to CONMLS:

```
epsilon_r = 3.6 + .2*(40e9-$freq$)/(40e9-1e9)
```

Variables are enclosed by dollar symbols. Here, *$freq$* denotes the frequency which is subject to a parameter sweep. The permittivity of the given dielectric is linearly interpolated between two known samples $\varepsilon_r\,(1\,\text{GHz}) = 3.8$ and $\varepsilon_r\,(40\,\text{GHz}) = 3.6$.

4.1.2 Parameter Sweeping

Listing 4.1 includes a generic parameter sweep. While this example defines only one sweep, multiple instances may be declared. Sweeps declare a variable *name*, start and stop values, the number of steps, as well as the type of steps (linear, log) as shown in the following listing:

```
[sweep]
# name of the variable to be swept
name                    = freq
start                   = 1e8
stop                    = 4e10
steps                   = 400
# possible sweep types: linear, log
type                    = linear
```

Note: the relative order of declaration of sweeps impacts the processing through CONMLS. Ultimately, this affects the indexes that are assigned to the result sets. Given three sweeps A, B, and C with three steps each, Table 4.1 denotes the outcome for two different orderings of declaration.

Table 4.1: Numbering of partial results according to interleaving parameter sweeps.

Order of Sweep Declaration	Index Pattern
A, B, C	**#1:** A1 / B1 / C1, **#2:** A1 / B1 / C2, **#3:** A1 / B1 / C3, **#4:** A1 / B2 / C1, **#5:** A1 / B2 / C2, . . . , **#27:** A3 / B3 / C3
C, B, A	**#1:** C1 / B1 / A1, **#2:** C1 / B1 / A2, **#3:** C1 / B1 / A3, **#4:** C1 / B2 / A1, **#5:** C1 / B2 / A2, . . . , **#27:** C3 / B3 / A3

The syntax of the INI-format that CONMLS utilizes allows for number formatting. This may be utilized for discrete selection and thus sweeping of numbered items in the geometry specification. The following example sweeps the variable *plateIdx* over the values 1, 2, 3, and 4. The parameter sweep definition is given in Listing 4.2.

Consider the stackup in Listing 4.3 and the blind vias defined in Listing 4.4. The PCB consists of four cavities which are confined by the metallic layers *pl00* through *pl04*. Via v1 reaches through the first cavity only, i.e. between layer *pl00* and *pl01*. Via *v2*, on the other hand, ranges down to a variable layer. In order to reproduce the label of the layer in the last argument of the specification, the variable is printed as two-digit integer.

Listing 4.2: Advanced parameter sweeping, definition of parameter sweep.

```
[sweep]
name  = plateIdx
start = 1
stop  = 4
steps = 4
type  = linear
```

Listing 4.3: Advanced parameter sweeping, geometry definition of stackup: four cavities, the upper most cavity includes a signal layer.

```
[stack]
layer1  = pl00
layer2  =   diel01a
layer3  =     signal01
layer4  =   diel01b
layer5  = pl01
layer6  =   diel02
layer7  = pl02
layer8  =   diel03
layer9  = pl03
layer10 =   diel04
layer11 = pl04
```

Listing 4.4: Advanced parameter sweeping, geometry definition of vias in board file.

```
[vias]
v1={blind, std, n1,  40, 100, pl00, pl01}
v2={blind, std, n1, 120, 100, pl00, pl$plateIdx,i2.2,x$}
```

4.1.3 Algorithm Configuration

The following options select and configure the algorithms to be used for simulation. The attributes are to be placed in the *parameter* section of the .sim-file.

airap_top_bot — "airy" antipads
Assume that antipads in the top and bottom metal layer of a board are filled with air instead of the dielectric material of the respective cavity. This property often applies to manufactured PCBs. Option for research purposes, has not been tested thoroughly.

1	yes
0	no *(default)*

autoseg_parameter — network parameter representation during cavity segmentation
Choice of network parameters to be used for the segmentation of cavities.

1	S-parameter, original formulation, which may be more numerically stable (annot. of editor: not sufficiently proven) *(default)*
3	Y-parameter, faster than S-parameter since no additional parameter conversion (inversion) needs to be conducted

bc — boundary condition
Boundary conditions to be used at dielectric board edges.

PEC	perfectly electrically conducting surface at boundary
PMC	perfectly magnetically conducting surface at boundary *(default, for all cavities)*
PML	perfectly matched layer surface at boundary

boundary_parameter — network parameter representation for cavity bounds
Choice of network parameters to be used for boundary layer generation. This parameter should be chosen in alignment with *autoseg_parameter* to avoid unnecessary parameter conversion. Benchmark testing is carried out only for a selection that aligns with *autoseg_parameter*.

1	S-parameter *(default)*
3	Y-parameter

coax_ext_all — flag for coaxial port extensions
Select whether to add coaxial extensions to all ports (or only first half of ports, e.g. upper side only) after the simulation.

1	all *(default)*
0	first half only

coax_ext_len — length of coaxial port extensions
Length of coaxial extensions to be added to all ports after the simulation (length in mil). This feature may be useful when correlating to results obtained from CST Microwave Studio [17].

[numeric]	length of extension (default: 0 mil)

direct_ypp — direct calculation of Z^{pp}
Instead of calculating the parallel plate impedance Z_{pp} and subsequently converting the network parameters in order to obtain the parallel plate admittance Y^{pp}, Y^{pp}, the admittances are calculated directly. This spares an inversion and is thus more efficient. Dependencies: only relevant if method = cim_pml.

1	yes
0	no *(default)*

frequency — frequency to of operation
Numerical value of operating frequency at which the given geometry is to be considered.

[numeric]	frequency of operation for simulated PCB

gnd_reduction — reduce gnd vias using Schur's complement (2D)
Vias that are connected to both reference planes of a cavity are eliminated from the $Z_t ext pp$ matrix of that cavity in an early step of the calculation using the Schur's complement. This is the most efficient option, since it reduces the matrix size for all following operations.

1	yes *(default)*
0	no

mem_reduction — interchange computational steps to save memory
Instead of collecting the data for all cavities and performing the segmentation as a final step, the cavity results are combined as soon as they are available. While this option is deactivated by default (original implementation), it may enforce an early release of allocated memory and reduces the overall memory requirements.

1	yes
0	no *(default)*

method_zpp — method for determination of Z_{pp}
The specified method is used for the calculation of the parallel-plate impedance Z_{pp}.

cim	Contour integral method for regularly and irregularly shaped PCBs [16]. The boundary conditions may be freely chosen $\in$ {PEC, PMC, PML}.
cim_pml	Analytical formulation of contour integral method for infinite planes [18]. Solves passivity violations observed with the *rwg* for via arrays with a small pitch. Numerical effort increases by a factor of about 2 (with *direct_ypp* option) or a factor of about 3 (without *direct_ypp* option) in comparison to the *rwg*. Dependencies: *bc* = PML.
crm_double	Cavity resonator model, double summation [19, ch. 2]. Dependencies: *bc* $\in$ {PEC, PMC}.
rwg	Radial waveguide method [20]. Neglects the size of the second port. Dependencies: *bc* = PML.
rwg2	[not implemented yet] – radial waveguide method, including Bessel function for finite size of second port [21] Dependencies: *bc* = PML.

modesm — number of modes m for crm_single and crm_double
Sets the number of modes *m* to be employed in the cavity resonator method calculation. The required number depends on factors such as the board size and the frequency range. If a high accuracy is required, the convergence should be tested with at least two different values for *m*. Dependencies: only relevant if method $\in$ {crm_double, crm_single}.

[numeric]	number of modes *(default: 200)*

modesn — number of modes n for crm_double
Sets the number of modes *n* to be employed in the cavity resonator method calculation, cf. option *modesm*. Dependencies: only relevant if method = crm_double.

[numeric]	number of modes *(default: 200)*

nmodes_cap — number of modes for via-to-plate capacitance calculation
Sets the number of modes to be during calculation of via-to-plate capacitances. The required number depends on factors such as the antipad and via radius and the frequency range. If a high accuracy is required, the convergence should be tested with at least two different values.

[numeric]	number of modes *(default: 20)*

nmodes_pad — number of modes for pad-to-plate calculations
Sets the number of modes to be during calculation of pad-to-plate capacitances. The required number depends on factors such as the pad and via radius and the frequency range. If a high accuracy is required, the convergence should be tested with at least two different values.

[numeric]	number of modes *(default: 20)*

subtract_tlc — subtract coaxial capacitance from imported capacitance
Subtract the coaxial capacitance (calculated by CONMLS) from the imported, fixed capacitance values. This is useful if transmission line models are employed to model the coaxial sections of vias, since the transmission line models already take the coaxial capacitance into account. Note: a subtraction is only necessary, if the imported capacitances include the coaxial sections in the first place.
Dependencies: *tl_coax* = 1

1	yes *(default)*
0	no

tlcoax —
use transmission line segments to describe coaxial region of vias

1	yes *(default)*
0	no

zpp_red — reuse the Z_{pp} matrix
All cavities feature the same geometrical via arrangement and have the same height. Recycle the Z^{pp} of the first cavity for all of the other cavities. Different connectivities of the vias in the respective cavities are accounted for separately.

1	yes
0	no *(default)*

zpp_red_gnd — reuse the Z_{pp} matrix after ground reduction
The prerequisites of the option *zpp_red* are extended by means of same via connectivity in all cavities. The reduced Z_{pp} is computed only once for all cavities. Dependencies: *zpp_red = 1*.

1	yes
0	no *(default)*

4.1.4 Configuration of CIM Algorithm

The following options are to be placed in the *cim_config* section of the .sim-file.

bc_ho_via — termination of higher order circular modes at vias
Boundary condition to terminate higher order circular modes at vias.

PEC	perfect electric conductor *(default)*
PML	perfectly matched layer

max_elem_len — maximum length of a segment
Maximum length of a segment (unit: mm).

[numeric]	length value *(default: 10)*

nmode — number of higher order circular modes
Specifies the highest order of circular modes that is to be to considered in determining Z_{pp}.

[numeric]	length value *(default: 0)*

seg_per_wavelen — minimum number of segments per wavelength
Defines the minimum number of segments per wavelength. The algorithm tries to set the number of segments to the given value and increases the amount of segments if subdivision fails.

[numeric]	length value *(default: 10)*

4.1.5 Linking Geometry Specifications

The descriptions of the PCB and the stackup are mandatory. The geometrical description of MTLs is subject to choice of transmission line modeling and thus optional.

geometry — name of PCB .geo-file
References a file for the two-dimensional description of a PCB.

[character string]	Name of .geo-file (with file-extension).

stackup — name of stackup .geo-file
References a file for the description of a stackup.

[character string]	Name of .geo-file (including file-extension).

mtl — name of MTL .geo-file
References a file for the description of the multiconductor transmission lines (MTLs).

[character string]	Name of .geo-file (including file-extension).

4.1.6 Selection of Output Data

The following options are to be placed in the *output* section of the .sim-file.

append — flag for output file multiplicity
This flag determines whether all data is to be stored in one file or separate files, i.e. one file per configuration in case of parameter sweeping.

1	yes *(default)*
0	no

format — output file format
Choice of output file format. Results may be stored using the HDF5-based [22] lydite-format, i.e. binary .lyd.h5-files. These files contain additional meta-data, including the entire analysis-setup. This format is recommended whenever processing ten or more ports. The file-size is about 30 % of a touchstone®-file and less.

lydite	*(default)*
touchstone	(to be available as of June 2016)

filename — file-/basename of output file
Sets output filename. If *append*==0, this string is used as prefix and "_###" is added to the individual filenames where "###" denotes the index of the simulation set during parameter sweep operation.

[character string]	file-/basename of output file

precision — floating point precision of resulting network parameters
Selection the preferred floating point precision of resulting network parameters.

double	use double precision floating point numbers
single	use single precision floating point numbers

4.2 Specification of Board Geometry (2D) and Components

The geometry of the printed circuit board (PCB) under consideration are provided by means of a .geo-file. It contains the units used for the geometrical parameters, the two-dimensional PCB description and as well as the description of components such as vias and transmission lines. Finally, ports are placed in this file.

This file is referenced from the main .sim-file. An illustrative board description is given in Listing 4.5. Valid sections are:

global contains the general header and information for file sanity checks.

ideal_tls for the placement of ideal transmission lines.

lumped specifies lumped RLC elements.

mstls refers to microstrip transmission lines. **Warning**: this is an untested feature.

mtls references to separately defined multiconductor transmission lines.

pads places via pads. **Warning**: this is an untested feature.

ports lists all external ports of the system which are to be accessible by means of resulting S-parameters.

shape defines the shape of the PCB under test.

viamodel declares details of via geometry and modeling. Multiple instances of this sections are allowed, e.g. to declare different types of vias.

vias for the geometrical placement and auxiliary specification of vias.

If not stated otherwise, all of these sections may only be placed once in the .geo-file.

Listing 4.5: Syntax of a PCB description as .geo-file

```
[global]
identifier = mlss.2
type       = board
units      = mil

[shape]
type = rect
xmin = 0
xmax = 600
ymin = 0
ymax = 2000

[viamodel]
name               = std_vmod
### via-geometry
via_radius         = 5
inner_radius       = 4
#..note that there are n_cavities+1 planes
antipad_radius     = 15
pad_radius_plane   = {10, 0, 0, 0, 10}
pad_radius_signal  = 0
material           = copper
### capdef
cap_type = auto
#..frequency dependency (yes or no)
cap_fd   = yes
#..definition of upper value of via fringing capacitance
cap_cfu  = 0
#..definition of lower value of via fringing capacitance
cap_cfl  = 0

[mtls]
#label={id,  layer,netparam,k,    nPort,v1_lbl,v2_lbl,...,vN_lbl}
mtl1  ={tl1, sig1, y,       -0.5, 2,    via1,  via2}

[vias]
#label = {type, model,     net, Xpos, Ypos}
via1   = {thru, std_vmod,  s1,  250,  250}
via2   = {thru, std_vmod,  s1,  350,  250}

[ports]
#label    = {via_id, type, side, ref-impedance (w/o function)}
via1_up   = {via1,   se,   up,   50}
via2_up   = {via2,   se,   up,   50}
via1_low  = {via1,   se,   low,  50}
via2_low  = {via2,   se,   low,  50}
```

Similar to the .sim-file, a *global* section is mandated for explicitly dedicate this file for use with CONMLS. Furthermore, the type of the geometry description is explicitly stated as well as the units which are to be applied to all geometrical dimensions:

Details of the remaining sections and all supported (as well as some proposed) attributes are

units — unit for geometry specification
The unit of geometrical specification of the module may be used globally. Several components will use this definition, if not specified otherwise.

[character string]	Abbreviation of geometrical unit $\in \{\mathrm{mm}, \mathrm{cm}, \mathrm{m}, \mathrm{mil}, \mathrm{inch}\}$.

given in the following sections.

4.2.1 Shape

The *shape* section specifies the two-dimensional outline of the PCB.

type — outline of a printed circuit board
Selects the type of shape for the printed circuit board and its cavities.

rect	Set board to be rectangular. Additionally, the arguments *xmin, xmax, ymin* and *ymax* are required to define the boundary of the board.
poly	Irregular/arbitrary plane stack: definition of arbitrarily shaped planes (identical for all cavities). This feature is used in conjunction with the CIM algorithm. Additionally, the arguments *xlist* and *ylist* are required to define the x- and y-coordinates of a polyline that curtails the board. Note: the coordinates need to be ordered and listed in mathematical positive direction of rotation, i.e. counter-clockwise.
gmsh	Choose to provide a gmsh-file which defines a discretized polyline. This feature is used in conjunction with the CIM algorithm. Additionally, the argument *filename_gmsh* is required to link the corresponding file. The file is read in and processed within the CIM algorithm, the *units* setting is subject to consideration therein.

filename_gmsh — filename for discretized gmsh-polyline
Links a gmsh-file which defines a discretized polyline.

[character string]	filename

gmsh_entity — contour index that refers to gmsh input file
Physical entity as provided by gmsh, used to select from multiple contours that may be provided in one file.

[numeric]	integer value

xlist — x reference points
This attribute is used to enter the x-values of the reference coordinates that span the printed circuit board and its cavities:

```
{val_1, val_2, ..., val_N}
```

[character string]	array of values

xmax — upper bound for PCB size in x-direction
Part of a rectangular board definition, cf. type:rect. Should be larger than *xmin*.

[numeric]	double-precision floating point value $\in (-10^{300}, 10^{300})$

xmin — lower bound for PCB size in x-direction
Part of a rectangular board definition, cf. type:rect. Should be smaller than *xmax*.

[numeric]	double-precision floating point value $\in (-10^{300}, 10^{300})$

ylist — y reference points
This attribute is used to enter the y-values of the reference coordinates that span the printed circuit board and its cavities:

```
{val_1, val_2, ..., val_N}
```

[character string]	array of values

ymax — upper bound for PCB size in y-direction
Part of a rectangular board definition, cf. type:rect. Should be larger than *ymax*.

[numeric]	double-precision floating point value $\in (-10^{300}, 10^{300})$

ymin — upper bound for PCB size in y-direction
Part of a rectangular board definition, cf. type:rect. Should be smaller than *ymax*.

[numeric]	double-precision floating point value $\in (-10^{300}, 10^{300})$

4.2.2 Via Geometry and Modeling

A via model describes the via geometry, its material, and configures the calculation of via-to-plane capacitances. One model may be used multiple times throughout a board description. On the other hand, multiple *viamodel* sections may be defined in order to allow for different vias on a PCB.

The syntax of a *viamodel* section is illustrated in Listing 4.6. Details on valid attributes and respective values are provided in the subsequent list. The geometric parameters are illustrated in Fig. 4.1.

! The number of planes N_{pl} is the number of cavities plus one ($N_{\mathrm{cav}} + 1$), since the cavities are by means of confining metallic planes.

Listing 4.6: Illustrative Via Model

```
[viamodel]
### general specification
name              = my_via_model
material          = copper
### geometry
via_radius        = 5
inner_radius      = 4
#..stating equal antipad radius for all layers, ..
antipad_radius    = 15
#..plane pad radius > 0 only at top and bottom (5 layers), ..
pad_radius_plane  = {10, 0, 0, 0, 10}
#..and the same (no) signal pad radius for all signal layers
pad_radius_signal = 0
### parameters that concern the capacitance calculation
cap_type          = auto
#frequency dependency (yes or no)
cap_fd            = yes
#definition of upper value of via fringing capacitance
cap_cfu           = 0
#definition of lower value of via fringing capacitance
cap_cfl           = 0
```

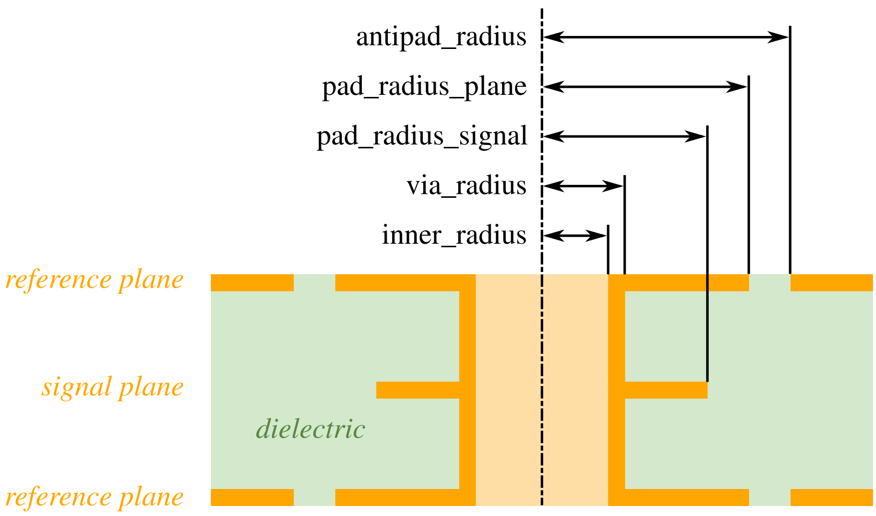


Figure 4.1: Geometry of a via model, cross-sectional view.

An essential part of via models are via-to-plane capacitances that are used to represent the interaction between via barrel and reference planes, cf. Figure 4.2. Attributes with prefix *cap_* concern the algorithms to be used for determining the via-to-plane capacitances.

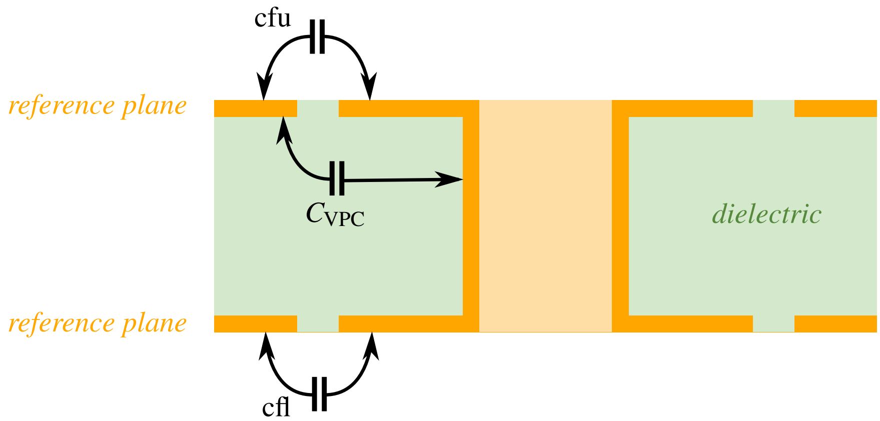


Figure 4.2: Via-to-plane capacitances in a via model. The capacitance C_{VPC} refers to the via-to-plane capacitance.

antipad_radius — radius of dielectric antipad
Geometry parameter as specified in Fig. 4.1. A range of N_{pl} values may be given to set different radii for each plane that is pierced. If only one value is given, it is used for all cavities.

[numeric]	Set value(s) with regard to chosen *unit* specified in .geo-file of board geometry.

cap_fd — frequency dependency
Frequency dependency of analytical values

yes	Do account for frequency dependent behavior of the via-to-plane capacitances.
no	No frequency dependence, use the value determined at *fmin* for all frequencies. Experience showed, that the capacitances hardly change in the considered frequency ranges. On the other hand, the frequency dependency may be included at almost no cost (calculated once per capacitance-specification).

cap_cfl — lower fringing capacitance
The lower fringing capacitance is the capacitance due to electric field lines through air (outside the PCB) at a lower via end. It may be required to accurately model open terminations in some cases, especially for thin PCBs (only one or a few layers). It is given as a fixed value for all capacitance types and is added to the respective computed or given values.

[numeric]	Set value of lower via fringing capacitances.

cap_cfu — upper fringing capacitance
The upper fringing capacitance is the capacitance due to electric field lines through air (outside the PCB) at a lower via end. It may be required to accurately model open terminations in some cases, especially for thin PCBs (only one or a few layers). It is given as a fixed value for all capacitance types and is added to the respective computed or given values.

[numeric]	Set value of upper via fringing capacitances.

cap_type — source for values
Selection of calculation method for the via near-field model.

auto	Use via barrel-to-plane capacitances calculated from analytical formulas available in the CONMLS code based on [23]. Possibly specified fixed values will be ignored.
auto2	This option is based on the coaxial to radial waveguide junction model described in [24], using only the elements which correspond to the via barrel-to-plane capacitances. This option may be interesting for studies in a research context. For practical investigations, the option *wg_junct* should be preferred, which uses the more accurate full waveguide junction model. Possibly specified fixed values will be ignored.
fix	Use fixed values as defined in the .cap-file. Values may be given as samples with regard to frequency to account for a frequency-dependent behavior. This option can be used to import capacitance values extracted with an external tool.
semiauto	Automatic computation (similar to *auto*) for entries assigned with -1, others assumed as fixed values. Used e.g. for buried via models. Number of capacitances given should be consistent with the stackup definition.
wg_junct	Coaxial to radial waveguide junction model based on [24]. The model uses additional elements (frequency dependent admittances and an ideal transformer) to model the via near-field more accurately. This model gives the most accurate results, without (noticeably) increasing the computation time.

inner_radius — inner radius of via barrel
Geometry parameter as specified in Fig. 4.1. A range of N_{cav} values may be given to set different radii within each cavity. If only one value is given, it is used for all cavities. **Note: values are currently not relevant for the calculation!**

[numeric]	Set value with regard to chosen *unit* specified in .geo-file of board geometry.

material — name of .matr-file
References a files for the definition of the conducting material which the via is made of.

[character string]	Name of .matr-file (without file-extension).

name — identifier of via model
Name of via model for ease of referencing in other parts of the problem description.

[character string]	name of via model.

pad_radius_plane — radius of pad via within reference planes
Geometry parameter as specified in Fig. 4.1. A range N_{pl} of values may be given to set different radii for each plane that is pierced. If only one value is given, it is used for all cavities.

[numeric]	Set value with regard to chosen *unit* specified in .geo-file of board geometry.

pad_radius_signal — radius of pad via within signal layers
Geometry parameter as specified in Fig. 4.1. A range of N_{pl} values may be given to set different radii for each plane that is pierced. If only one value is given, it is used for all cavities. **Note: except for special cases, signal pads will be ignored in the calculation!**

[numeric]	Set value with regard to chosen *unit* specified in .geo-file of board geometry.

via_radius — outer radius of via barrel
Geometry parameter as specified in Fig. 4.1. A range of N_{cav} values may be given to set different radii within each cavity. If only one value is given, it is used for all cavities.

[numeric]	Set value with regard to chosen *unit* specified in .geo-file of board geometry.

4.2.3 Via Placement

Vias are placed using a *vias* section of the board description (.geo-file). The general syntax for the array that specifies a via is *{[type], [model], [connectivity/net], [xpos], [ypos], [start_layer][1], [end_layer][1], [start_depth][2], [end_depth][2]}*, where
[1] applies to buried, blind, and stripline probes only, and
[2] applies to buried vias only. The values specify the penetration depth for the first and last layer, respectively (e.g. 0.5).

```
[vias]
#label       = {type, model, net, Xpos, Ypos[, pFrom, pTo, d0, d1]}
via_thru     = {thru,     vmod,     s1,  0,   0}
via_blind    = {blind,    vmod,     s1, 40,   0,  pFrom, pTo}
via_buried   = {buried,   vmod,     s1, 40,  40,  pFrom, pTo, d0, d1}
via_slprobe  = {slprobe, generic,        0,  40,  pFrom, pTo}
```

Valid types of vias are:

thru A via that range from the upper most layer down to the bottom of the stackup. Syntax:

```
label={thru, model, net, xpos, ypos}
```

blind A backdrilled via, that does not range down to the bottom of the stackup. Syntax:

```
label={blind, model net, xpos, ypos, start_layer, end_layer}
```

buried Vias that are enclosed within the stackup, i.e. do not necessarily reach top or bottom of the stackup. Syntax:

```
label={buried, model, net, xpos, ypos, start_layer, end_layer,
    start_depth, end_depth}
```

slprobe Available as of rev560 for S-parameter-based segmentation [7, 8]. This type is not a physical via. It terminates a stripline without the need of placing an actual via. This type currently does not require a model (the model can be anything, the string is currenly ignored).
Syntax:

```
label={slprobe, generic, xpos, ypos, start_layer, end_layer}
```

Application notes: Stripline probes (probe-vias) are located at the end of a single stripline and exist only within one cavity. Thus, the description by means of Z_{pp} is not the same for all cavities. Functionality for the reduction of memory (e.g. Zpp_reduction) are not available if this type of via is used. Furthermore, a network description and handling by means ABCD-parameter has not been tested for the segmentation of cavity/Z_{pp} and transmission lines.

! Due to legacy reasons, stripline ports (slports) do not make use of the conventional current/voltage definition near ports, i.e. current flowing *into* the network. Instead, the current features a phase shift of 180°. This needs to be considered if the obtained S-parameter are to be used in subsequent simulations, e.g. using TiDE.

4.2.4 Ideal Transmission Lines

Ideal transmission lines between two vias, with dielectric losses but without conductor loss. This type of transmission line is the fastest during calculation because it is based on analytical expressions. They may be specified in the *ideal_tls* section of the board description (.geo-file) according to the syntax

```
[ideal_tls]
label={type, layer, startVia, endVia, width, lenght, k, Z0}
```

Valid types include

Single ended Single ended transmission lines
Type: *se*

where k denotes the coupling factor for the modal decomposition [25, 26, 27] and Z_0 is the characteristic impedance of the transmission line

The *layer* corresponds to the label in the stackup layer as opposed to the implementation in its predecessor VPF, in which it corresponds to the interconnect identifier.

4.2.5 Microstrip Transmission Lines (untested)

Microstrip lines may be specified in the *mstls* section of the board description (.geo-file) according to the syntax

```
[mstls]
label={type, side, origin, end, width, lenght, L}
```

! This feature is currently not subject to code testing and validation. For more details refer to the latest source code from the subversion repository.

4.2.6 Multiconductor Transmission Lines

Multiconductor (coupled) transmission lines (MTLs) are specified in a separate .geo-file as discussed below. They are referenced in the *mtls* section of the board description (.geo-file) according to the syntax

```
[mtls]
#label={id, layer, netparam, k, nPort, via1_lbl, ..., viaN_lbl}
my_mtl={id, signal3, y, -0.5, 4, via1, via2, via3, via4}
```

where the value of *id* refers to the unique identifier that is given to the geometry specification in the separate .geo-file. By this, repeating transmission lines and their corresponding network parameters are reused during time of simulation. Each MTL is characterized only once using the code 2D-MTL of Concept-II which is included as external module.

The *layer* specifier refers to the stackup to obtain the geometry and material of the considered cavity, cf. 4.3. The value of *netparam* is currently expected to equal "y" and is a placeholder for future extension. The modal decomposition factor *k* depends on the relative positioning of the trace within the substrate, cf. [25, 26, 27]. While *k* is implicitly given by the stackup, it can be entered separately due to historic reasons.

Finally, the number *nPort* indicates the number of single-ended ports of the considered MTL, followed by the corresponding amount of references to vias that are to be connected to the MTL, cf. Figure 4.3.

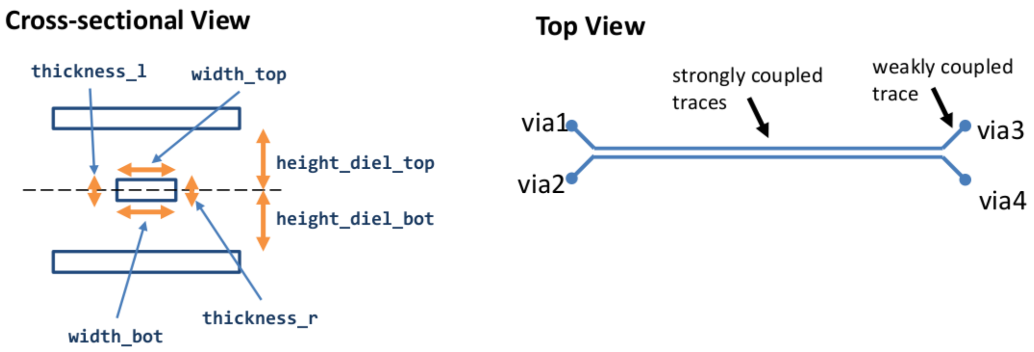


Figure 4.3: MTL geometry specification and simple MTL example with four vias/ports.

MTL Specification in Separate .geo-file

All MTL-systems are defined in one .geo-file of type *mtl*. This file is linked to the project by means of the attribute *mtl* in the *files* section of the configuration in the .sim-file:

```
[files]
mtl = mtl_filename.geo
```

The MTL .geo-file of may contains the description of all multiconductor transmission lines in scope that are to be simulated using the 2D-MTL algorithm. The description allows to define irregular traces with mixed coupled and uncoupled segments, which are interpreted and treated in the mtl_seg module (see [12]). Listing 4.7 provides an illustrative example for the syntax with regard to the system depicted in Figure 4.3.

As before, a *global* section provides sanity information regarding the content of the file, cf. illustrative example in Listing 4.7. The file may contain one or more *mtl* sections and one or more linked *line* sections, which are used to specify independent MTL-systems. The *mtl* sections contain the following information:

kc — coupling factor, lower limit
Lower limit for estimation of minimum coupling considered among adjacent conductors. Must be below 0.1.

[numeric]	sets the lower limit for k_c

line — identifier of traces (lines) that are part of the MTL
Multiple attributes may be used to reference multiple traces that are to be considered during the same simulation, e.g. to consider coupling.

[character string]	identifier

matr_cond — reference to material specification of conductor
Reference to conductor material specification.

[character string]	name of the conducting material

matr_diel — reference to material specification of dielectric
Reference to dielectric material specification.

[character string]	name of the dielectric

mode — choice of algorithm
Determines the algorithm to be used.

1	classical 2D-MTL algorithm, optimized
2	non-optimized 2D-MTL (untested)
3	Runge-Kutta (untested)
4	reuse identical cross-sections (using look-up table), can speed up simulation by 30 % to 50 %

name — identifier of MTL system
This identifier is used to reference the MTL system from the board description.

[character string]	identifier

rfactor — minimum relative deviation of two geometry points
Sets the minimal trace length to be considered.

[numeric]	sets the resolution

Note that the *mtl* section may link to one or more *line* sections that define the geometry of the TLs. Each *line* sections contains the following attributes and values:

height_diel_bot — height of dielectric between trace center and lower reference plane
This value specifies a geometrical parameter of the trace, cf. Figure 4.3.

[numeric]	dimension: length

height_diel_top — height of dielectric between trace center and upper reference plane
This value specifies a geometrical parameter of the trace, cf. Figure 4.3.

[numeric]	dimension: length

name — identifier of line
This identifier is used to reference the line system from the MTL description.

[character string]	identifier

p — reference point
This attribute is used to enter the 2D coordinates of the trace center. As least two reference points are required to define a trace. Points are connected in order of their respective index:

```
{index, xpos, ypos}
```

[character string]	array of values

thickness_l — left-hand thickness of trace
This value specifies a geometrical parameter of the trace, cf. Figure 4.3.

[numeric]	dimension: length

thickness_r — right-hand thickness of trace
This value specifies a geometrical parameter of the trace, cf. Figure 4.3.

[numeric]	dimension: length

units — unit for geometry specification
The unit of geometrical specification of a model are given explicitly.

[character string]	Abbreviation of used unit $\in \{\mathrm{mm}, \mathrm{cm}, \mathrm{m}, \mathrm{mil}, \mathrm{inch}\}$.

width_bot — bottom width of trace
This value specifies a geometrical parameter of the trace, cf. Figure 4.3.

[numeric] dimension: length

width_top — top width of trace
This value specifies a geometrical parameter of the trace, cf. Figure 4.3.

[numeric] dimension: length

Listing 4.7: Example of a .geo-file for MTL description.

```
[global]
identifier = mlss.2
type       = mtl
[mtl]
name          = mtl_seg_test2
mode          = 1
matr_cond     = copper
matr_diel     = meg
kc            = 0.001
rfactor       = 0.01
line          = line1
line          = line2
[line]
name            = line1
units           = mil
width_top       = 4
width_bot       = 4
height_diel_top = 6
height_diel_bot = 6
thickness_l     = 1
thickness_r     = 1
p               = {1,  0,     0}
p               = {2, 46,    46}
p               = {3, 46,  1454}
p               = {4,  0,  1500}
[line]
name            = line2
units           = mil
width_top       = 4
width_bot       = 4
height_diel_top = 6
height_diel_bot = 6
thickness_l     = 1
thickness_r     = 1
p               = {1, 100,     0}
p               = {2,  54,    46}
p               = {3,  54,  1454}
p               = {4, 100,  1500}
```

4.2.7 Lumped Elements

Lumped elements, that is, 1-port RLC series or parallel networks, may be specified in the *lumped* section of the board description (.geo-file) by means of the syntax

```
[lumped]
label={type, layer, via, R_value, L_value, C_value}
```

Types: RLCser, RLCpar fixed values given for R,L, C elements connected to one via port in series or parallel, respectively. Useful to define e.g. terminations and decoupling capacitors.

R Lumped elements are placed between a via and the corresponding layer at which they are placed. A short circuit can be defined using the parallel-type lumped elements with $R = L = 0$ and $C = 1 \neq 0$, e.g.

```
short = {RLCpar, myLayer, myViaPort, 0, 0, 1}
```

! Only one lumped element, that is, one set of R, L, and C is supported per location. The algorithm does not issue a warning in case there is more than one set of lumped elements connected. It is not clear, if only the first or all sets of elements are included in the simulation.

4.2.8 Pads (untested)

Pads may be specified in the *pads* section of the board description (.geo-file) by means of the syntax

```
[pads]
label={type, layer, via, pshape, size, cup, clow}
```

! This feature is currently not subject to code testing and validation. For more details refer to the latest source code from the subversion repository.

4.2.9 Ports

Ports may be specified in the *pads* section of the board description (.geo-file) by means of the syntax

```
[ports]
label = {viaref, type, side, impedance}
```

In case the results are written to the HDF5-based lydite-format, the label is stored as well for ease of reference.
Valid types of ports are:

Single ended Single ended ports
Type: *se*

Valid sides for ports are:

Upper side Ports located on the upper side of the stackup. Note stripline-ports are always located at the upper side.
Type: *up*

Lower side Ports located on the lower side of the stackup.
Type: *low*

As of the current revision, the reference impedance is without impact. The algorithm assumes a reference impedance of $50\,\Omega$ during network parameter conversion.

R Stripline ports that terminate microstrip lines are feasible in general. However, this feature has not been tested or validated until now and is considered to be experimental.

4.3 Stackup Geometry

The stackup is defined in a .geo-file of type *stackup* and is linked from the configuration .sim-file. As illustrated in Listing 4.8, the *global* section includes sanity check information as well as the unit of the length dimensions throughout the file:

units — unit for geometry specification
The unit of geometrical specification of a model are given explicitly.

[character string] Abbreviation of used unit $\in \{\mathrm{mm}, \mathrm{cm}, \mathrm{m}, \mathrm{mil}, \mathrm{inch}\}$.

The *stack* section lists all layers. A continuous index is appended to the keyword *label* to specify the relative location of a layer in the stack, cf. Listing 4.8. Each layer is subsequently specified in a separate *layer* section. Some attributes may depending on the type the layer (dielectric, plane, signal):

connectivity — identifier of connecting net
Reference to the net to which the plane or signal layer is connected. This attribute is without function in case of a dielectric layer.

[character string] name of the material

material — reference to material specification
Reference to conductor or dielectric material specification.

[character string] name of the material

name — identifier of layer
This identifier is used to reference the layer in the *stack* section of the stackup description.

[character string] identifier

thickness — layer thickness
This value defines the thickness of the layer in accordance with the unit that is defined in the *global* section of the stackup .geo-file.

[numeric] dimension of length

type — functionality of layer in stackup
Select the functionality of the layer within the given stackup. The value may be one of the following:

dielectric	Dielectric layer.
plane	Conducting reference plane.
signal	This is a placeholder for a signal layer. The thickness of this layer is filled using the dielectric of the adjacent dielectric layers.

Listing 4.8: Illustrative stackup description using a .geo-file

```
[global]
identifier = mlss.2
units      = mil
type       = stackup

[stack]
layer1 = power1
layer2 =   diel1a
layer3 =     signal1
layer4 =   diel1b
layer5 = power2

[layer]
name      = diel1a
type      = dielectric
material  = meg
thickness = 5.5
[layer]
name      = diel1b
type      = dielectric
material  = meg
thickness = 5.5

[layer]
name          = power1
type          = plane
connectivity  = gnd
thickness     = 1
material      = copper
[layer]
name          = power2
type          = plane
connectivity  = gnd
thickness     = 1
material      = copper

[layer]
name         = signal1
type         = signal
connectivity = sl1
thickness    = 1
material     = copper
```

4.4 Material Specification

The .matr-files are used to define material parameters and may be frequency dependent, cf. remark below. The syntax behaves according to the generic examples given in Listings 4.9 and 4.10. The filenames are referenced from stackup, via models, as well as external MTL algorithm. They need be spelled lower case.

Listing 4.9: Generic Conductor Description in a .matr-file.

```
[global]
identifier = mlss.2
type       = conductor
# Material definition for Copper (frequency independent)
[properties]
conductivity = 5.8e7
mu_r         = 1
```

Listing 4.10: Generic Dielectric Description in a .matr-file.

```
[global]
identifier = mlss.2
type       = dielectric
# Material definition for Megtron (frequency independent)
[properties]
tand      = 0.02
epsilon_r = 3.8
mu_r      = 1
```

R Variables are enclosed by dollar symbols. In the following example, *$freq$* denotes the frequency which is subject to a parameter sweep. The permittivity of the given dielectric is linearly interpolated between two known samples $\varepsilon_r\,(1\,\mathrm{GHz}) = 3.8$ and $\varepsilon_r\,(40\,\mathrm{GHz}) = 3.6$.

```
#== MATERIAL DEFINITION ================================
[global]
identifier = mlss.2
type       = dielectric

# Material definition for meg (frequency independent)
[properties]
tand      = 0.02
epsilon_r = 3.6 + .2*(40e9-$freq$)/(40e9-1e9)
mu_r      = 1
```

5 — Licensing

Interested in working with the full version of CONMLS? To request a Windows license file from the contact person, you will need to name the MAC-address and the C: drive volume number of the computer to be used. This information can be obtained from the command prompt using the commands:

```
pconfig /all
```

and

```
vol c:
```

Alternatively, the script mlss_cpuinfo.exe in the CONMLS binary directory can be used to automatically generate a text file containing this information.

3rd-Party Packages

CONMLS uses the following 3rd-Party packages:

- Qt 4.8.6
  - LGPL
  - *https://www.qt.io/download-open-source/#section-2*
  - The usage of the software is according to the GPL and the LGPL (see below).
- HDF5 1.8.12
  - BSD
  - *http://www.hdfgroup.org/products/licenses.html*
- Inno Setup Compiler 5.5.5
  - *http://www.jrsoftware.org/isinfo.php*

For details on the respective licenses/copyrights see the files in the directory *$INSTALLATION_DIRECTORY/licenses*. Text files of the GPL and the LGPL are included (gpl.txt, lgpl.txt).

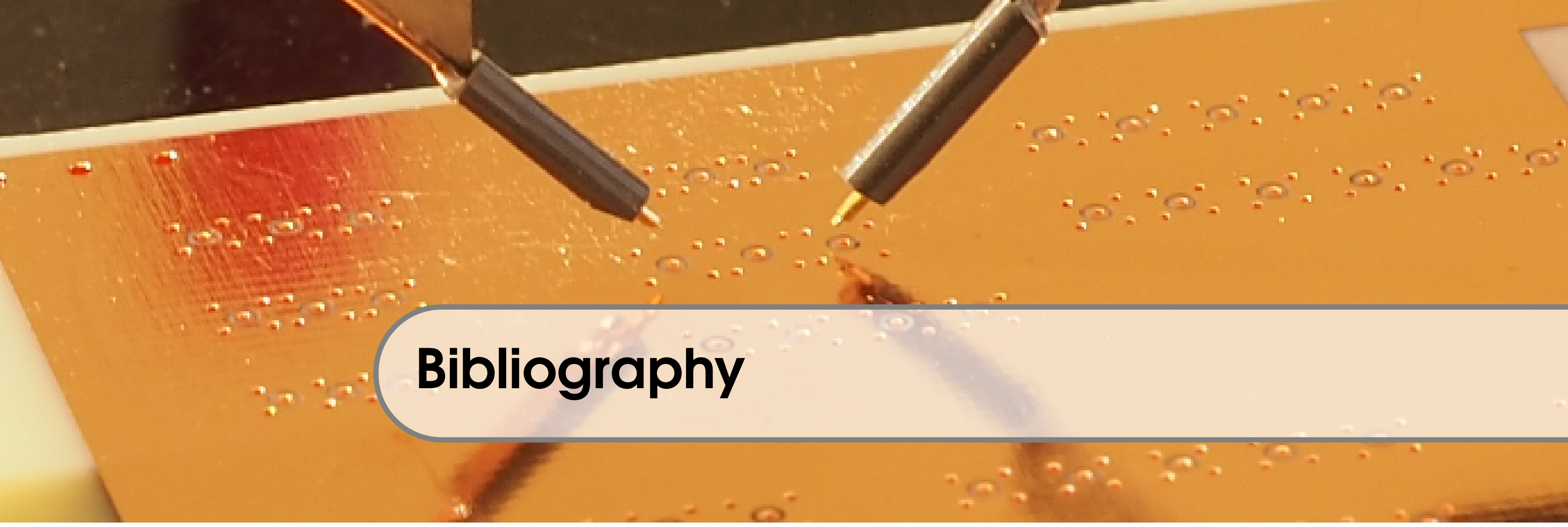
Contact

For license generation, assistance, or questions please contact:

Technische Universität Hamburg (TUHH)
Institut für Theoretische Elektrotechnik
c/o Dr.-Ing. Heinz-D. Brüns (bruens@tuhh.de)
Harburger Schloss Str 20
21079 Hamburg

Germany

WWW.TET.TUHH.DE



Bibliography

[1] S. Chapman, *Fortran 95/2003 for scientists & engineers*, 3rd ed. New York, USA: McGraw-Hill, 2008.

[2] Mathworks Corporation. (2009) Matlab ver. 2009. Natick, MA. [Online]. Available: http://www.mathworks.com

[3] R. Rimolo-Donadio, *Via Pin Field (VPF)-Tool Documentation*, Institut of Electromagnetic Theory, Hamburg University of Technology, Hamburg, Germany, 2010, internal document.

[4] R. Rimolo-Donadio, "Development, validation, and application of semi-analytical interconnect models for efficient modeling of multilayer substrates," Ph.D. dissertation, Hamburg University of Technology, Hamburg, Germany, 2010.

[5] R. Rimolo-Donadio, X. Gu, Y. Kwark, M. Ritter, B. Archambeault, F. de Paulis, Y. Zhang, J. Fan, H.-D. Brüns, and C. Schuster, "Physics-based via and trace models for efficient link simulation on multilayer structures up to 40 GHz," *IEEE Transactions on Microwave Theory and Techniques*, vol. 57, no. 8, pp. 2072–2083, Aug. 2009.

[6] R. Rimolo-Donadio, A. Stepan, H.-D. Brüns, J. Drewniak, and C. Schuster, "Simulation of via interconnects using physics-based models and microwave network parameters," in *Signal Propagation on Interconnects (SPI), 12th IEEE Workshop*, May 2008.

[7] T. Reuschel, S. Müller, and C. Schuster, "Segmented physics-based modeling of multilayer printed circuit boards using stripline ports," *IEEE Transactions on Electromagnetic Compatibility*, vol. 58, no. 1, pp. 197–206, Feb. 2016.

[8] T. Reuschel, M. Kotzev, D. Dahl, and C. Schuster, "Modeling of differential striplines in segmented simulation of printed circuit board links," in *IEEE Int. Symp. Electromagn. Compat. (EMC)*, Ottawa, ON, Canada, Jul. 2016.

[9] EIA/IBIS Open Forum, "Touchstone® (.SnP) file format specification. Rev 1.1," http://vhdl.org/ibis/connector/touchstone_spec11.pdf, Mar. 2014.

[10] A. J. Stepan, "Implementation of a matlab-code for high-frequency via pin field simulation in printed circuit boards," Bachelorarbeit, TUHH, 2008.

[11] D. Timmermann, "Implementation of efficient algorithms for parallel-plate impedance calculation," Studienarbeit, TUHH, 2009.

[12] C. Schuster, "STL-Toolbox: A collection of Matlab functions for S-Parameters, T-Lines and Links," Aug. 2006.

[13] X. Duan, R. Rimolo-Donadio, H.-D. Brüns, and C. Schuster, "A combined method for fast analysis of signal propagation, ground noise, and radiated emission of multilayer printed circuit boards," *IEEE Transactions on Electromagnetic Compatibility*, vol. 52, no. 2, pp. 487–495, May 2010.

[14] H. Färber, "Segmentierte Modellierung gekoppelter Mehrfachstreifenleitungen auf Leiterplatten," Bachelorarbeit, TUHH, 2011.

[15] S. Müller, "Including multiconductor transmission lines in a quasi-analytical model for multilayer structures," Diplomarbeit, TUHH, 2009.

[16] Xiaomin Duan, R. Rimolo-Donadio, H.-D. Brüns, and C. Schuster, "Circular ports in parallel-plate waveguide analysis with isotropic excitations," *IEEE Transactions on Electromagnetic Compatibility*, vol. 54, no. 3, pp. 603–612, Jun. 2012.

[17] CST. Microwave studio. Darmstadt, Germany. [Online]. Available: http://www.cst.com

[18] X. Duan, R. Rimolo-Donadio, S. Müller, K. J. Han, X. Gu, Y. H. Kwark, H.-D. Brüns, and C. Schuster, "Impact of multiple scattering on passivity of equivalent-circuit via models," in *2011 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, Hanzhou, China, Dec. 12–14, 2011.

[19] T. Okoshi, *Planar Circuits for Microwaves and Lightwaves*, ser. Springer Series in Electrophysics. Berlin and Heidelberg: Springer Berlin Heidelberg, 1985, vol. 18.

[20] J. Parker, "Via coupling within parallel rectangular planes," *IEEE Transactions on Electromagnetic Compatibility*, vol. 39, no. 1, pp. 17–23, Feb. 1997.

[21] A. R. Chada, Y. Zhang, G. Feng, J. L. Drewniak, and J. Fan, "Impedance of an infinitely large parallel-plane pair and its applications in engineering modeling," in *2009 IEEE International Symposium on Electromagnetic Compatibility - EMC 2009*, Austin, TX, USA, Aug. 17–21, 2009, pp. 78–82.

[22] *HDF5 Reference Manual*, Release 1.8.8 ed., The HDF Group, Nov. 2011. [Online]. Available: http://www.hdfgroup.org/HDF5/doc/PSandPDF/HDF5_RefManual.PDF

[23] Yaojiang Zhang, J. Fan, G. Selli, M. Cocchini, and F. d. Paulis, "Analytical evaluation of via-plate capacitance for multilayer printed circuit boards and packages," *IEEE Transactions on Microwave Theory and Techniques*, vol. 56, no. 9, pp. 2118–2128, Sep. 2008.

[24] A. G. WILLIAMSON, "Radial-line/coaxial-line junctions: analysis and equivalent circuits," *International Journal of Electronics*, vol. 58, no. 1, pp. 91–104, 1985.

[25] E. Engin, W. John, G. Sommer, W. Mathis, and H. Reichl, "Modeling of striplines between a power and a ground plane," *IEEE Transactions on Advanced Packaging*, vol. 29, no. 3, pp. 415–426, Aug. 2006.

[26] A. Ege Engin, K. Bharath, K. Srinivasan, and M. Swaminathan, "Modeling of multilayered packages and boards using modal decomposition and finite difference methods," in *IEEE Int. Symp. Electromagn. Compat. (EMC)*, vol. 3, Portland, OR, USA, Aug. 2006, pp. 646–650.

[27] R. Rimolo-Donadio, H.-D. Brüns, and C. Schuster, "Including stripline connections into network parameter based via models for fast simulation of interconnects," in *Int. Zurich Symp. Electromagnetic Compatibility*, Zurich, Switzerland, Jan. 2009.

[28] A. Arsenovic et al. (2014) scikit-rf. [Online]. Available: http://www.scikit-rf.org

[29] S. Zhang and J.-M. Jin, *Computation of Special Functions*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1996.