Qucs Help Documentation

Release 0.0.18

Qucs Team (2014)

February 22, 2016

Contents

Contents:

CHAPTER 1

Getting Started with Analog Simulations

Qucs (speak: kju:ks) is a circuit simulator with graphical user interface. It is able to perform many different kinds of simulation (e.g. DC, s parameter). This document should give you a short description on how to use Qucs.

When you start Qucs the first time, it creates the directory ".qucs" within your home directory. Every file is saved into this directory or into one of its subdirectories. After Qucs has been loaded, one sees the main window looking simular like the one in figure 1. On the right side, there is the working area (6) containing the schematics, data displays and so on. Using the tabular bar (5) above this area, you can quickly switch to every document currently open. On the left side of the Qucs main window, there is another area (1), whose content depend on the status of the above-lying tabular bar: "Projects" (2), "Content" (3) and "Components" (4). After running Qucs, the "Projects" (2) tab is activated. As it is the first time you started this program, the area is empty because you haven't yet any project. Press the "New" button right above the area (1) and a dialog opens. Enter the name for your first project, e.g. "firstProject" and press the "Ok" button. Qucs creates a project directory into the ~/.qucs directory, for this example "firstProject_prj". Every file belonging to this new project will be saved within this directory. The new project is immediately opened (as can be read on the window title bar) and the tabular bar is switched to "Content" (3), where the content of the currently opened project is displayed. You do not yet have any document, so press save button on the toolbar (or use the main menu: File->Save) in order to save the untitled document that still fills the working area (6). You will be ask for the name of your new document. Enter "firstSchematic" and press the "Ok" button.

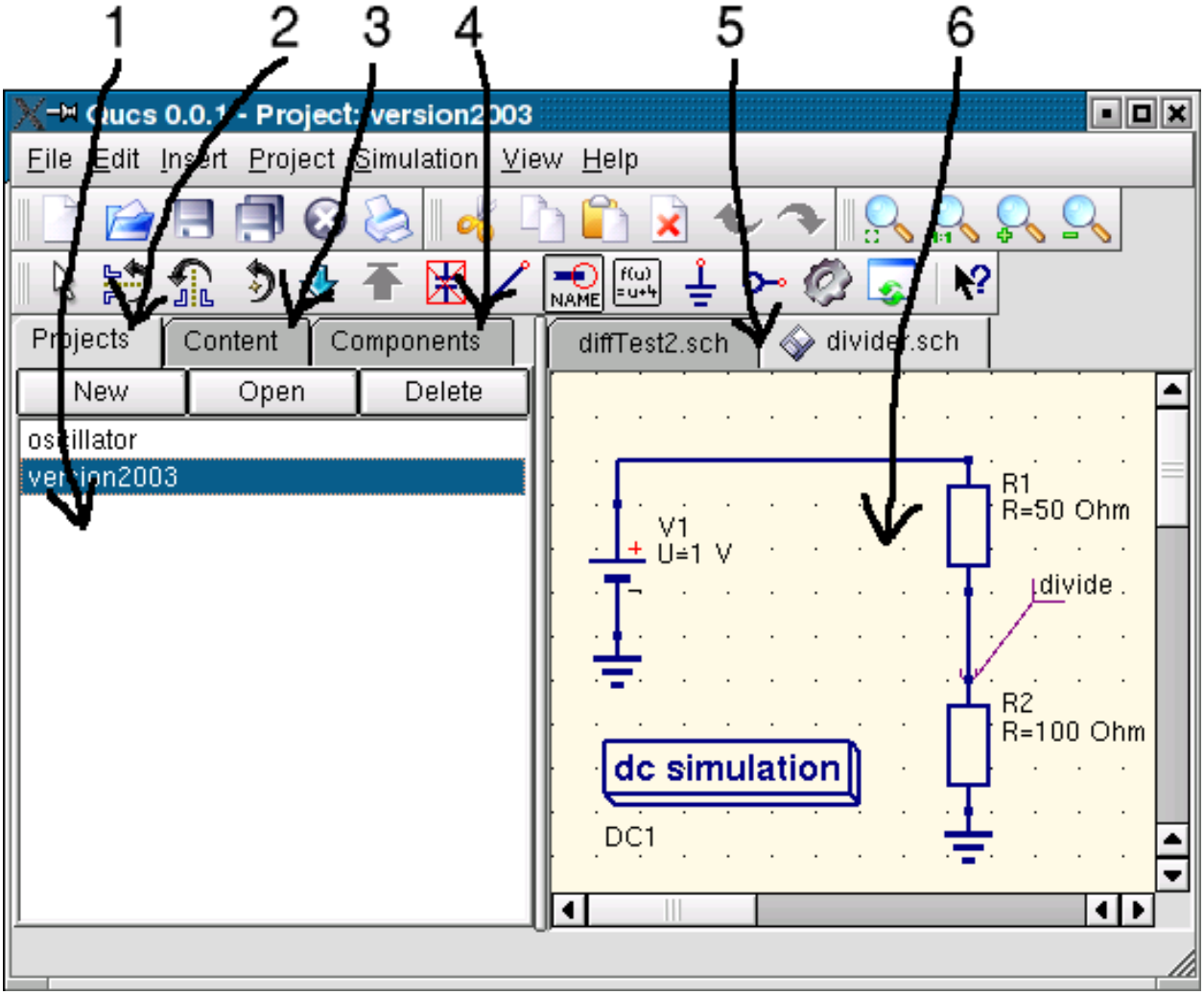


Figure 1 - Qucs main window

Now we want to make a simple DC simulation, i.e. we want to analyse the circuit in figure (1). Activate the "Components" tab ((4) in figure 1). There, you see a combo box where you can choose a component group and, below, the components of the chosen component group. Choose "lumped components" and click on the first symbol: "Resistor". Moving the mouse cursor into the working area (6) you are carrying a drawing of a resistor symbol. Pressing the right mouse button rotates the symbol, pressing the left mouse button places the component onto the schematic. Repeat this process for all components shown in figure 1. The voltage source can be found in the "sources" component class, the ground symbol can be taken from "lumped components" class or from the toolbar, the wanted simulation is defined by the big simulation blocks found in the "simulations" component class. To edit the parameters of the second resistor, double-click on it. A dialog opens where you can change the resistance. Enter "100 Ohm" into the edit field on the right side and press enter.

To connect the components, press the wire toolbar button (or use the main menu: Insert->Wire). Move the cursor onto an open port (marked by the small red circles). Clicking on it starts the wire. Now move to the end point and click again. The components are now connected. If you want to change the corner direction of the wire, click on the right mouse button before setting the end point. You can also end a wire without pressing on an open port or on a wire: Just double-click the left mouse button.

Last but not least, you must label the node where you want Qucs to caculate the voltage. Press on the label toolbar button (or use the menu: Insert->Wire Label). Now click on the chosen wire. A dialog opens and you can enter the node name. Type "divide" and click the "Ok" button. Now the circuit should look like the one in figure 1.

To start the simulation press the simulate toolbar button (or use the menu: Simulation->Simulate). A window opens and shows the progress. After finishing the simulation successfully, the data display is opened. Normally, all this happens so fast that you only see a short flickering. Now you have to place a diagram to see the simulation results. On the left side the “diagrams” component class has already chosen automatically. Press on the “Tabular” item, move to the working area and place it by clicking the left mouse button. A dialog opens where you can choose what should be displayed by the new diagram. In the left area you see the node name you have defined: “divide”. Double-click on it and it will be transfered to the right area. Leave the dialog by clicking the “Ok” button. Now you see the simulation result: 0.666667 volts. Wonderful, give yourself a clap on the shoulder!

CHAPTER 2

Getting Started with Digital Simulations

Qucs is also a graphical user interface for performing digital simulations. This document should give you a short description on how to use it.

For digital simulations Qucs uses the FreeHDL program (http://www.freehdl.seul.org). So the FreeHDL package as well as the GNU C++ compiler must be installed on the computer.

There is no big difference in running an analog or a digital simulation. So having read the Getting Started for analog simulations, it is now easy to get a digital simulation work. Let us compute the truth table of a simple logical AND cell. Select the digital components in the combobox of the components tab on the left-hand side and build the circuit shown in figure 1. The digital simulation block can be found among the other simulation blocks.

The digital sources *S1* and *S2* are the inputs, the node labeled as *Output* is the output. After performing the simulation, the data display page opens. Place the diagram *truth table* on it and insert the variable *Output*. Now the truth table of a two-port AND cell is shown. Congratulation, the first digital simulation is done!

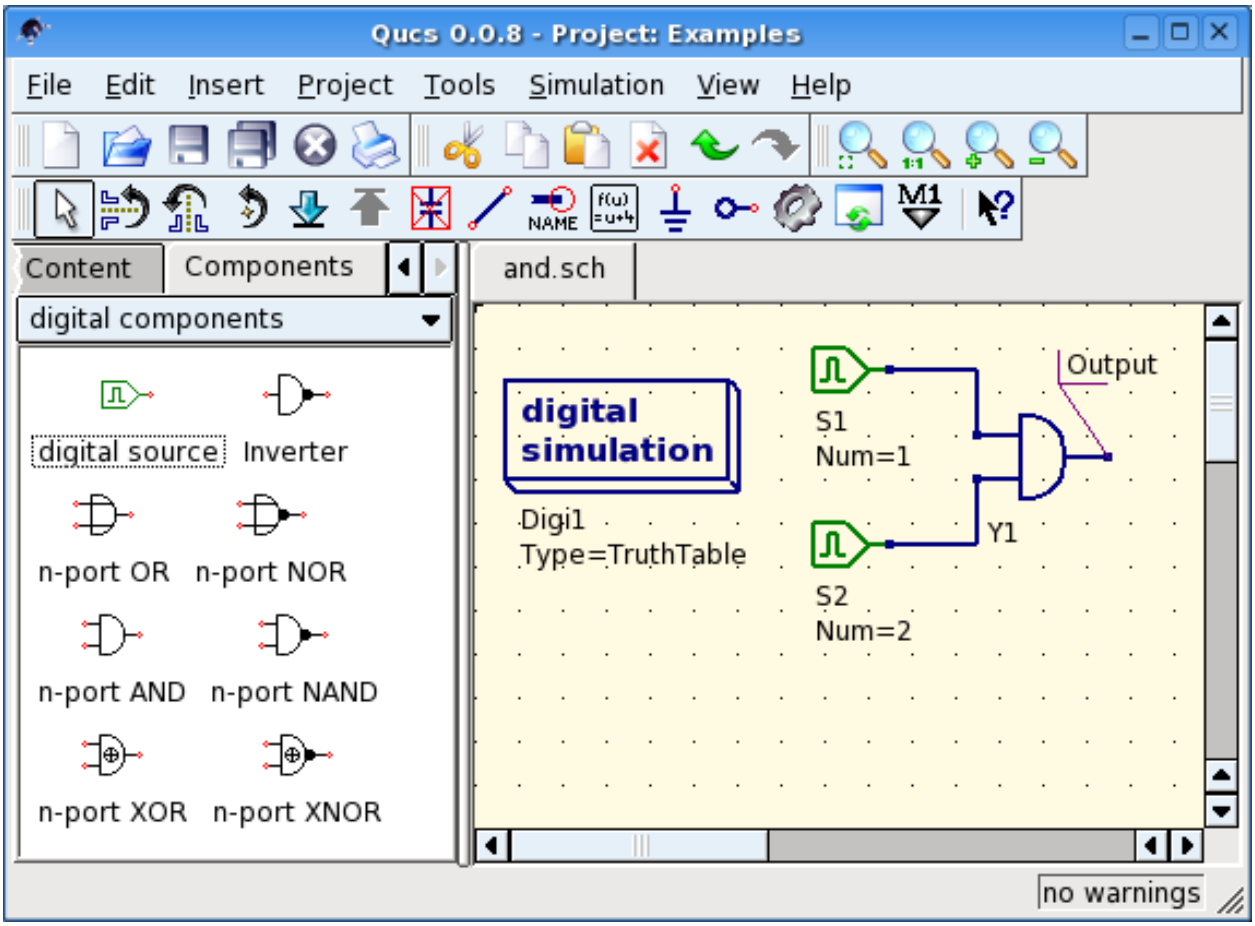


Figure 1 - Qucs main window

A truth table is not the only digital simulation that Qucs can perform. It is also possible to apply an arbitrary signal to a circuit and see the output signal in a timing diagram. To do so, the parameter *Type* of the simulation block must be changed to *TimeList* and the duration of the simulation must be entered in the next parameter. The digital sources have now a different meaning: They can output an arbitrary bit sequence by defining the first bit (low or high) and a list that sets the times until the next change of state. Note that this list repeats itself after its end. So, to create a 1GHz clock with pulse ratio 1:1, the list writes: 0.5ns; 0.5ns

To display the result of this simulation type, there is the diagram *timing diagram*. Here, the result of all output nodes can be shown row by row in one diagram. So, let's have fun...

2.1 VHDL File Component

More complex and more universal simulations can be performed using the component "VHDL file". This component can be picked up from the component list view (section "digital components"). Nevertheless the recommended usage is the following: The VHDL file should be a member of the project. Then go to the content list view and click on the file name. After entering the schematic area, the VHDL component can be placed.

The last entity block in the VHDL file defines the interface, that is, all input and output ports must be declared here. These ports are also shown by the schematic symbol and can be connected to the rest of the circuit. During simulation the source code of the VHDL file is placed into the top-level VHDL file. This must be considered as it causes some limitations. For example, the entity names within the VHDL file must differ from names already given to subcircuits. (After a simulation, the complete source code can be seen by pressing F6. Use it to get a feeling for this procedure.)

CHAPTER 3

Getting Started with Optimization

For circuit optimization, Qucs uses the ASCO tool (http://asco.sourceforge.net/). A brief description on how to prepare your schematic, execute and interprete the results is given below. Before using this functionality, ASCO must be installed on the computer.

Optimization of a circuit is nothing more than the minimization of a cost function. It can either be the delay or the rise time of a digital circuit, or the power or gain of an analog circuit. Another possibility is defining the optimization problem as a composition of functions, leading in this case to the definition of a figure-of-merit.

To setup a netlist for optimization two things must be added to the already existing netlist: insert equation(s) and the optimization component block. Take the schematic from Figure 1 and change it until you have the resulting schematic given in Figure 2.

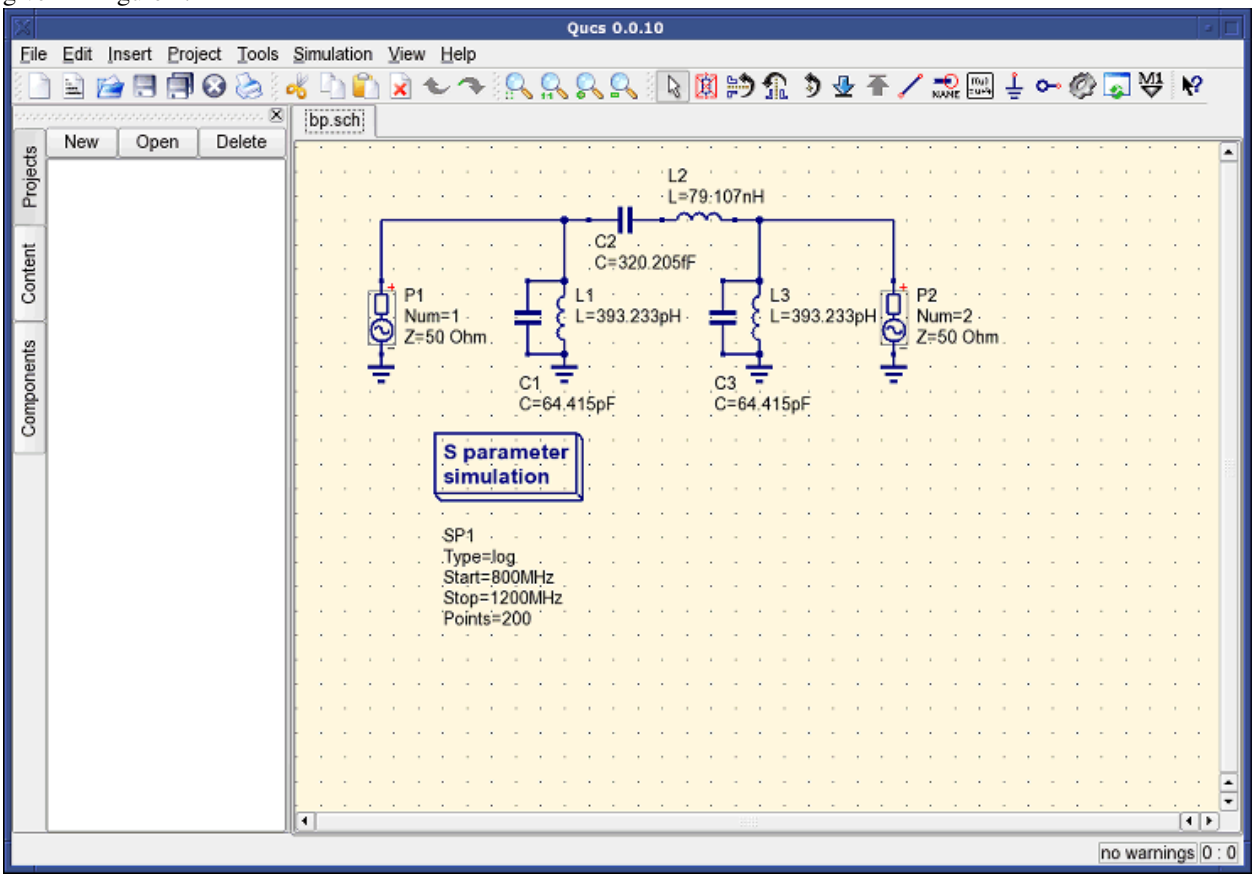


Figure 1 - Initial schematic.

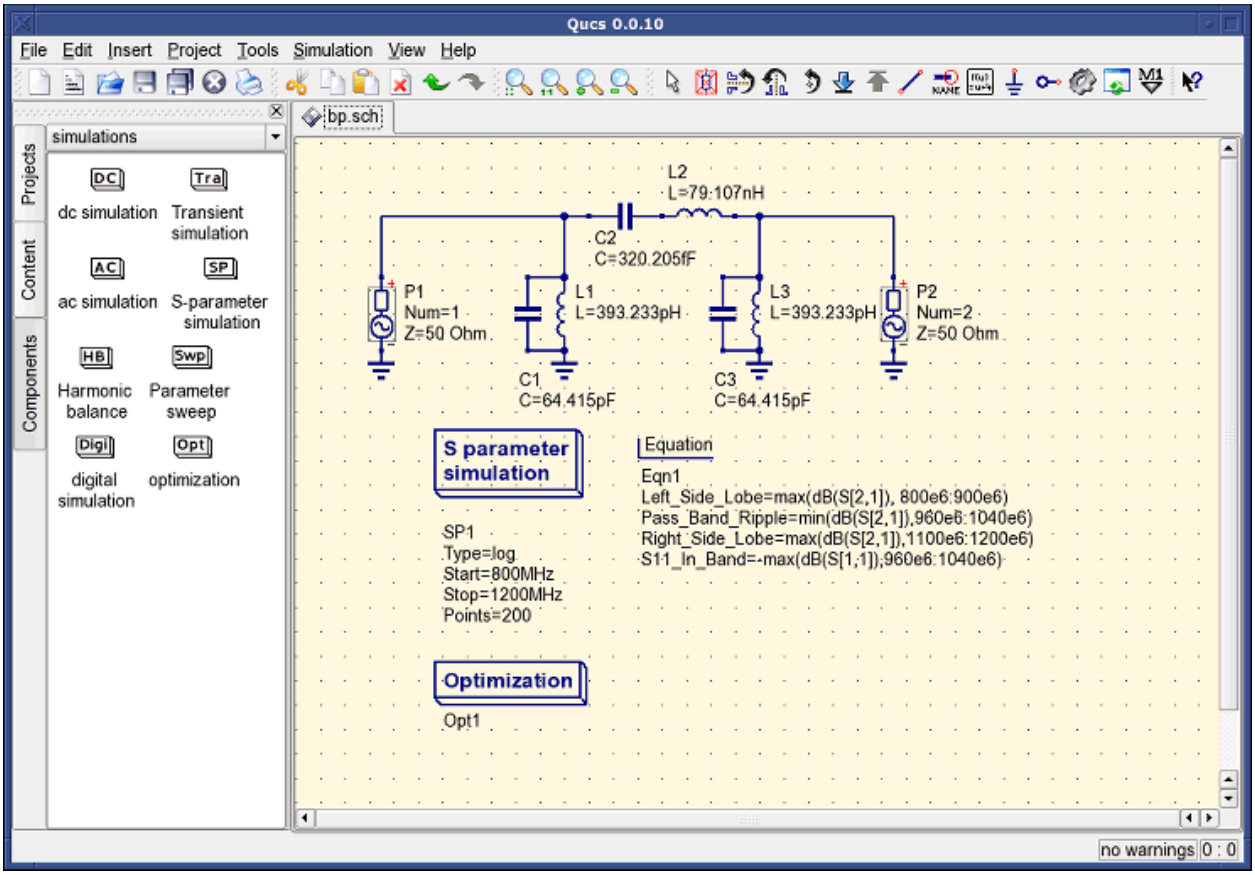


Figure 2 - Prepared schematic.

Now, open the optimization component and select the optimization tab. From the existing parameters, special attention should be paid to ‘Maximum number of iterations’, ‘Constant F’ and ‘Crossing over factor’. Over- or underestimation can lead to a premature convergence of the optimizer to a local minimum or, a very long optimization time.

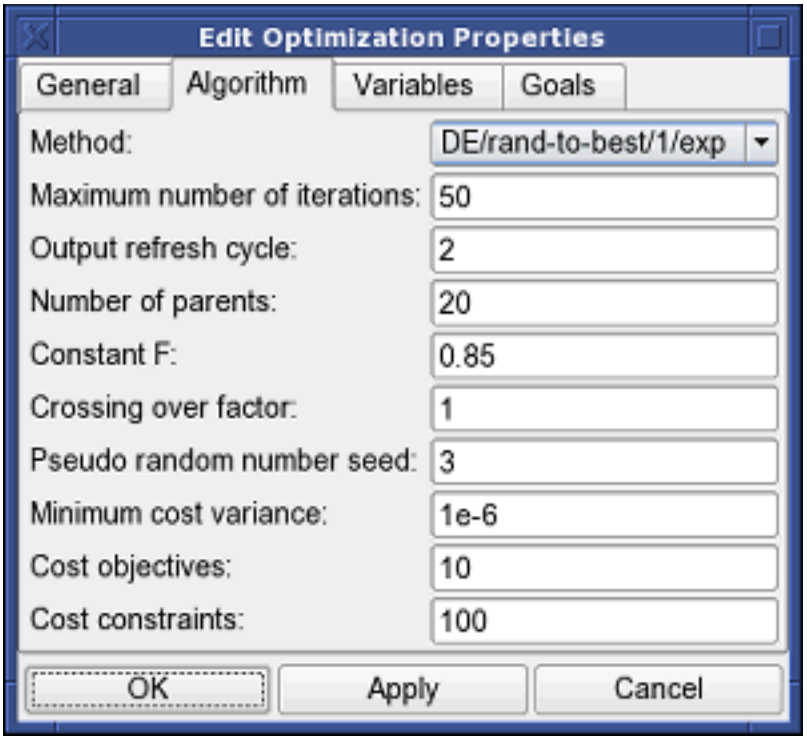


Figure 3 - Optimization dialog, algorithm options.

In the Variables tab, defining which circuit elements will be chosen from the allowed range, as shown in Figure 4. The variable names correspond to the identifiers placed into properties of components and **not** the components' names.

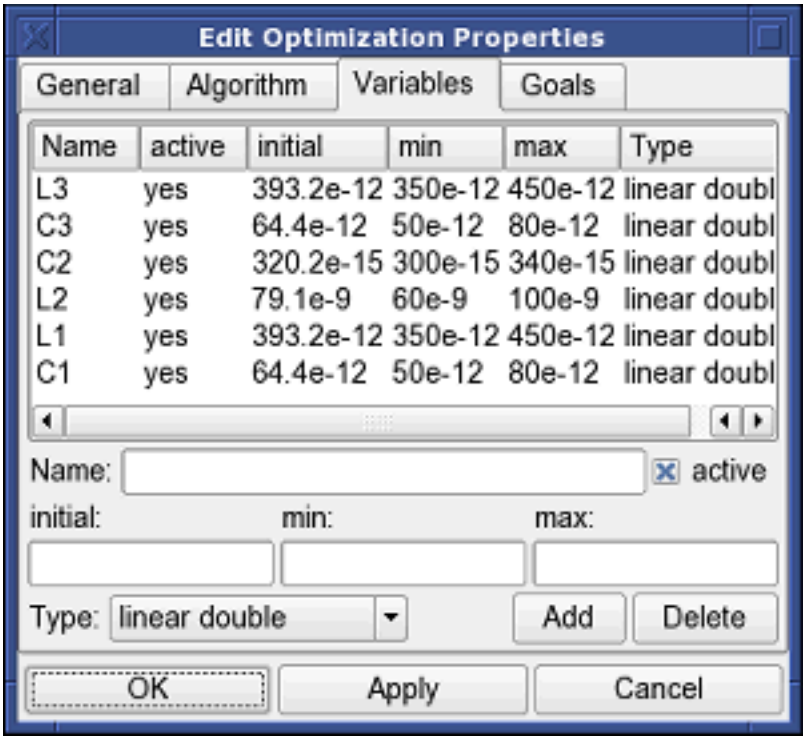


Figure 4 - Optimization dialog, variables options.

Finally, go to Goals where the optimization objective (maximize, minimize) and constraints (less, greater, equal) are

defined. ASCO then automatically combines them into a single cost function, that is then minimized.

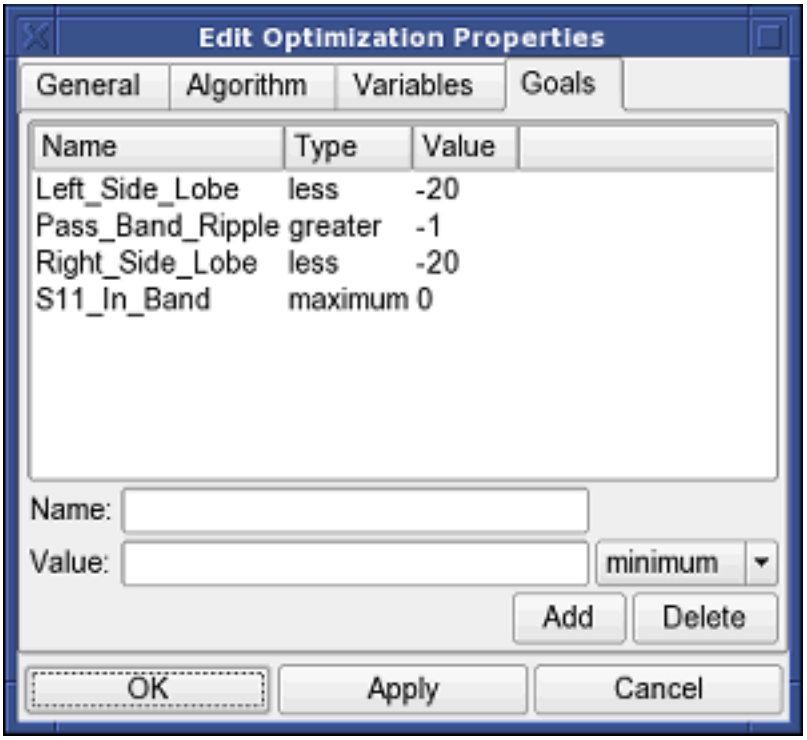


Figure 5 - Optimization dialog, goals options.

The next step is to change the schematic, and define which circuit elements are to be optimized. The resulting schematic is show in Figure 6.

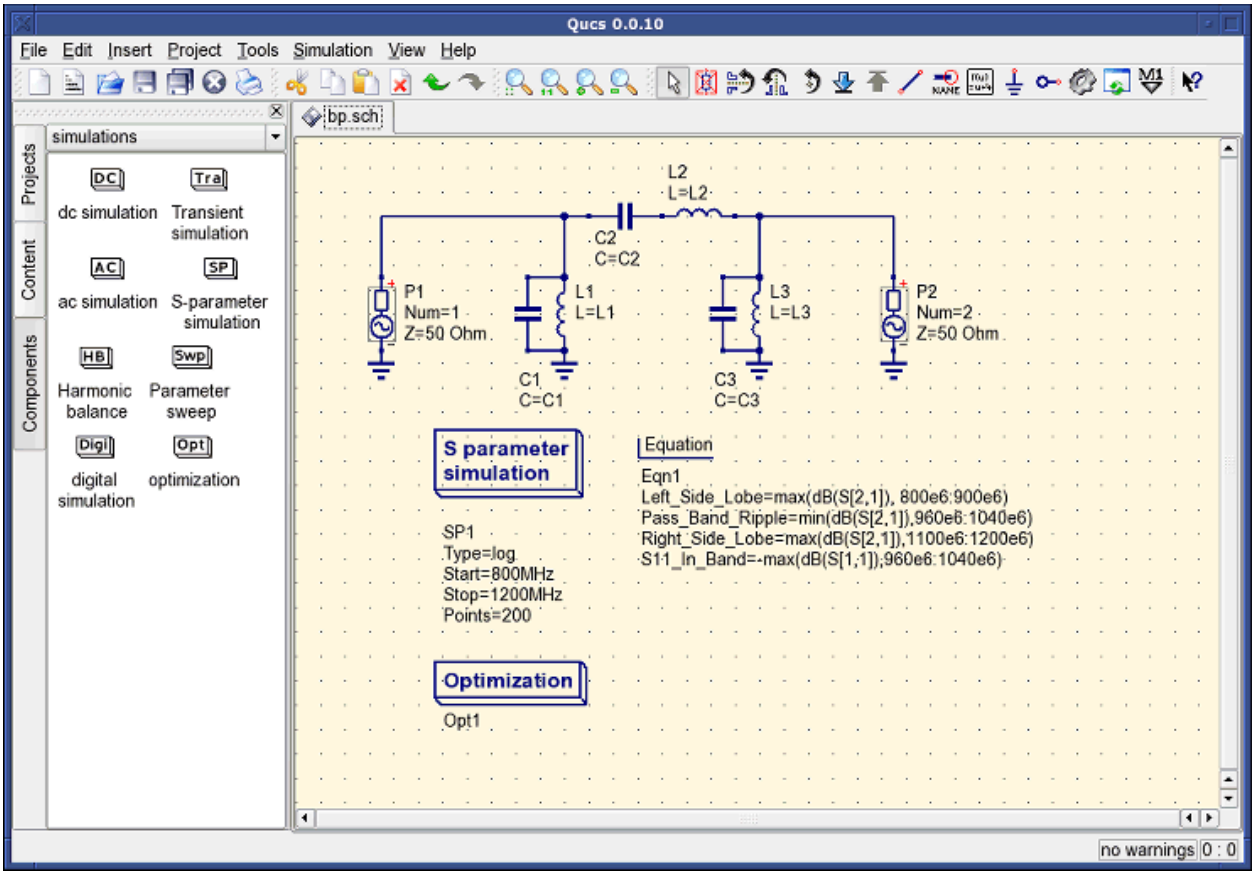


Figure 6 - New Qucs main window.

The last step is to run the optimization, i.e. the simulation by pressing F2. Once finished, which takes a few seconds on a modern computer, the best simulation results is shown in the graphical waveform viewer.

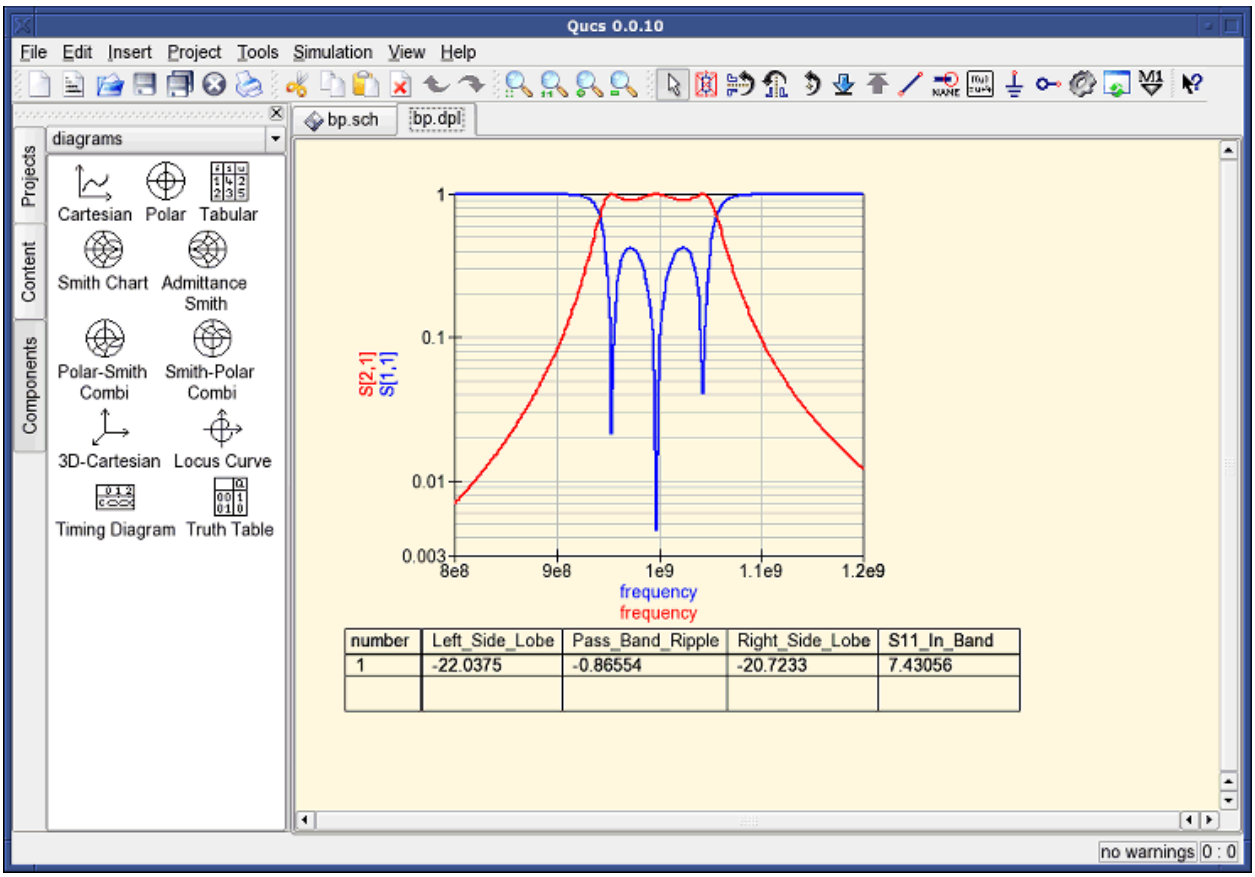


Figure 7 - Qucs results window.

The best found circuit sizes can be found in the optimization dialog, in the Variables tab. They are now the initial values for each one of introduced variables (Figure 8).

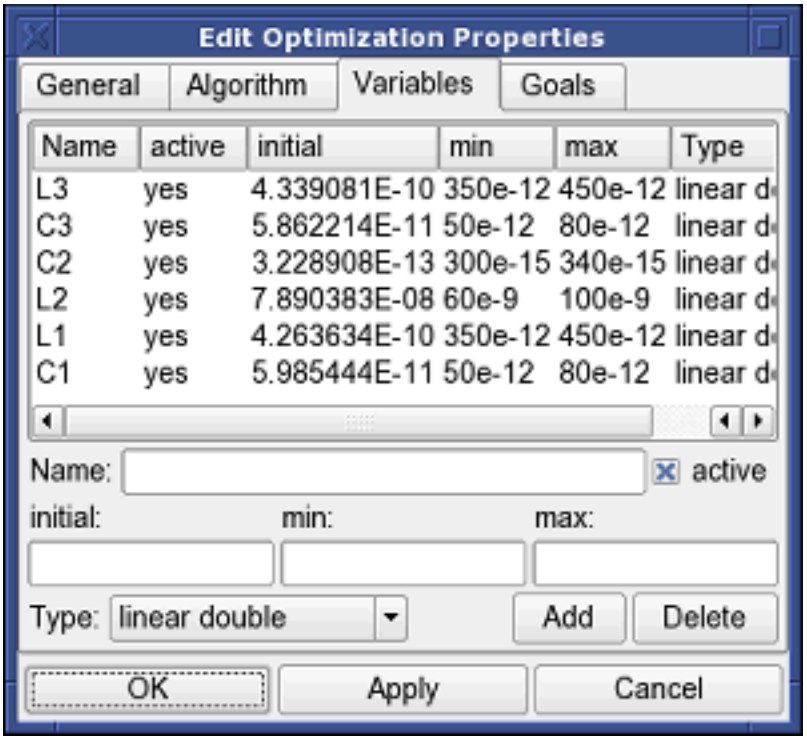


Figure 8 - The best found circuit sizes.

CHAPTER 4

Getting Started with Octave Scripts

Qucs can also be used to develop Octave scripts (see http://www.octave.org). This document should give you a short description on how to do this.

If the user creates a new text document and saves it with the Octave extension, e.g. 'name.m' then the file will be listed at the Octave files of the active project. The script can be executed with F2 key or by pressing the simulate button in the toolbar. The output can bee seen in the Octave window that opens automatically (per default on the right-hand side). At the bottom of the Octave window there is a command line where the user can enter single commands. It has a history function that can be used with the cursor up/down keys.

There are two Octave functions that load Qucs simulation results from a dataset file: loadQucsVariable() and loadQucsDataset(). Please use the help function in the Octave command line to learn more about them (i.e. type help loadQucsVariable and help loadQucsDataset).

4.1 Postprocessing

Octave can also be used for automatic postprocessing of a Qucs simulation result. This is done by editing the data display file of a schematic (Document Settings... in File menu). If the filename of an Octave script (filename extension m) from the same project is entered, this script will be executed after the simulation is finished.

CHAPTER 5

Short Description of Actions

5.1 General Actions

(valid in all modes)

mouse wheel	Scrolls vertically the drawing area. You can also scroll outside the current size.
mouse wheel + Shift Button	Scrolls horizontally the drawing area. You can also scroll outside the current size.
mouse wheel + Ctrl Button	Zooms into or outof the drawing area.
drag'n'drop file into document area	Tries to open file as Qucs schematic or data display.

5.2 "Select"-Mode

(Menu: Edit->Select)

left mouse button	Selects the element below the mouse cursor. If several components are placed there, you can clicking several times in order to select the wanted one. Keeping the mouse button pressed, you can move the component below the mouse cursor and all selected ones. If you want to fine position the components, press the CTRL key during moving and the grid is disabled. Keeping the mouse button pressed without any element below it opens a rectangle. After releasing the mouse button, all elements within this rectangle are selected. A selected diagram or painting can be resized by pressing the left mouse button over one of its corners and moving by keeping the button pressed. After clicking on a component text, it can be edited directly. The enter key jumps to the next property. If the property is a selection list, it can only be changed with the cursor up/down keys. Clicking on a circuit node enters the "wire mode".
left mouse button + Ctrl Button	Allows more than one element to be selected, i.e. selecting an element does not deselect the others. Clicking on a selected element deselects it. This mode is also valid for selecting by opening a rectangle (see item before).
right mouse button	Clicking on a wire selects a single straight line instead of the complete line.
double-click right mouse button	Opens a dialog to edit the element properties (The labels of wires, the parameters of components, etc.).

5.3 “Insert Component”-Mode

(Click on a component/diagram in the left area)

left mouse button	Place a new instance of the component onto the schematic.
right mouse button	Rotate the component. (Has no effect on diagrams.)

5.4 “Wire”-Mode

(Menu: Insert->Wire)

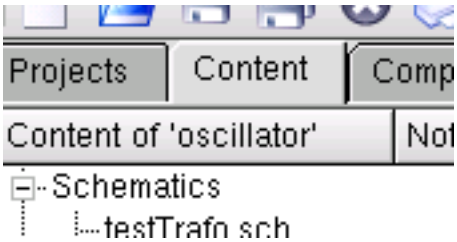
left mouse button	Sets the starting/ending point of the wire.
right mouse button	Changes the direction of the wire corner (first left/right or first up/down).
double-click right mouse button	Ends a wire without being on a wire or a port.

5.5 “Paste”-Mode

(Menu: Edit->Paste)

left mouse button	Place the elements onto the schematic (from the clipboard).
right mouse button	Rotate the elements.

5.6 Mouse in “Content” Tab



left click	Selects file.	
double-click	Opens file.	
right click	Displays menu with:	
	“open”	• open selected file
	“rename”	• change name of selected file
	“delete”	• delete selected file
	“delete group	• delete selected file and its relatives (schematic, data display, dataset)

5.7 Keyboard

Many actions can be activated/done by the keyboard strokes. This can be seen in the main menu right beside the command. Some further key commands are shown in the following list:

“Delete” or “Backspace”	Deletes the selected elements or enters the delete mode if no element is selected.
Cursor left/right	Changes the position of selected markers on their graphs. If no marker is selected, move selected elements. If no element is selected, scroll document area.
Cursor up/down	Changes the position of selected markers on more-dimensional graphs. If no marker is selected, move selected elements. If no element is selected, scroll document area.
Tabulator	Changes to the next open document (according to the TabBar above).

CHAPTER 6

Working with Subcircuits

Subcircuits are used to bring more clarity into a schematic. This is very useful in large circuits or in circuits, in which a component block appears several times.

In Qucs, each schematic containing a subcircuit port is a subcircuit. You can get a subcircuit port by using the toolbar, the components listview (in lumped components) or the menu (Insert->Insert port). After placing all subcircuit ports (two for example) you need to save the schematic (e.g. CTRL-S). By taking a look into the content listview (figure 1) you see that now there is a “2-port” right beside the schematic name (column “Note”). This note marks all documents which are subcircuits. Now change to a schematic where you want to use the subcircuit. Then press on the subcircuit name (content listview). By entering the document area again, you see that you now can place the subcirciut into the main circuit. Do so and complete the schematic. You can now perform a simulation. The result is the same as if all the components of the subcircuit are placed into the circuit directly.

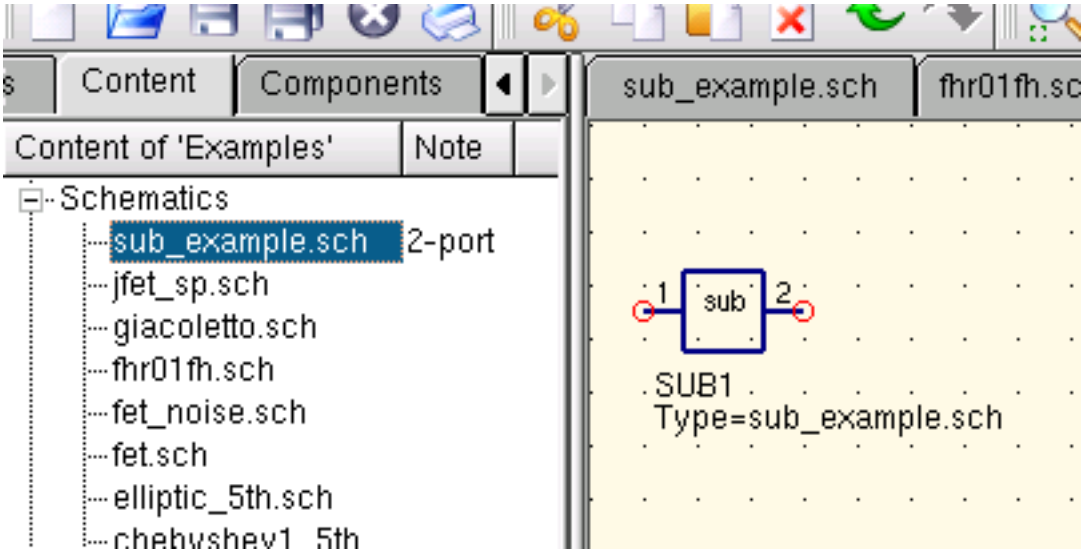


Figure 1 - Accesing a subcircuit

If you select a subcircuit component (click on its symbol in the schematic) you can step into the subcircuit schematic by pressing CTRL-I (of course, this function is also reachable via toolbar and via menu). You can step back by pressing CTRL-H.

If you do not like the component symbol of a subcircuit, you can draw your own symbol and put the component text at your favourite position. Just make the subcircuit schematic the current and go to the menu: File->Edit Circuit Symbol. If you have not yet drawn a symbol for this circuit. A simple symbol is created automatically. You now can edit this symbol by painting lines and arcs. After finished, save it. Now place it on another schematic, and you have a new symbol.

Just like all other components, subcircuits can have parameters. To create your own parameters, go back to the editor where you edited the subcircuit symbol and double-click on the subcircuit parameter text (SUB1 in the Figure 1). A dialog apperas and you now can fill in parameterswith default values and descriptions. When you are ready, close the dialog and save the subcircuit. In every schematic where the subcircuit is placed, it owns the new parameters which can be edited as in all other components.

6.1 Subcircuits with Parameters

A simple example using subcircuits with parameters and equations is provided here.

Create a subcircuit:

- Create a new project
- New schematic (for subcircuit)
- Add a resistor, inductor, and capacitor, wire them in series, add two ports
- Save the subcircuit as RLC.sch
- Give value of resistor as 'R1'
- Add equation 'ind = L1',
- Give value of inductor as 'ind'
- Give value of capacitor as 'C1'
- Save
- File > Edit Circuit Symbol
- Double click on the 'SUB File=name' tag under the rectangular box
  - Add name = R1, default value = 1
  - Add name = L1, default value = 1
  - Add name = C1, default value = 1
  - OK

Insert subcircuit and define parameters:

- New schematic (for testbench)
- Save Test_RLC.sch
- Project Contents > pick and place the above RLC subcircuit
- Add AC voltage source (V1) and ground
- Add AC simulation, from 140Hz to 180Hz, 201 points
- Set on the subcircuit symbol
  - R1=1
  - L1=100e-3
  - C1=10e-6
- Simulate
- Add a Cartesian diagram, plot V1.i
- The result should be the resonance of the RLC circuit.

- The parameters of the RLC subcircuit can be changed on the top schematic.

CHAPTER 7

Short Description of Mathematical Functions

The following operations and functions can be applied in Qucs equations. For detailed description please refer to the "Measurement Expressions Reference Manual". Parameters in brackets "[]" are optional.

7.1 Operators

7.1.1 Arithmetic Operators

+x	Unary plus
-x	Unary minus
x+y	Addition
x-y	Subtraction
x*y	Multiplication
x/y	Division
x%y	Modulo (remainder of division)
x^y	Power

7.1.2 Logical Operators

!x	Negation
x&&y	And
x||y	Or
x^^y	Exclusive or
x?y:z	Abbreviation for conditional expression "if x then y else z
x==y	Equal
x!=y	Not equal
x<y	Less than
x<=y	Less than or equal
x>y	Larger than
x>=y	Larger than or equal

7.2 Math Functions

7.2.1 Vectors and Matrices: Creation

`eye(n)`	Creates `n` x `n` identity matrix
`length(y)`	Returns the length of the given vector
`linspace(from,to,n)`	Creates a real vector with `n` linearly spaced components between `from` and `to`, both inclusively
`logspace(from,to,n)`	Creates a real vector with `n` logarithmically spaced components between `from` and `to`, both inclusively

7.2.2 Vectors and Matrices: Basic Matrix Functions

`adjoint(x)`	Adjoint matrix of `x` (transposed and conjugate complex)
`det(x)`	Determinant of a matrix `x`
`inverse(x)`	Inverse matrix of `x`
`transpose(x)`	Transposed matrix of `x` (rows and columns exchanged)

7.2.3 Elementary Mathematical Functions: Basic Real and Complex Functions

`abs(x)`	Absolute value, magnitude of complex number
`angle(x)`	Phase angle in radians of a complex number. Synonym for `arg()`
`arg(x)`	Phase angle in radians of a complex number
`conj(x)`	Conjugate of a complex number
`deg2rad(x)`	Converts phase from degrees into radians
`hypot(x,y)`	Euclidean distance function
`imag(x)`	Imaginary part of a complex number
`mag(x)`	Magnitude of a complex number
`norm(x)`	Square of the absolute value of a vector
`phase(x)`	Phase angle in degrees of a complex number
`polar(m,p)`	Transform from polar coordinates (magnitude `m`, phase `p`) into complex number
`rad2deg(x)`	Converts phase from radians into degrees
`real(x)`	Real part of a complex number
`sign(x)`	Signum function
`sqr(x)`	Square (power of two) of a number
`sqrt(x)`	Square root
`unwrap(p[,tol[,step]])`	Unwraps the angle `p` (in radians – default `step` is 2*pi) using the optional tolerance value `tol` (default is pi)

7.2.4 Elementary Mathematical Functions: Exponential and Logarithmic Functions

`exp(x)`	Exponential function to basis e
`limexp(x)`	Limited exponential function
`log10(x)`	Decimal logarithm
`log2(x)`	Binary logarithm
`ln(x)`	Natural logarithm (base e)

7.2.5 Elementary Mathematical Functions: Trigonometry

`cos(x)`	Cosine function
`cosec(x)`	Cosecant
`cot(x)`	Cotangent function
`sec(x)`	Secant
`sin(x)`	Sine function
`tan(x)`	Tangent function

7.2.6 Elementary Mathematical Functions: Inverse Trigonometric Functions

`arccos(x)`	Arc cosine (also known as “inverse cosine”)
`arccosec(x)`	Arc cosecant
`arccot(x)`	Arc cotangent
`arcsec(x)`	Arc secant
`arcsin(x)`	Arc sine (also known as “inverse sine”)
`arctan(x[,y])`	Arc tangent (also known as “inverse tangent”)

7.2.7 Elementary Mathematical Functions: Hyperbolic Functions

`cosh(x)`	Hyperbolic cosine
`cosech(x)`	Hyperbolic cosecant
`coth(x)`	Hyperbolic cotangent
`sech(x)`	Hyperbolic secant
`sinh(x)`	Hyperbolic sine
`tanh(x)`	Hyperbolic tangent

7.2.8 Elementary Mathematical Functions: Inverse Hyperbolic Functions

`arcosh(x)`	Hyperbolic area cosine
`arcosech(x)`	Hyperbolic area cosecant
`arcoth(x)`	Hyperbolic area cotangent
`arsech(x)`	Hyperbolic area secant
`arsinh(x)`	Hyperbolic area sine
`artanh(x)`	Hyperbolic area tangent

7.2.9 Elementary Mathematical Functions: Rounding

`ceil(x)`	Round to the next higher integer
`fix(x)`	Truncate decimal places from real number
`floor(x)`	Round to the next lower integer
`round(x)`	Round to nearest integer

7.2.10 Elementary Mathematical Functions: Special Mathematical Functions

`besseli0(x)`	Modified Bessel function of order zero
`besselj(n,x)`	Bessel function of first kind and `n`-th order
`bessely(n,x)`	Bessel function of second kind and `n`-th order
`erf(x)`	Error function
`erfc(x)`	Complementary error function
`erfinv(x)`	Inverse error function
`erfcinv(x)`	Inverse complementary error function
`sinc(x)`	Sinc function (sin(`x`)/`x` or 1 at `x` = 0)
`step(x)`	Step function

7.2.11 Data Analysis: Basic Statistics

`avg(x[,range])`	Arithmetic average of vector elements; if a range is given then `x` must have a single data dependency
`cumavg(x)`	Cumulative average of vector elements
`max(x,y)`	Returns the greater of the values `x` and `y`
`max(x[,range])`	Maximum value in vector; if a range is given then `x` must have a single data dependency
`min(x,y)`	Returns the lesser of the values `x` and `y`
`min(x[,range])`	Minimum value in vector; if a range is given then `x` must have a single data dependency
`rms(x)`	Root Mean Square of vector elements
`runavg(x)`	Running average of vector elements
`stddev(x)`	Standard deviation of vector elements
`variance(x)`	Variance of vector elements
`random()`	Random number between 0.0 and 1.0
`srandom(x)`	Give random seed

7.2.12 Data Analysis: Basic Operation

`cumprod(x)`	Cumulative product of vector elements
`cumsum(x)`	Cumulative sum of vector elements
`interpolate(f[,n])`	Equidistant spline interpolation of real function vector `f(x)` using `n` equidistant datapoints; the latter can be omitted and defaults to a reasonable value
`prod(x)`	Product of vector elements
`sum(x)`	Sum of vector elements
`xvalue(f,yval)`	Returns the x-value which is associated with the y-value nearest to a specified y-value `yval` in a given vector `f`; therefore the vector `f` must have a single data dependency
`yvalue(f,xval)`	Returns the y-value of the given vector `f` which is located nearest to the x-value `xval`; therefore the vector `f` must have a single data dependency

7.2.13 Data Analysis: Differentiation and Integration

`ddx(expr,var)`	Derives mathematical expression `expr` with respect to the variable `var`
`diff(y,x[,n])`	Differentiate vector `y` with respect to vector `x` `n` times. If `n` is omitted it defaults to `n` = 1
`integrate(x,h)`	Integrate vector `x` numerically assuming a constant step-size `h`

7.2.14 Data Analysis: Signal Processing

`dft(x)`	Discrete Fourier Transform of vector `x`
`fft(x)`	Fast Fourier Transform of vector `x`
`fftshift(x)`	Shuffles the FFT values of vector `x` in order to move the frequency 0 to the center of the vector
`Freq2Time(V,f)`	Inverse Discrete Fourier Transform of function `V(f)` interpreting it physically
`idft(x)`	Inverse Discrete Fourier Transform of vector `x`
`ifft(x)`	Inverse Fast Fourier Transform of vector `x`
`kbd(x[,n])`	Kaiser-Bessel derived window
`Time2Freq(v,t)`	Discrete Fourier Transform of function `v(t)` interpreting it physically

7.3 Electronics Functions

7.3.1 Unit Conversion

`dB(x)`	dB value
`dbm(x)`	Convert voltage to power in dBm
`dbm2w(x)`	Convert power in dBm to power in Watts
`w2dbm(x)`	Convert power in Watts to power in dBm
`vt(t)`	Thermal voltage for a given temperature `t` in Kelvin

7.3.2 Reflection Coefficients and VSWR

`rtoswr(x)`	Converts reflection coefficient to voltage standing wave ratio (VSWR)
`rtoy(x[,zref])`	Converts reflection coefficient to admittance; by default reference `zref` is 50 ohms
`rtoz(x[,zref])`	Converts reflection coefficient to impedance; by default reference `zref` is 50 ohms
`ytor(x[,zref])`	Converts admittance to reflection coefficient; by default reference `zref` is 50 ohms
`ztor(x[,zref])`	Converts impedance to reflection coefficient; by default reference `zref` is 50 ohms

7.3.3 N-Port Matrix Conversions

`stos(s,zref[,z0])`	Converts S-parameter matrix to S-parameter matrix with different reference impedance(s)
`stoy(s[,zref])`	Converts S-parameter matrix to Y-parameter matrix
`stoz(s[,zref])`	Converts S-parameter matrix to Z-parameter matrix
`twoport(m,from,to)`	Converts a two-port matrix from one representation into another, possible values for `from` and `to` are 'Y', 'Z', 'H', 'G', 'A', 'S' and 'T'.
`ytos(y[,z0])`	Converts Y-parameter matrix to S-parameter matrix
`ytoz(y)`	Converts Y-parameter matrix to Z-parameter matrix
`ztos(z[,z0])`	Converts Z-parameter matrix to S-parameter matrix
`ztoy(z)`	Converts Z-parameter matrix to Y-parameter matrix

7.3.4 Amplifiers

`GaCircle(s,Ga[,arcs])`	Circle(s) with constant available power gain `Ga` in the source plane
`GpCircle(s,Gp[,arcs])`	Circle(s) with constant operating power gain `Gp` in the load plane
`Mu(s)`	Mu stability factor of a two-port S-parameter matrix
`Mu2(s)`	Mu' stability factor of a two-port S-parameter matrix
`NoiseCircle(Sopt,Fmin,Rn[,arcs])`	Generate circle(s) with constant Noise Figure(s) `F`. `Arcs` specifies the angles in degrees created by e.g. `linspace(0,360,100)`. If Arcs is a number it specifies the number of equally spaced circle segments, if it is omitted this number defaults to a reasonable value
`PlotVs(data,dep)`	Returns a data item based upon vector or matrix vector `data` with dependency on a given vector `dep`, e.g. `PlotVs(Gain,frequency/1e9)`
`Rollet(s)`	Rollet stability factor of a two-port S-parameter matrix
`StabCircleL(s[,arcs])`	Stability circle in the load plane
`StabCircleS(s[,arcs])`	Stability circle in the source plane
`StabFactor(s)`	Stability factor of a two-port S-parameter matrix. Synonym for `Rollet()`
`StabMeasure(s)`	Stability measure B1 of a two-port S-parameter matrix

7.4 Nomenclature

7.4.1 Ranges

`LO:HI`	Range from `LO` to `HI`
`:HI`	Up to `HI`
`LO:`	From `LO`
`:`	No range limitations

7.4.2 Matrices and Matrix Elements

`M`	The whole matrix `M`
`M[2,3]`	Element being in 2nd row and 3rd column of matrix `M`
`M[:,3]`	Vector consisting of 3rd column of matrix `M`

7.4.3 Immediate

`2.5`	Real number
`1.4+j5.1`	Complex number
`[1,3,5,7]`	Vector
`[11,12;21,22]`	Matrix

7.4.4 Number suffixes

E	exa, 1e+18
P	peta, 1e+15
T	tera, 1e+12
G	giga, 1e+9
M	mega, 1e+6
k	kilo, 1e+3
m	milli, 1e-3
u	micro, 1e-6
n	nano, 1e-9
p	pico, 1e-12
f	femto, 1e-15
a	atto, 1e-18

7.4.5 Name of Values

S[1,1]	S-parameter value
nodename.V	DC voltage at node *nodename*
name.I	DC current through component *name*
nodename.v	AC voltage at node *nodename*
name.i	AC current through component *name*
nodename.vn	AC noise voltage at node *nodename*
name.in	AC noise current through component *name*
nodename.Vt	Transient voltage at node *nodename*
name.It	Transient current through component *name*

Note: All voltages and currents are peak values. Note: Noise voltages are RMS values at 1 Hz bandwidth.

7.5 Constants

i, j	Imaginary unit (“square root of -1”)
pi	4*arctan(1) = 3.14159...
e	Euler = 2.71828...
kB	Boltzmann constant = 1.38065e-23 J/K
q	Elementary charge = 1.6021765e-19 C

CHAPTER 8

List of Special Characters

It is possible to use special characters in the text painting and in the text of the diagram axis labels. This is done by using LaTeX tags. The following table contains a list of currently available characters.

Note: Which of those characters are correctly displayed depends on the font used by Qucs!

Small Greek letters

LaTeX tag	Unicode	Description
\alpha	0x03B1	alpha
\beta	0x03B2	beta
\gamma	0x03B3	gamma
\delta	0x03B4	delta
\epsilon	0x03B5	epsilon
\zeta	0x03B6	zeta
\eta	0x03B7	eta
\theta	0x03B8	theta
\iota	0x03B9	iota
\kappa	0x03BA	kappa
\lambda	0x03BB	lambda
\mu	0x03BC	mu
\textmu	0x00B5	mu
\nu	0x03BD	nu
\xi	0x03BE	xi
\pi	0x03C0	pi
\varpi	0x03D6	pi
\rho	0x03C1	rho
\varrho	0x03F1	rho
\sigma	0x03C3	sigma
\tau	0x03C4	tau
\upsilon	0x03C5	upsilon
\phi	0x03C6	phi
\chi	0x03C7	chi
\psi	0x03C8	psi
\omega	0x03C9	omega

Capital Greek letters

LaTeX tag	Unicode	Description
\Gamma	0x0393	Gamma
\Delta	0x0394	Delta
\Theta	0x0398	Theta
\Lambda	0x039B	Lambda
\Xi	0x039E	Xi
\Pi	0x03A0	Pi
\Sigma	0x03A3	Sigma
\Upsilon	0x03A5	Upsilon
\Phi	0x03A6	Phi
\Psi	0x03A8	Psi
\Omega	0x03A9	Omega

Mathematical symbols

LaTeX tag	Unicode	Description
\cdot	0x00B7	multiplication dot (centered dot)
\times	0x00D7	multiplication cross
\pm	0x00B1	plus minus sign
\mp	0x2213	minus plus sign
\partial	0x2202	partial differentiation symbol
\nabla	0x2207	nabla operator
\infty	0x221E	infinity symbol
\int	0x222B	integral symbol
\approx	0x2248	approximation symbol (waved equal sign)
\neq	0x2260	not equal sign
\in	0x220A	“contained in” symbol
\leq	0x2264	less-equal sign
\geq	0x2265	greater-equal sign
\sim	0x223C	(central european) proportional sign
\propto	0x221D	(american) proportional sign
\diameter	0x00F8	diameter sign (also sign for average)
\onehalf	0x00BD	one half
\onequarter	0x00BC	one quarter
\twosuperior	0x00B2	square (power of two)
\threesuperior	0x00B3	power of three
\ohm	0x03A9	unit for resistance (capital Greek omega)

CHAPTER 9

Matching Circuits

Creating matching circuits is an often needed task in microwave technology. Qucs can do this automatically. These are the neccessary steps:

Perform an S-parameter simulation in order to calculate the reflexion coefficient.

Place a diagram and display the reflexion coefficient (i.e. S[1,1] for port 1, S[2,2] for port 2 etc.)

Set a marker on the graph and step to the desired frequency.

Click with the right mouse button on the marker and select “power matching” in the appearing menu.

A dialog appears where the values can be adjusted, for example the reference impedance can be chosen different from 50 ohms.

After clicking “create” the page switches back to the schematic and by moving the mouse cursor the matching circuit can be placed.

The left-hand side of the matching circuit is the input and the right-hand side must be connected to the circuit.

If the marker points to a variable called “Sopt”, the menu shows the option “noise matching”. Note that the only different to “power matching” is the fact that the conjugate complex reflexion coefficient is taken. So if the variable has another name, noise matching can be chosen by re-adjusting the values in the dialog.

The matching dialog can also be called by menu (Tools->matching circuit) or by short-cut (<CTRL-5>). But then all values has to be entered manually.

9.1 2-Port Matching Circuits

If the variable name in the marker text is an S-parameter, then an option exists for concurrently matching input and output of a two-port circuit. This works quite alike the above-mentioned steps. It results in two L-circuits: The very left node is for connecting port 1, the very right node is for connectiong port 2 and the two node in the middle are for connecting the two-port circuit.

CHAPTER 10

Installed Files

The Qucs system needs several programs. These are installed during the installation process. The path of Qucs is determined during the installation (`configure` script). The following explanations assume the default path (`/usr/local/`).

- `/usr/local/bin/qucs` - the GUI
- `/usr/local/bin/qucsator` - the simulator (console application)
- `/usr/local/bin/qucsedit` - a simple text editor
- `/usr/local/bin/qucshelp` - a small program displaying the help system
- `/usr/local/bin/qucstrans` - a program for calculation transmission line parameters
- `/usr/local/bin/qucsfilter` - a program synthesizing filter circuits
- `/usr/local/bin/qucsconv` - a file format converter (console application)

All programs are stand-alone applications and can be started independently. The main program (GUI)

- calls `qucsator` when performing a simulation,
- calls `qucsedit` when showing text files,
- calls `qucshelp` when showing the help system,
- calls `qucstrans` when calling this program from menu "Tools",
- calls `qucsfilter` when calling this program from menu "Tools",
- calls `qucsconv` when placing the SPICE component and when performing a simulation with the SPICE component.

Furthermore, the following directories are created during installation:

- `/usr/local/share/qucs/bitmaps` - contains all bitmaps (icons etc.)
- `/usr/local/share/qucs/docs` - contains HTML documents for the help system
- `/usr/local/share/qucs/lang` - contains the translation files

10.1 Command line arguments

`qucs [file1 [file2 ...]]`

`qucsator [-b] -i netlist -o dataset` (b = progress bar)

`qucsedit [-r] [file]` (r = read-only)

`qucshelp` (no arguments)

```
qucsconv -if spice -of qucs -i netlist.inp -o netlist.net
```

CHAPTER 11

Schematic File Format

This document describes the schematic file format of Qucs. This format is used for schematics (usually with suffix `.sch`) and for data displays (usually with suffix `.dpl`). The following text shows a short example of a schematic file.

```
<Qucs Schematic 0.0.6>
<Properties>
  <View=0,0,800,800,1,0,0>
</Properties>
<Symbol>
  <.ID -20 14 SUB>
</Symbol>
<Components>
  <R R1 1 180 150 15 -26 0 1 "50 Ohm" 1 "26.85" 0 "european" 0>
  <GND * 1 180 180 0 0 0 0>
</Components>
<Wires>
  <180 100 180 120 "" 0 0 0 "">
  <120 100 180 100 "Input" 170 70 21 "">
</Wires>
<Diagrams>
  <Polar 300 250 200 200 1 #c0c0c0 1 00 1 0 1 1 1 0 5 15 1 0 1 1 315 0 225 "" "" "">
  <"acnoise2:S[2,1]" #0000ff 0 3 0 0 0>
  <Mkr 6e+09 118 -195 3 0 0>
</Polar>
</Diagrams>
<Paintings>
  <Arrow 210 320 50 -100 20 8 #000000 0 1>
</Paintings>
```

The file contains several section. Each of it is explained below. Every line consists of not more than one information block that starts with a less-sign < and ends with a greater-sign >.

11.1 Properties

The first section starts with `<Properties>` and ends with `</Properties>`. It contains the document properties of the file. Each line is optional. The following properties are supported:

- `<View=x1,y1,x2,y2,scale,xpos,ypos>` contains pixel position of the schematic window in the first four numbers, its current scale and the current position of the upper left corner (last two numbers).
- `<Grid=x,y,on>` contains the distance of the grid in pixel (first two numbers) and whether grid is on (last number 1) or off (last number 0).

- <DataSet=name.dat> contains the file name of the data set associated with this schematic.
- <DataDisplay=name.dpl> contains the file name of the data display page associated with this schematic (or the file name of the schematic if this document is a data display).
- <OpenDisplay=yes> contains 1 if the data display page opens automatically after simulation, otherwise contains 0.

11.2 Symbol

This section starts with <Symbol> and ends with </Symbol>. It contains painting elements creating a schematic symbol for the file. This is usually only used for schematic files that meant to be a subcircuit.

11.3 Components

This section starts with <Components> and ends with </Components>. It contains the circuit components of the schematic. The line format is as follows:

```
<type name active x y xtext ytext mirrorX rotate "Value1" visible "Value2" visible ...>
```

- The type identifies the component, e.g. R for a resistor, C for a capacitor.
- The name is the unique component identifier of the schematic, e.g. R1 for the first resistor.
- A 1 in the active field shows that the component is active, i.e it is used in the simulation. A 0 shows it is inactive.
- The next two numbers are the x and y coordinates of the component center.
- The next two numbers are the x and y coordinates of the upper left corner of the component text. They are relative to the component center.
- The next two numbers indicate the mirroring about the x axis (1 for mirrored, 0 for not mirrored) and the counter-clockwise rotation (multiple of 90 degree, i.e. 0...3).
- The next entries are the values of the component properties (in quotation marks) followed by an 1 if the property is visible on the schematic (otherwise 0).

11.4 Wires

This section starts with <Wires> and ends with </Wires>. It contains the wires (electrical connection between circuit components) and their labels and node sets. The line format is as follows:

```
<x1 y1 x2 y2 "label" xlabel ylabel dlabel "node set">
```

- The first four numbers are the coordinates of the wire in pixels: x coordinate of starting point, y coordinate of starting point, x coordinate of end point and y coordinate of end point. All wires must be either horizontal (both x coordinates equal) or vertical (both y coordinates equal).
- The first string in quotation marks is the label name. It is empty if the user has not set a label on this wire.
- The next two numbers are the x and y coordinates of the label or zero if no label exists.
- The next number is the distance between the wire starting point and and the point where the label is set on the wire.

- The last string in quotation marks is the node set of the wire, i.e. the initial voltage at this node used by the simulation engine to find the solution. This is empty if the user has not set a node set for this wire.

11.5 Diagrams

This section starts with `<Diagrams>` and ends with `</Diagrams>`. It contains the diagrams with their graphs and their markers. The line format is as follows (line break not allowed):

```
<x y width height grid gridcolor gridstyle log xAutoscale xmin xstep
 xmax yAutoscale ymin ystep ymax zAutoscale zmin zstep zmax xrotate
 yrotate zrotate "xlabel" "ylabel" "zlabel">
```

- The first two numbers are x and y coordinate of lower left corner.
- The next two numbers are width and height of diagram boundings.
- The fifth number is 1 if grid is on and 0 if grid is off.
- The next is grid color in 24 bit hexadecimal RGB value, e.g. #FF0000 is red.
- The next number determines the style of the grid.
- The next number determines which axes have logarithmical scale.

11.6 Paintings

This section starts with `<Paintings>` and ends with `</Paintings>`. It contains the paintings that are within the schematic.

Technical description concerning the simulator

Available online at http://qucs.sourceforge.net/tech/technical.html

Example schematics

Available online at http://qucs.sourceforge.net/download.html#example