Qucs-S: Help and Reference Documentation Release 0.0.23-S

Mike Brinson and Vadim Kusnetsov (2015 to 2022)

April 12, 2022

Authors Mike Brinson (mbrin72043@yahoo.co.uk) and Vadim Kusnetsov (re3xdh@gmail.com)

Contents

Chapter 1

Background and an overview of Qucs-S

Following the release of Qucs-0.0.18 in August 2014 the Qucs Development Team considered in detail a number of possible directions that future versions of the software could take. Qucs-S is one of these routes. It addresses a number of problems observed while attempting to combine Qucs with the best features of other Free Open Source Software (FOSS) SPICE circuit simulation packages. The Qucs-S project aims to add additional model development tools to those currently available in Qucs while retaining a high degree of compatibility with de facto industrial SPICE simulation. Qucs was originally written as an RF and microwave engineering design tool which provided features not found in SPICE, like for example, S-parameter simulation and RF network synthesis. Since it was first released under the General Public License (GPL) in 2003 Qucs has provided users with a relatively stable, flexible and reasonably functional circuit simulation package, particularly for RF circuit design and analysis. Qucs-S was first released for general use in 2016. The last stable Qt4 version (Qucs-S 0.0.22) being available since January 2020. The primary purpose of Qucs-S is to provide a range of device modelling, circuit design and simulation tools that meet the needs of academia and industry. In February 2022 Qucs-S 0.0.23 was released for general use. This is the first version of Qucs-S ported to Qt5. Originally, it was intended that Qucs-S would at some point merge back with Qucs. However, this is no longer considered likely due to significant divergence between the two software packages. Qucs-S is also developing rapidly with each new release. Today it has become a stable device modelling, simulation and data processing tool with the best features of Qucs and SPICE that are central to a FOSS package of surprising power and utility.

One of the most often requested Qucs-S features is ”better documentation”, especially documentation outlining the use and limitations of the simulation and the modelling features built into Qucs-S. Qucs-S is a large and complex package which is very flexible in the way it can be used as a circuit design aid. Hence, however much documentation is written describing its functionality there are always likely be simulation and modelling examples that are missing from the documentation. In future new Qucs-S releases will be accompanied by a set of documentation. This will introduces a range of simulation and modelling topics and present a number of typical circuit simulation and compact device modelling examples. In the text these are also linked to Qucs-S reference material. The Qucs and Qucs-S Development Teams, and other authors, have published a body of work concerning Qucs/Qucs-S and their applications. A bibliography of these publications can be found at the end of this document. Anyone interested in learning about Qucs-S is recommended to read them as they provide a wealth of information on basic and advanced topics. The Qucs-S documentation is very much work in progress. Moreover, to keep everyone up to date with current developments it is planned to updated them during future Qucs-S development phases.

1.1 Introduction to Qucs-S

Three of the primary aims of the work undertaken by the Qucs-S Development Team are firstly to remove software bugs and improve the overall performance of the package, secondly to address known weaknesses and limitations and thirdly to develop the package by adding features which increase it’s utility. Readers who are not familiar with open source software development may be unaware of how the process works. By ”Qucs-S Development Team” we mean a group of interested individuals who freely give both their time and expertise for the improvement of the Qucs-S package. The Qucs-S Development Team is not a fixed group but rather a dynamic organisation where individuals contribute, simultaneously or at different times, to the same part or separate parts of the software. The Qucs-S group was formed to address the known limitations of the previous Qucs releases and to take advantage of the work done by other GPL circuit simulation teams working on the Ngspice (`http://ngspice.sourceforge.net/`), Xyce (`https://xyce.sandia.gov/`) and SPICE OPUS (`http://www.spiceopus.si/`) simulation engines. The Qucs-S project is an ongoing initiative which attempts to:

- Correct known weaknesses observed with the Qucs analogue simulation engine ”qucsator”. Qucsator is based on classical numerical mathematics routines for the solution of electrical networks, linear and non-linear real and complex equations and time domain algebraic and differential equations. For small circuits, qucsator works well in the DC and AC small signal domains. However it does appear to have propblems in the transient and Harmonic Balance simulation domains, where it can fail to converge to an acceptable solution. Its performance is also often below that expected of a modern circuit simulator employing sparse matrix algorithms.

- Provide Qucs-S users with a choice of SPICE simulation engines selected from Ngspice, Xyce and SPICE OPUS. By selecting Ngspice, Xyce or SPICE OPUS users may capitalise on all the features offered by the extensive SPICE developments which have taken place over the last fifty years. Ngspice, Xyce and SPICE OPUS offer improved transient simulation convergence and speed, particularly for large non-linear circuits. Xyce brings a stable implementation of single and multi-tone Harmonic Balance simulation to Qucs-S with much improved convergence properties for both linear and non linear components and devices. The latest version of Xyce, 7.4 at the time of writing, also offers, for example, S-parameter and transient domain sensitivity analysis. In contrast, SPICE OPUS adds the transient shooting method for steady state analysis of time domain signals and extensive optimization methods.

- Extend Qucs subcircuit, Equation-Defined Device (EDD), Radio Frequency Equation-Defined Device (RFEDD) and Verilog-A device modelling capabilities. The latest Qucs-S offers much improved component and device modelling features that work as interlinked tools, supporting model development as a continuous flow from physical concept to compiled C/C++ code. This feature is centred around a ”turn-key” version of the XSPICE Code Model construction tools. It is also possible to synthesis Ngspice, Xyce and SPICE OPUS netlists from Qucs-S schematics and to synthesise Verilog-A models from Qucs EDD and SPICE B components. Further improvements to the Verilog-A module development tool chain are planned.

- Offer Qucs-S users access to the additional simulation tools and extra component and device models provided by Ngspice, Xyce and SPICE OPUS. This includes much improved component library facilities which allow the use of device manufacturers SPICE models and XSPICE Code Models

- Offer with Qucs-S a true mixed-mode analogue-digital circuit simulation capability using Qucs-S/Ngspice/SPICE OPUS/XSPICE simulation.

.

The Qucs-S project is on going and must be considered as very much work in progress. In its early releases not all the features introduced in the text will be available for public use. It is however, the intention of the Qucs-S Development Team to introduce them as quickly as possible. Other features not listed may also be introduced.

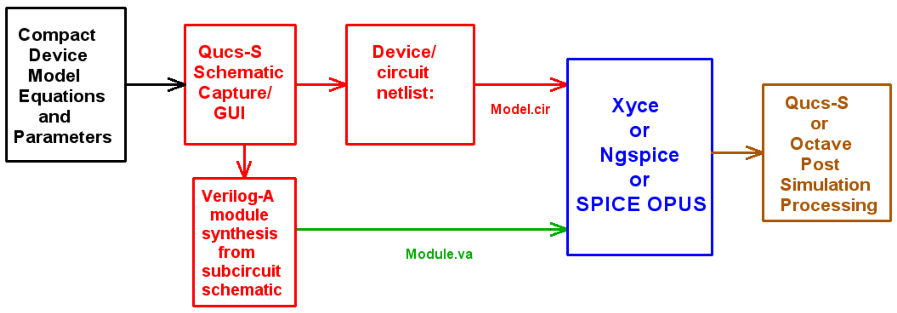

Compact Device Model Equations and Parameters
Qucs-S Schematic Capture/ GUI
Device/ circuit netlist:
Model.cir
Xyce or Ngspice or SPICE OPUS
Qucs-S or Octave Post Simulation Processing
Verilog-A module synthesis from subcircuit schematic
Module.va

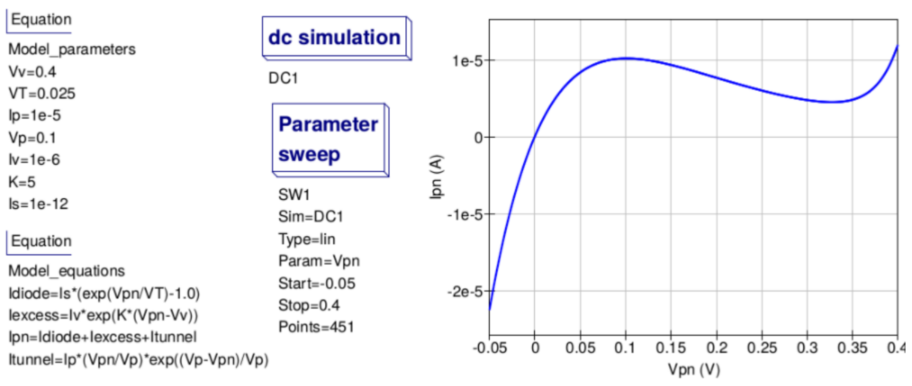
Figure 1.1: A block diagram outlining the Qucs-S simulation and modelling tools.

1.2 Qucs-S structure

A block diagram showing the main analogue modelling and simulation functions of the Qucs-S package are illustrated in Figure 1.1. For convenience, particularly easy identification, blocks with similar modelling or simulation functions have been coded with identical colours, for example dark red indicates the Qucs-S GUI and dark blue the major simulation engines. The direction of data flow between blocks is also indicated with arrows. Central to the operation of Qucs-S is a graphical user interface (GUI), the Ngspice, Xyce and SPICE OPUS simulation engines and a post simulation data processing feature (brown block) for the extraction of device and circuit parameters and the visualisation of simulated signal waveforms. The brown block in Figure 1.1 includes the well known Octave numerical analysis package (https://www.gnu.org/software/octave/). Qucs-S uses Octave for advanced post simulation data processing related tasks.

1.3 A first look at the extended Qucs-S device modelling and simulation features

At this point it seems appropriate to introduce a short example which demonstrates how much Qucs-S has evolved since the release of Qucs version 0.0.18 when Qucs-S development began. This example has been deliberately chosen to present an overview of a number of new features either already developed by the Qucs-S project or planned for future releases. To provide readers with adequate information on how to make the best use of the new features they are described in detail in later chapters of this document.



Equation
Model_parameters
Vv=0.4
VT=0.025
Ip=1e-5
Vp=0.1
Iv=1e-6
K=5
Is=1e-12

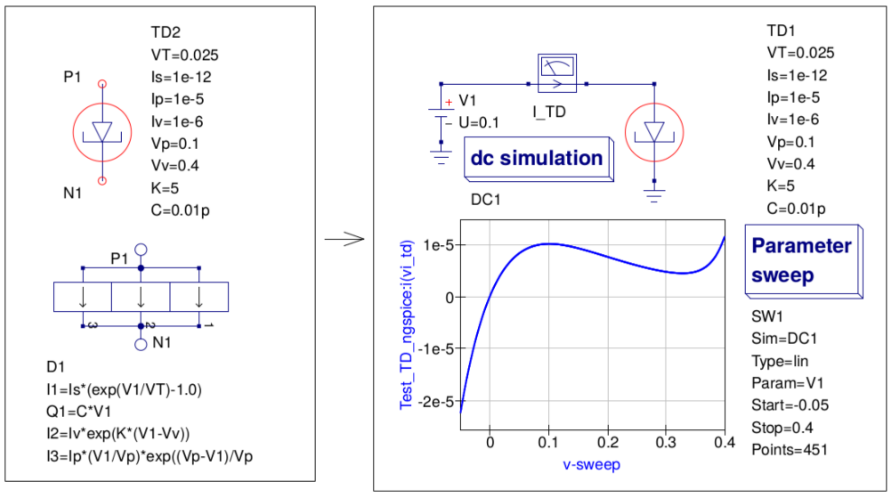
Equation
Model_equations
Idiode=Is*(exp(Vpn/VT)-1.0)
Iexcess=Iv*exp(K*(Vpn-Vv))
Ipn=Idiode+Iexcess+Itunnel
Itunnel=Ip*(Vpn/Vp)*exp((Vp-Vpn)/Vp)

dc simulation
DC1

Parameter sweep
SW1
Sim=DC1
Type=lin
Param=Vpn
Start=-0.05
Stop=0.4
Points=451

Figure 1.2: Equation representation of a semiconductor tunnel diode with $Id = f(Vd)$, including device model parameters, model equations and DC scan test.

Qucs-S is a surprisingly sophisticated program with quite a number of hidden features which are not obvious to most users who are not familiar with the software. Given in Figure 1.2 is a circuuit schematic which demonstrates a little known application of the Qucs family of circuit simulators. Qucs-S is ideal for developing high level behavioural models of new devices or circuits which are not distributed with the release software. The schematic in Figure 1.2 introduces



TD2
VT=0.025
Is=1e-12
Ip=1e-5
Iv=1e-6
Vp=0.1
Vv=0.4
K=5
C=0.01p

D1
I1=Is*(exp(V1/VT)-1.0)
Q1=C*V1
I2=Iv*exp(K*(V1-Vv))
I3=Ip*(V1/Vp)*exp((Vp-V1)/Vp

TD1
VT=0.025
Is=1e-12
Ip=1e-5
Iv=1e-6
Vp=0.1
Vv=0.4
K=5
C=0.01p

V1
U=0.1

dc simulation
DC1

Parameter sweep
SW1
Sim=DC1
Type=lin
Param=V1
Start=-0.05
Stop=0.4
Points=451

Figure 1.3: Qucs-S EDD behavioural model for the tunnel diode.

the physical equations and device parameters for a semiconductor tunnel diode. By using the Qucs-S parameter sweep and DC simulation operations it is possible to scan the diode bias voltage Vpn, calculate the tunnel diode bias current Ipn at each bias point and plot the device $id = f(vd)$ characteristics. Note that in this introductory example the Qucs schematic does not include any electrical components. Moreover, the tunnel diode current is calculated directly from its physical **model equations** and **model parameters**. Illustrated in Figure 1.3 is a second model for the tunnel diode plus a test circuit for simulating the device DC current versus voltage characteristics. It shows how a

Qucs-S Equation Defined Device (EDD) represents the physical model of the tunnel diode and how this model can be represented with it's own symbol and tested by combining it with other components to form a DC characteristic test circuit. The Qucs EDD is not implemented in SPICE 2 and 3 simulators. However, SPICE 3 and later simulators have other similar features, including the B independent voltage and current sources.

```
// Tunnel diode Verilog-A compact device model.
// Verilog-A code translated, by hand, from
// MAPP model.
//
`include "disciplines.vams"
`include "constants.vams"
//
module TD(p,n);
inout p,n;
electrical p,n;
//
parameter Is=1e-12 from [1e-20 : inf);
parameter Ip=1e-5  from [1e-20 : inf);
parameter Iv=1e-6  from [1e-20 : inf);
parameter Vp=0.1   from [1e-20 : inf);
parameter Vv=0.4   from [1e-20 : inf);
parameter KTD = 5    from [1e-20 : inf);
parameter VT=0.025 from [1e-20 : inf);
parameter C=1e-15  from [1e-20 : inf);
//
real Idiode, Iexcess, Itunnel;
//
analog begin
 Idiode = Is*(exp(V(p,n)/VT)-1.0);
 Iexcess = Iv*exp(KTD*(V(p,n)-Vv));
 Itunnel = Ip*(V(p,n)/Vp)*exp(Vp-V(p,n)/Vp);
 I(p,n) <+ Idiode+Iexcess+Itunnel;
 I(p,n) <+ ddt(C*V(p,n));
end
endmodule
```

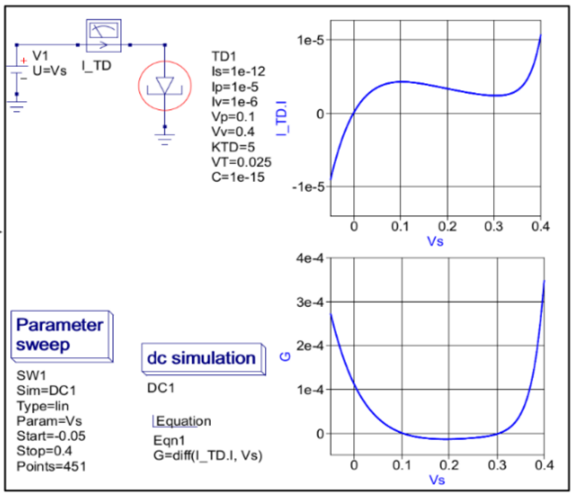


Figure 1.4: A Verilog-A compact tunnel diode model and test circuit.

The Qucs-S EDD component has one feature which makes it particularly important for developing compact device simulation models, namely that its structure and modelling capabilities are similar to those available with the Verilog-A hardware description language. Hence, once a Qucs-S EDD model is operating satisfactorily it can be transcribed into a Verilog-A compact model by inspection or by computer synthesis. Such a Verilog-A model and test circuit is shown in Figure 1.4. One of the main aims of the Qucs-S initiative is to improve Qucs compact device modelling capabilities and to streamline the flow of information between each part of the modelling and simulation sequence. In all Qucs releases prior to the Qucs-S project a number of modelling tools were implemented in the distribution software but users had to translate manually each type of model format to other formats if they wished to use a model with a different simulator or modelling tool. One exception was the rudimentary translation tool called **qucsconv** for translating SPICE netlists to Qucs netlist format. It is not possible to simulate Qucs models encoded in the Qucs netlist format with a SPICE simulator or to generate a Verilog-A code model directly from a Qucs EDD model. This situation will change significantly as the Qucs-S project moves forward: in the medium to long term a number of synthesis-translation routines will be added to Qucs-S making the process of model translation transparent to the Qucs-S user. The first of these is the link between the Qucs netlist format and the Ngspice, Xyce and SPICE OPUS simulator netlist formats. Figure 1.5 lists an Ngspice simulation netlist generated by Qucs-S. Notice that this netlist

```
* Qucs 0.0.19
* Qucs 0.0.19  TD.sch
.SUBCKT TD _net0 _net1 VT=0.025 Is=1e-12 Ip=1e-5 Iv=1e-6 Vp=0.1 Vv=0.4 K=5 C=0.01p
BD1I0 _net0 _net1 I=Is*(exp((V(_net0)-V(_net1))/VT)-1.0)
GD1Q0 _net0 _net1 nD1Q0 _net1 1.0
LD1Q0 nD1Q0 _net1 1.0
BD1Q0 nD1Q0 _net1 I=-(C*(V(_net0)-V(_net1)))
BD1I1 _net0 _net1 I=Iv*exp(K*((V(_net0)-V(_net1))-Vv))
BD1I2 _net0 _net1 I=Ip*((V(_net0)-V(_net1))/Vp)*exp((Vp-(V(_net0)-V(_net1)))/Vp)
.ENDS
XTD2  _net0 0 TD VT=0.025 Is=1E-12 Ip=1E-5 Iv=1E-6 Vp=0.1 Vv=0.4 K=5 C=0.01P
VI_TD1 _net1 _net0 DC 0 AC 0
V1 _net1 0 DC 0.1
.control
set filetype=ascii
DC V1 -0.05 0.4 0.000997783
write _dc.txt VI_TD1#branch
destroy all
exit
.endc
.END
```

Figure 1.5: A synthesized Ngspice netlist for the tunnel diode circuit.

is not simply a list of SPICE component statements but includes an embedded Ngnutmeg script between the SPICE **.control .... and .... .endc** statements. Figures 1.6 and 1.7, introduce a user defined XSPICE Code Model for the tunnel diode example. A new extension to the Qucs-S compact device modelling capabilities adds a ”turn-key” feature which allows user defined XSPICE Code Models to be added to Qucs-S and automatically compiled to C code by the package.

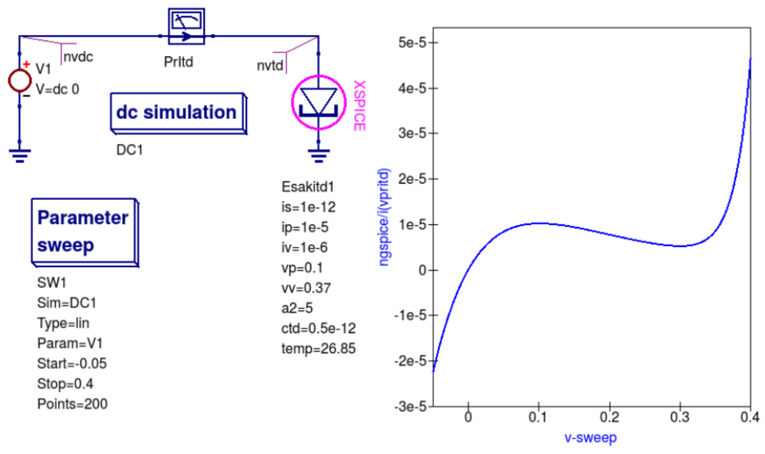
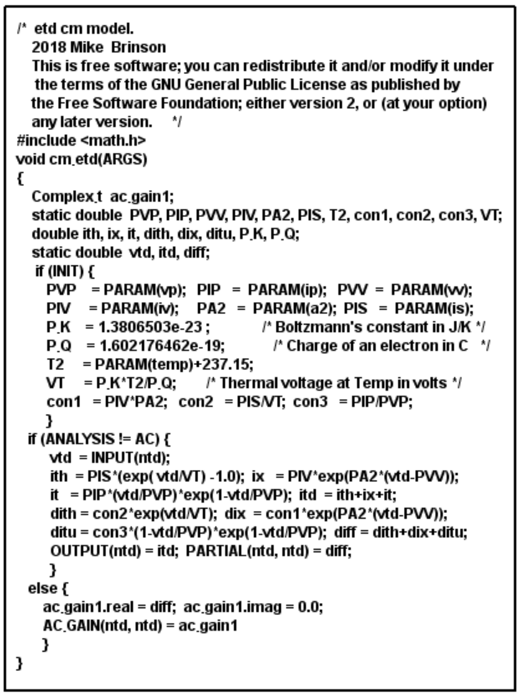


Figure 1.6: XSPICE tunnel diode test circuit.

1.4 Qucs-S usage

The previous section presented a very brief outline of a number of the modelling and simulation features provided by Qucs-S. For many readers device modelling will not be their primary reason for using Qucs-S. Qucs-S combines the best of the original Qucs schematic capture features with the power of SPICE circuit simulation, making the software a good choice as a tool for both printed circuit and integrated circuit design, particularly for academic and industrial use. New simulation features, like S-parameter and Harmonic Balance analysis plus transient sensitivity add capabilities that take Qucs-S way beyond those implemented in the original SPICE 2 and 3 circuit simulators. More on these topics and all the others introduced earlier can be found in later sections of this document.



```
/*  etd cm model.
   2018 Mike  Brinson
   This is free software; you can redistribute it and/or modify it under
    the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2, or (at your option)
   any later version.     */
#include <math.h>
void cm_etd(ARGS)
{
   Complex_t  ac_gain1;
   static double  PVP, PIP, PVV, PIV, PA2, PIS, T2, con1, con2, con3, VT;
   double ith, ix, it, dith, dix, ditu, P_K, P_Q;
   static double  vtd, itd, diff;
    if (INIT) {
      PVP    = PARAM(vp);   PIP   =  PARAM(ip);   PVV  =  PARAM(vv);
      PIV    = PARAM(iv);    PA2   =  PARAM(a2);  PIS   =  PARAM(is);
      P_K    = 1.3806503e-23 ;            /* Boltzmann's constant in J/K */
      P_Q    = 1.602176462e-19;            /* Charge of an electron in C   */
      T2     = PARAM(temp)+237.15;
      VT     = P_K*T2/P_Q;        /* Thermal voltage at Temp in volts */
      con1   = PIV*PA2;   con2   = PIS/VT;  con3   = PIP/PVP;
      }
  if (ANALYSIS != AC) {
      vtd  = INPUT(ntd);
      ith  = PIS*(exp( vtd/VT) -1.0);  ix   = PIV*exp(PA2*(vtd-PVV));
      it   = PIP*(vtd/PVP)*exp(1-vtd/PVP);  itd  = ith+ix+it;
      dith = con2*exp(vtd/VT);  dix  = con1*exp(PA2*(vtd-PVV));
      ditu = con3*(1-vtd/PVP)*exp(1-vtd/PVP);  diff = dith+dix+ditu;
      OUTPUT(ntd) = itd;  PARTIAL(ntd, ntd) = diff;
      }
  else {
     ac_gain1.real = diff;  ac_gain1.imag = 0.0;
     AC_GAIN(ntd, ntd) = ac_gain1
     }
}
```

Figure 1.7: XSPICE tunnel diode model code.

Chapter 2

Basic Ngspice, Xyce and SPICE OPUS simulation

2.1 Introduction

This section describes a number of fundamental methods for launching circuit simulations from the Qucs-S GUI using the Ngspice, Xyce and SPICE OPUS compatible simulator engines. Qucs-S includes built-in support for SPICE via a subsystem specifically designed for this purpose. The Ngspice, Xyce and SPICE OPUS simulators are not embedded in Qucs-S but operate as independent external simulators. Before use they must be installed on the computer operating system that you are running Qucs-S. Although Ngspice, Xyce and SPICE OPUS are all compatible SPICE simulators they also include extensions to the original SPICE 3f5 netlist syntax which are often incompatible and may not simulate on each of the three external simulators. The Qucs-S Development Team are aware of this limitation and are attempting to identify and correct such problems as quickly as possible. Please note this may take some time. However, if you do identify a compatibility bug, or indeed any bug, please inform us by sending in a bug report to the Qucs-S web site (with an example test schematic if possible) describing the problem you have identified.

2.2 Supported simulators

2.2.1 Ngspice

Ngspice is a mixed-level/mixed-signal circuit simulator implemented from three open source software packages: SPICE 3f5, Cider 1b1 and XSPICE. Ngspice is one of the most widely used and stable current generation open source SPICE simulators available. It implements the original SPICE3f5 simulation capabilities, including for example, DC, AC, and transient simulation, Fourier-analysis, and DC and AC sensitivity analysis, plus a significant number of extra simulation and device modelling extensions, including S-parameter network analysis (from release Ngspice-37). Distributed with Ngspice is a data manipulation package called Ngnutmeg. This provides numerical analysis and visualisation routines for post processing Ngspice simulation data. Instructions for installing Ngspice can be found on the Ngspice website at http://ngspice.sourceforge.net/download.html, The Ngspice website also gives free access to all the binary distribution and the development code sources.

2.2.2 Xyce

Xyce is an open source, SPICE-compatible, high-performance parallel analogue circuit simulator that is capable of solving extremely large circuit problems when installed on large-scale parallel computing platforms. It also supports serial execution on all common desktop platforms, and small-scale parallel operation on Unix-like systems. Xyce for Linux and Microsoft Windows can be downloaded from the official Xyce website at https://xyce.sandia.gov/Xyce. The Xyce parallel circuit simulator running on Linux requires installation of the openMPI libraries. Qucs-S supports both Xyce-Serial and Xyce-Parallel software (not currently available for the Microsoft Windows operating system).

2.2.3 SPICE OPUS

SPICE OPUS is an improved version of SPICE based on SPICE 3 code with extensions for circuit and device performance optimization and the transient simulation shooting method for large signal steady state analysis. SPICE OPUS can be downloaded from its official website at http://www.spiceopus.si/.

2.3 General simulation methods

The starting point for understanding how the SPICE extensions are built into the Qucs-S GUI is to study the basic operations needed to simulate circuit schematics with external simulators. For this purpose consider the RCL test circuit shown in Figure 2.1. This schematic specifies two simulations: (1) AC simulation from 4 MHz to 12 MHz

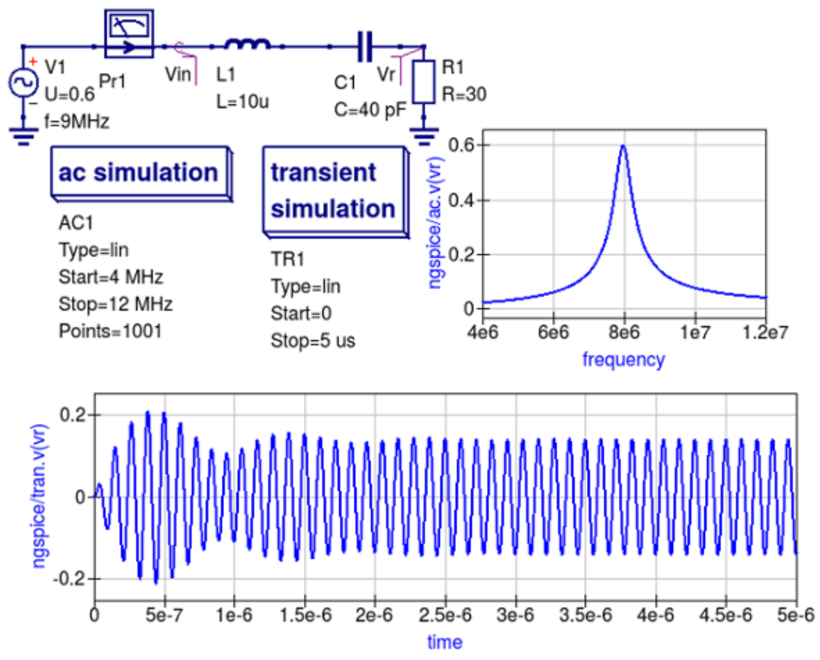
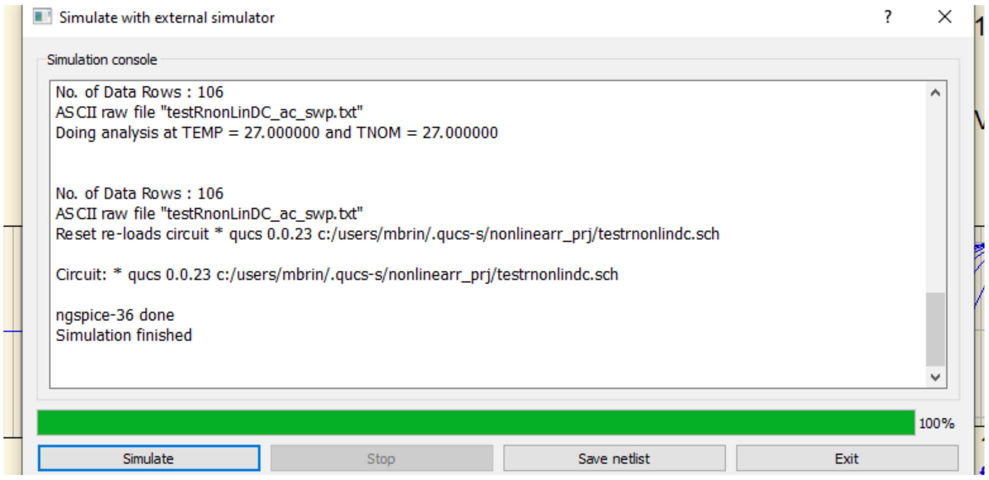


Figure 2.1: An RCL test circuit for demonstrating Ngspice, Xyce and SPICE OPUS simulation controlled from Qucs-S.

and (2) Transient simulation from 0 to 5 us. Draw the schematic drawn in Figure 2.1 with Qucs-S. Make sure the schematic is entered correctly then simulate it with Ngspice using the sequence "Simulation − > Simulate", or by pressing key F2. After Qucs-S finishes the AC and transient simulations, plot the output data listed below:

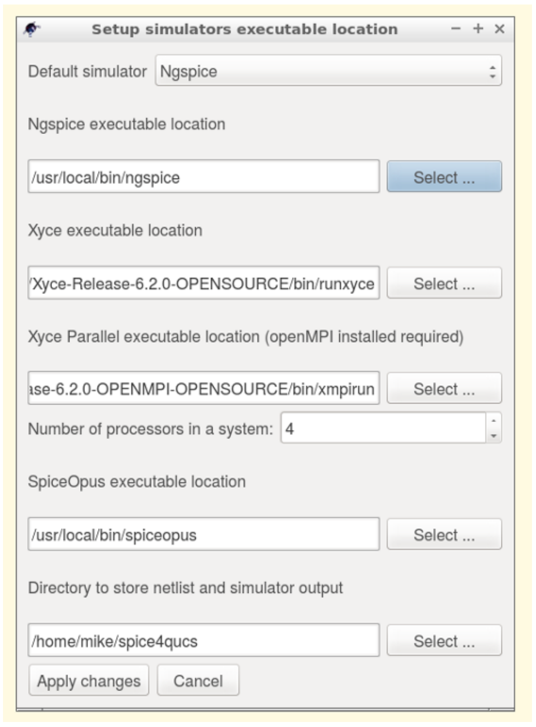
- The frequency domain input and output voltage waveforms (given by vatages at the "Vin" and "Vr" nodes) - your plots should be similar to those shown in Figure 2.1,
- The current in the frequency domain ("Pr1" current probe,)
- The transient current waveform, sensed by the current probe "Pr1".

Qucs-S allows schematic component properties to be defined in the same way as the original Qucs software. Component values and other icon properties are converted automatically into SPICE compatible netlist format. There is no need for manual adaptation by users. However, please note that not all the predefined Qucs components are available for simulation with Ngspice, Xyce or SPICE OPUS. A number of tables provided in later sections of the text list which components can be used with which simulator. Following placement and wiring of components, plus the addition of one or more simulation icons, SPICE simulation is launched using the Qucs-S menu sequence **Simulation** − > **Simulate** or by pressing key **F2**. An **External** − > **simulator** dialogue then appears. This is illustrated in Figure 2.3.



```
No. of Data Rows : 106
ASCII raw file "testRnonLinDC_ac_swp.txt"
Doing analysis at TEMP = 27.000000 and TNOM = 27.000000


No. of Data Rows : 106
ASCII raw file "testRnonLinDC_ac_swp.txt"
Reset re-loads circuit * qucs 0.0.23 c:/users/mbrin/.qucs-s/nonlinearr_prj/testrnonlindc.sch

Circuit: * qucs 0.0.23 c:/users/mbrin/.qucs-s/nonlinearr_prj/testrnonlindc.sch

ngspice-36 done
Simulation finished
```

Figure 2.2: External simulator dialogue: where button **Simulate** launches a circuit simulation, button **Stop** causes a running simulation to finish; button **Save netlist** generates, and stores, the netlist of the circuit being simulated and button **Exit** closes the external simulator dialogue.



Default simulator: Ngspice

Ngspice executable location: /usr/local/bin/ngspice

Xyce executable location: Xyce-Release-6.2.0-OPENSOURCE/bin/runxyce

Xyce Parallel executable location (openMPI installed required): ase-6.2.0-OPENMPI-OPENSOURCE/bin/xmpirun

Number of processors in a system: 4

SpiceOpus executable location: /usr/local/bin/spiceopus

Directory to store netlist and simulator output: /home/mike/spice4qucs

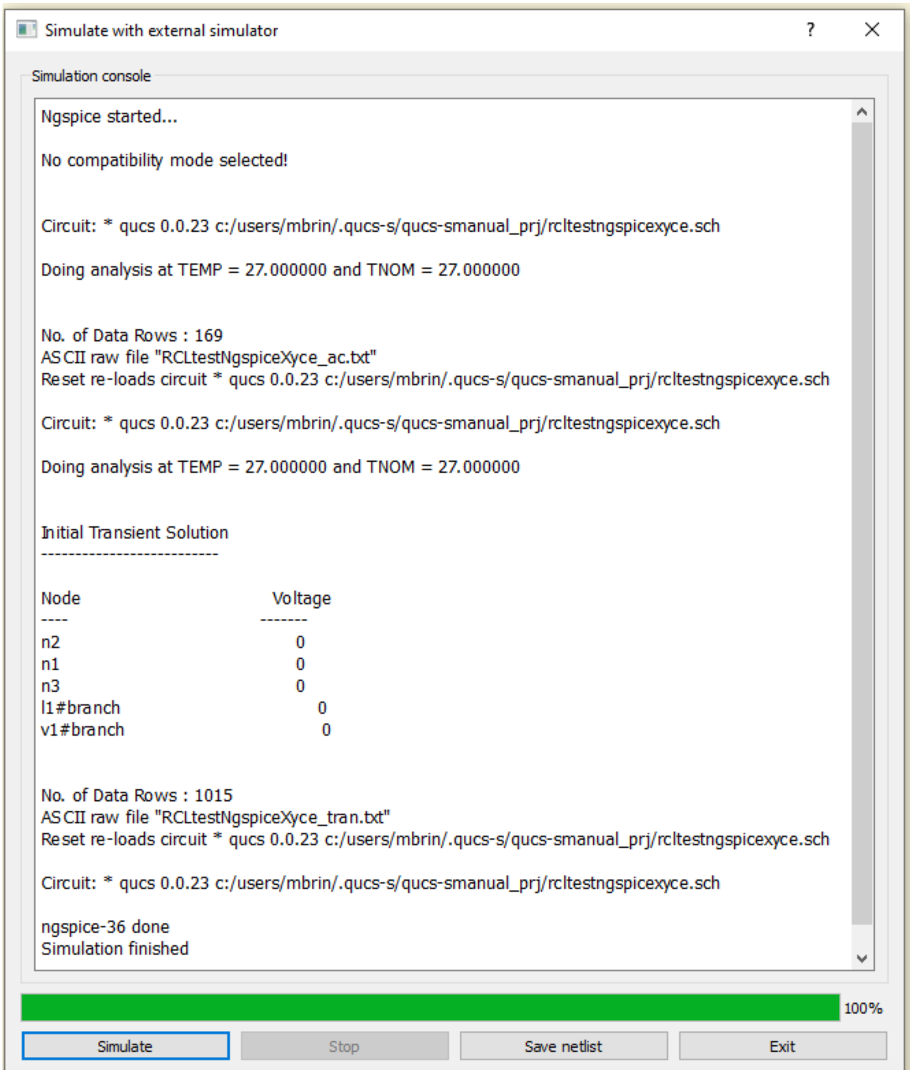
Figure 2.3: **Setup simulator executable locations** dialogue.

If the Ngspice, Xyce or SPICE OPUS installation directories are not included in the operating system shell **$PATH** statement the location of their executable code must be registered with Qucs-S before the Ngspice Xyce or SPICE OPUS simulations will work. This step is necessary for all the operating systems used to run Qucs-S. To register external circuit simulator installation directories Qucs-S users need to launch the **Select default simulator**, from the **Simulate** dialogue. The resulting **Setup simulators executable simulator location** dialogue is illustrated in Figure 2.3.

In this dialogue enter the absolute address of the Ngspice, Xyce or SPICE OPUS executable program code from the keyboard or by pressing the appropriate **Open File Select button**. In the case of the Xyce Parallel simulator the number of processors installed in your computer system must also be entered from the keyboard or selected using the dialogue up-down arrow controls. Please note the Xyce parallel command line for binary Xyce-Parallel package is `<Path_to_xyce_executable>/xmpirun` -np %p, where Qucs-S substitutes the number of processors for the %p wildcard entry. Also please note that "user builds" of Xyce-Parallel have no **xmpirun** script, implying that the full script must be completed by users during the external simulators set up process, for example if **opeMPI** is installed in directory `/usr/bin` and Xyce-Parallel installed in `/usr/local/Xyce_Parallel` the command line is:

`/usr/bin/mpirun` -np %p `/usr/local/Xyce_Parallel/bin/Xyce`.

Qucs-S users can also define a directory where temporary simulator data and netlists are stored: this working directory is by default assumed to be at `$HOME/.qucs/spice4qucs`. To simulate a Qucs schematic with the Ngspice simulator,

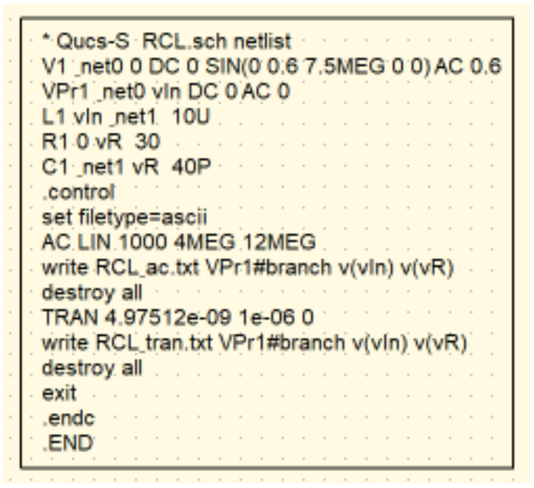


```
Ngspice started...

No compatibility mode selected!


Circuit: * qucs 0.0.23 c:/users/mbrin/.qucs-s/qucs-smanual_prj/rcltestngspicexyce.sch

Doing analysis at TEMP = 27.000000 and TNOM = 27.000000


No. of Data Rows : 169
ASCII raw file "RCLtestNgspiceXyce_ac.txt"
Reset re-loads circuit * qucs 0.0.23 c:/users/mbrin/.qucs-s/qucs-smanual_prj/rcltestngspicexyce.sch

Circuit: * qucs 0.0.23 c:/users/mbrin/.qucs-s/qucs-smanual_prj/rcltestngspicexyce.sch

Doing analysis at TEMP = 27.000000 and TNOM = 27.000000


Initial Transient Solution
--------------------------

Node                    Voltage
----                    -------
n2                            0
n1                            0
n3                            0
l1#branch                     0
v1#branch                     0


No. of Data Rows : 1015
ASCII raw file "RCLtestNgspiceXyce_tran.txt"
Reset re-loads circuit * qucs 0.0.23 c:/users/mbrin/.qucs-s/qucs-smanual_prj/rcltestngspicexyce.sch

Circuit: * qucs 0.0.23 c:/users/mbrin/.qucs-s/qucs-smanual_prj/rcltestngspicexyce.sch

ngspice-36 done
Simulation finished
```

Figure 2.4: A section of an Ngspice execution Log displayed in the **Simulate with an external simulator** dialogue window.

select simulator **Ngspice** and press the **Simulate** button shown in Figure 2.2. During simulation Ngspice produces a simulation log. This is displayed in the **External simulator** dialog window, see Figure 2.4. The Log text is also saved at Qucs-S system Log location `$HOME/.qucs/log.txt`. This can be viewed using the drop down menue sequence **Simulation** – > **Show last messages** (or by pressing key **F5**). If the Ngspice simulation fails, any errors reported during simulation are listed in simulation Log window. Similarly, a successful completion of a Qucs-S/Ngspice simulation is reported.

A novel feature introduced by Qucs-S is its ability to generate and save SPICE netlist files from the information contained in a Qucs schematic. To save the SPICE netlist file for the current simulation press the **Save netlist** button shown in Figure 2.2. This process causes a SPICE netlist to be saved as file **netlist.cir** in the `~/.qucs/spice4qucs` directory. The generated netlist for the RCL test example is listed in Figure 2.5.
The simulation sequence introduced in the previous sections of the Qucs-S -help text also applies to the Xyce and SPICE OPUS simulators. However, the information displayed in the simulation log is likely to be different for each simulator and indeed operating systems. After an Ngspice, Xyce or SPICE OPUS simulation has successfully completed close the **External simulation** dialogue by pressing the **Exit** button. The simulation data generated by a Qucs-S



```
* Qucs-S RCL.sch netlist
V1 _net0 0 DC 0 SIN(0 0.6 7.5MEG 0 0) AC 0.6
VPr1 _net0 vIn DC 0 AC 0
L1 vIn _net1  10U
R1 0 vR  30
C1 _net1 vR  40P
.control
set filetype=ascii
AC LIN 1000 4MEG 12MEG
write RCL_ac.txt VPr1#branch v(vIn) v(vR)
destroy all
TRAN 4.97512e-09 1e-06 0
write RCL_tran.txt VPr1#branch v(vIn) v(vR)
destroy all
exit
.endc
.END
```

Figure 2.5: RCL Ngspice netlist.

simulation is available for plotting using the existing Qucs visualisation routines: either drag a diagram icon, or table icon, onto the current Qucs-S schematic window or onto the associated Qucs-S display page. After a diagram or table is placed a **Diagram properties** dialog appears. On selecting the dataset for the current simulation the output quantities become available for plotting or tabulating in a similar fashion to the original Qucs software. Ngspice, Xyce and SPICE OPUS simulation data output is in raw-binary SPICE 3f5 output format. Results from different types of simulation, for example SPICE AC and TRAN, are combined into a single Qucs-S dataset. Qucs-S adds an appropriate suffix to each simulator dataset name in order to avoid name clashes and mixing up results from different types of simulation. In the RCL test example the Qucs-S schematic is named **RCL.sch**, yielding Ngspice, Xyce and SPICE OPUS simulations result in three different datasets:

- **RCL.dat.ngspice** — for Ngspice
- **RCL.dat.xyce** — for Xyce
- **RCL.dat.spopus** — for SPICE OPUS

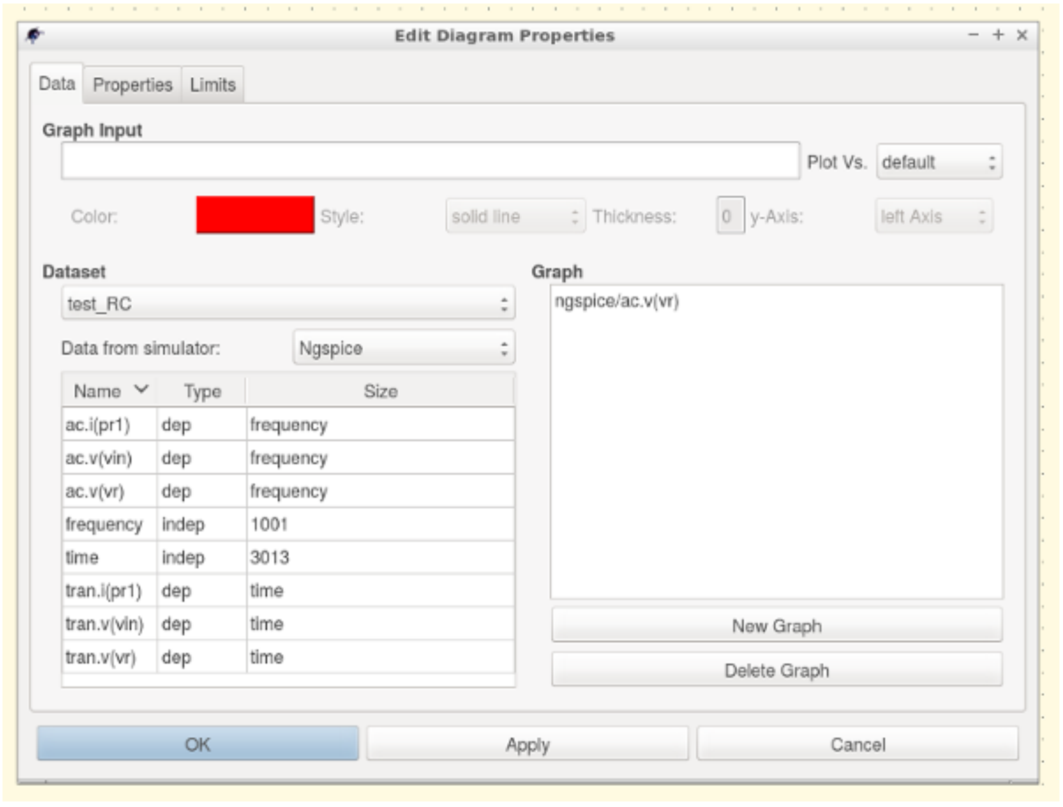
All three datasets have an extension **dat** to signify that each set contains Qucs-S data for post simulation visualisation. The Ngspice, Xyce and SPICE OPUS datasets also include a second extension to the file name to identify the name of the external Qucs-S simulator. The Dataset selector (see Figure 2.6) shows only the base names of a dataset. Users must also select the appropriate simulator from the **simulator name selecto**r drop-down list. This only gives existing simulator datasets which prevents users from selecting non-existent datasets by mistake. Following the selection of a specific data set users must select the variables that are to be plotted. Qucs-S preserves SPICE notation for **node voltage** names and **current probe** names. SPICE names are assumed to be case insensitive by Qucs-S, for example

- **v(out)** — Voltage at node **out**
- **i(Pr1)** — Current recorded by probe **Pr1**

The Qucs-S extension also adds a simulation-dependent prefix to each variable name in order to differentiate output variables from different SPICE simulations, for example **ac.** for AC simulation, **tran.** for transient simulation, and **dc.** or DC-sweep. There are also individual prefixes for each simulator:

- **ngspice/** —– Ngspice simulator prefix;
- **xyce/** —– Xyce simulator prefix;
- **spopus/** —– SPICE OPUS prefix;

Hence for example, the full name of variable from an Ngspice simulation could be **ngspice/v(out)**. This naming system helps to avoid dataset name conflicts. Individual items for plotting are selected by double clicking on a name in the variable list. As an example when double clicking on **ac.i(pr1)** its name is copied by Qucs into the right-hand plotting window. Like the original Qucs one or more variable items may be selected for plotting on the same 2D or 3D graph. Finally pressing the **Apply** button shown at the bottom of Figure 2.5. causes the selected variable



Plot Vs.: default
Style: solid line
Thickness: 0
y-Axis: left Axis

Dataset: test_RC

Data from simulator: Ngspice

Name	Type	Size
ac.i(pr1)	dep	frequency
ac.v(vin)	dep	frequency
ac.v(vr)	dep	frequency
frequency	indep	1001
time	indep	3013
tran.i(pr1)	dep	time
tran.v(vin)	dep	time
tran.v(vr)	dep	time

Graph: ngspice/ac.v(vr)

Figure 2.6: **Diagram properties** dialogue, listing the selected simulator and the available simulation data names.

items to be plotted. The plotted simulation results for the external Ngspice AC simulation of the RCL test circuit are shown in Figure 2.7. Plotting the transient simulation data for the RCL test example follows the same procedure as the sequence described for the AC simulation except that in the transient plot variables with **tran** in their name are selected, see Figure 2.8.

A similar procedure is adopted for plotting simulation data generated with the Xyce and SPICE OPUS simulators. Readers should make sure they can simulate the example RCL circuit with both Xyce and SPICE opus, then plot the resulting simulation data. More advanced techniques for post processing, plotting and undertaking a range of different visualization processes using Qucs-S and Octave are outlined in later chapters of this document.

2.4 Variable names

As part of the Qucs-S extensions Ngspice and Xyce simulation variable names are converted from the original Qucs notation to SPICE notation and vice versa. Table 2.1 shows the correspondence between the two notations. Also variable prefixes are used to designate data from different simulators (Table 2.2)

Table 2.1: Qucs and SPICE variable equivalences

Variable type	Qucs display notation	SPICE display notation
DC node voltage	Node.V	V(node)
AC node voltage	Node.v	ac.v(node)
TRAN node voltage	Node.Vt	tran.v(node)
HB node voltage	Node.Vb	hb.v(node)
DC probe current	Pri.I	i(pri)
AC probe current	Pri.i	ac.i(pri)
TRAN probe current	Pri.It	tran.i(pri)

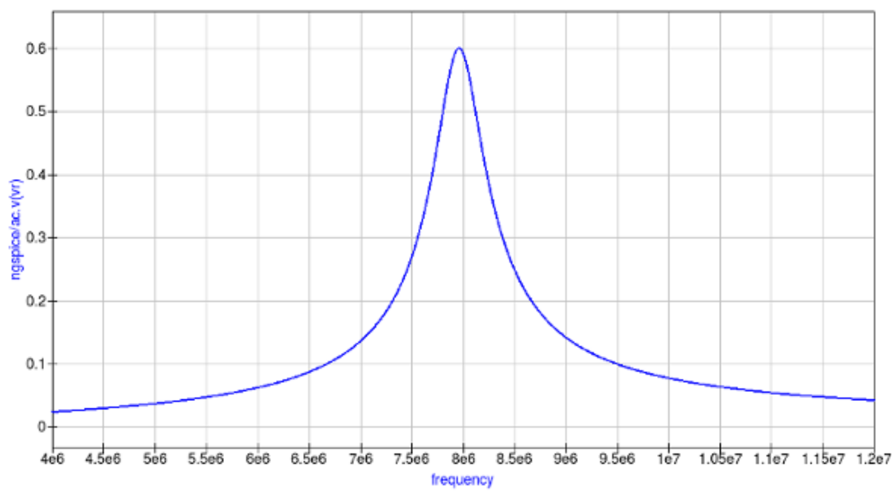


Figure 2.7: External SPICE AC simulation magnitude response for the current flowing in RCL circuit with a series resonant peak of roughly 8 MHz.

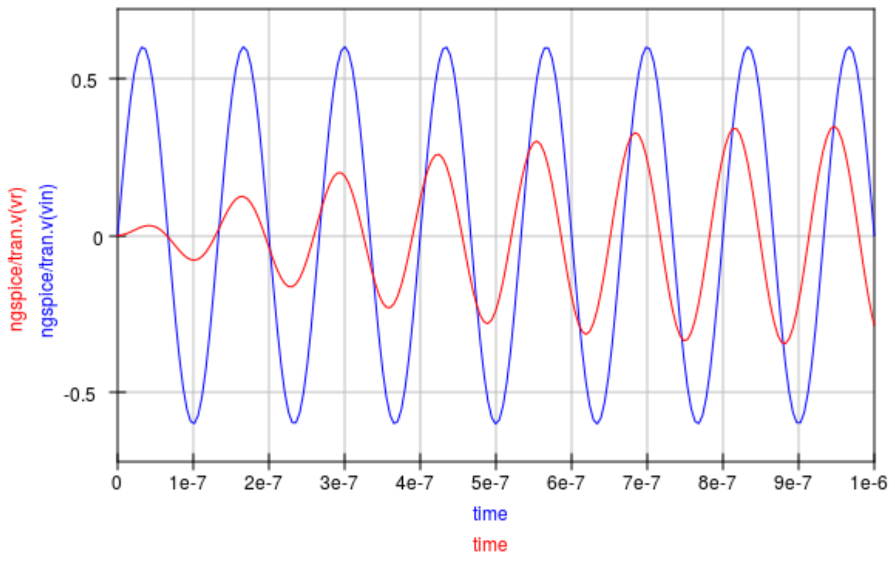


Figure 2.8: Transient simulation voltage waveforms at the input and output nodes of the RCL circuit.

2.5 DC simulation

Conventional SPICE 3f5 simulation commands OP and DC are not implemented by Qucs or indeed by Qucs-S. Instead more convenient versions of these simulation commands are implemented. These alternative forms are linked directly to circuit schematic capture, making them easy to use. Moreover, they provide Qucs-S users with a power full diagnostic and analysis tools for investigating basic circuit operation. The circuit shown in Figure 2.8 represents a simple resistive network with single voltage and current 1 V and 1 A sources respectively. Pressing key "F8" instigates a DC analysis and adds the DC node voltages, probe voltages and probe currents to the current schematic. This feature provides a practical method for scanning a circuit to see if the DC bias values are of the correct order of magnitude. The calculation of DC bias values via the F8 key applies to all the circuit simulators controlled by Qucs-S. Schematics which include the Qucs0S DC icon do not however, list a similar set of voltage and currents in the **Simulate with an external Simulator** dialogue window. In contrast, a DC voltage and current list is output when a schematic includes a transient simulation icon, see Figure 2.9. Qucs does not define a separate analysis type which is equivalent to the original SPICE 2g6 "DC sweep" simulation or the extended SPICE 3f5 version which allows current and voltage source scans plus resistor value scans. In contrast to SPICE the Qucs-S equivalent **DC sweep** is just a specific case of the more general Qucs **Parameter sweep** capability. To emulate the original SPICE DC sweep Qucs0S use a combination of DC simulation plus the parameter sweep of an independent DC voltage or DC

Table 2.2: Qucs and SPICE variable name prefixes

Prefix	**Explanation**
Node.Vt	Qucs simulation, default dataset
dataset:Node.Vt	Qucs simulation, external dataset
ngspice/tran.v(node)	Ngspice simulation, default dataset
xyce/tran.v(node)	Xyce simulation, default dataset
xyce/dataset:tran.v(node)	Xyce simulation, external dataset
spopus/tran.v(node)	SPICE OPUS simulation, default dataset
spopus/dataset:tran.v(node)	SPICE OPUS simulation, external dataset

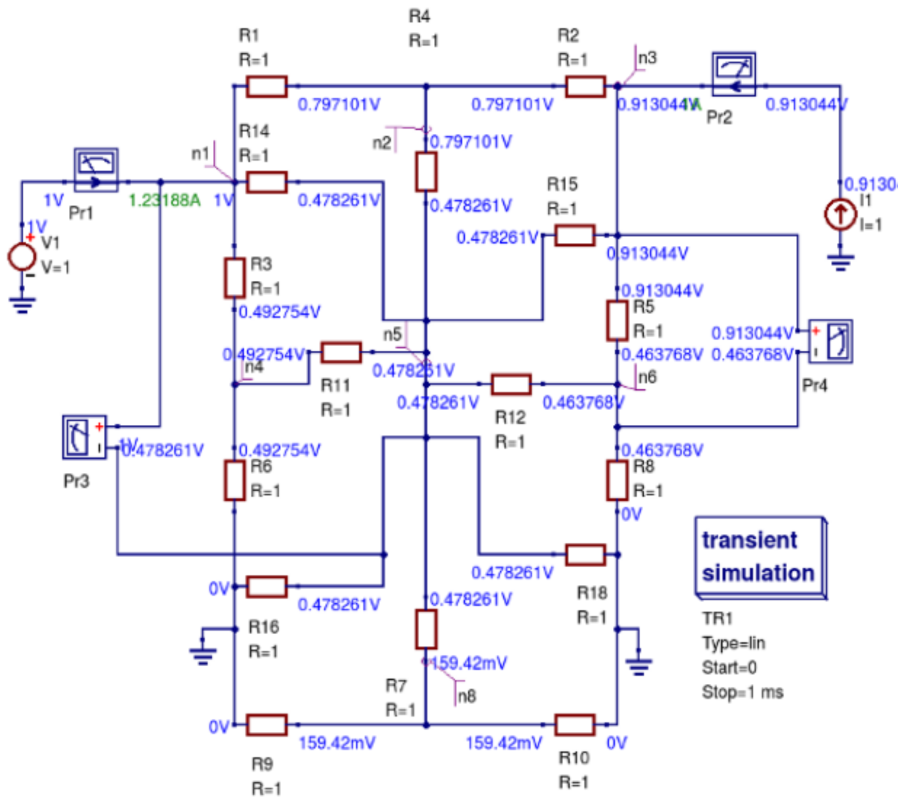


Figure 2.9: A simple linear resistive electrical network driven by single DC voltage and current sources: DC node voltages (V) and voltage probe values (V) are illustrated in blue and current probe values (A) in green.

current source or of a resistor numerical value. When the Qucs-S SPICE netlist builder finds these two linked types of simulation it synthesises them into a **DC** SPICE netlist entry. This procedure is demonstrated in Figure 2.10. where the test circuit consists of a diode DC bias network connected as a test bench for simulating the non-linear DC current-voltage characteristic of a 1N4148 diode. This example can be found in the Qucs examples directory tree listed as `examples\ngspice\diode.sch`.

Please note the following differences between SPICE and Qucs DC-sweep simulation:

- Specify a sweep source name or a resistor name **NOT** a source or resistor value; for example in Figure 2.10 **V1**,
- SPICE model parameters can be swept using the notation **Device.Param**, for example **T1.Bf** to sweep the **Bf** parameter of transistor **T1**.

2.6 AC simulation

Small signal AC simulation is fully supported by Qucs-S. It doesn't require any special adaptation. Just simple place the **AC simulation** component icon on a schematic and execute an Ngspice, Xyce or SPICE OPUS simulation. Variable name conversions are listed in Table 2.1. The Qucs-S dataset builder adds prefix **ac.** to all variables generated

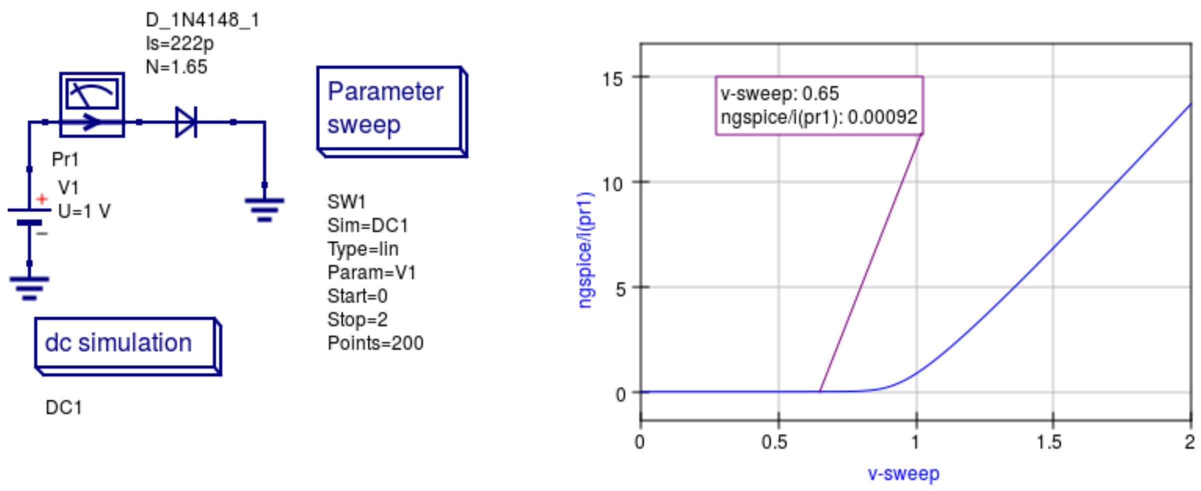


Figure 2.10: Test circuit and simulated DC current-voltage characteristics for a 1N4148 silicon diode.

by an AC simulation. Ngspice, Xyce and SPICE OPUS small signal frequency domain AC simulations use linear, logarithmic. list and constant frequency scales.

2.7 TRANsient simulation

Transient simulation is also fully supported by Qucs-S. Just place the **Transient simulation** component icon on a schematic and simulate it. There is a difference between the way the Ngspice, Xyce and SPICE OPUS simulators implement transient simulation time steps. The original Qucs simulation engine Qucsator used a fixed time step. Ngspice, Xyce and SPICE OPUS use adaptive time steps. The number of simulation points output during a simulation will only be approximately equal to the number of points specified in a **Transient simulation** properties list. For example, in an example test circuit 200 time points are specified on the schematic. However, due to the fact that the SPICE simulators use adaptive time steps, Ngspice employs 213 simulation points, and Xyce employs 799 time points. This difference should be taken into account during simulation data post processing or when comparing results.

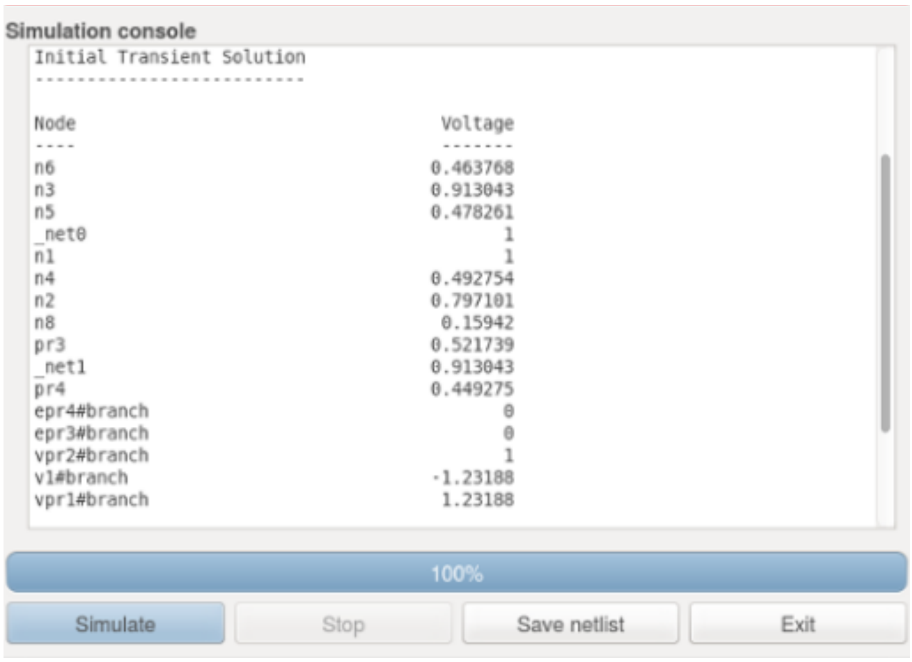
2.8 Other forms of simulation

In contrast to SPICE 3f5, the parameter sweep facility found in Qucs has also been implemented with Ngspice, Xyce and SPICE OPUS where the parameter sweep setup and control is organized by Qucs-S. The details of how this sweep feature works is the topic of a later section. As well as the fundamental DC, AC and transient simulation types, Ngspice, Xyce and SPICE OPUS also support the additional forms of simulation listed in Table 2.3. Fourier, distortion

Table 2.3: Qucs-S simulation types additional to DC, AC and TRAN

Simulation type	**Ngspice**	**Xyce**	**SPICE OPUS**
Fourier	x	x	x
Distortion	x		x
Noise	x	x	x
Pole-zero	x		x
Sensitivity	x	x	x
Harmonic Balance	x	x	
S-parameter analysis	x	x	
TRAN shooting method	x		x
Custom simulation	x	x	

pole-zero circuit simulation require special GUI icons. These can be found in the Qucs **SPICE simulations** group. In contrast sensitivity, the SPICE OPUS tran shooting method are accessed Custom simulation techniques. Again, these topics are introduced in detail in later chapters.


Simulation console

```
Initial Transient Solution
--------------------------

Node                                   Voltage
----                                   -------
n6                                    0.463768
n3                                    0.913043
n5                                    0.478261
_net0                                        1
n1                                           1
n4                                    0.492754
n2                                    0.797101
n8                                     0.15942
pr3                                   0.521739
_net1                                 0.913043
pr4                                   0.449275
epr4#branch                                  0
epr3#branch                                  0
vpr2#branch                                  1
v1#branch                             -1.23188
vpr1#branch                            1.23188
```

Figure 2.11: A screen dump showing transient simulation initial DC simulation voltage and current values in (V) and (A) respectively for the resistive circuit given in Figure 2.8: NOTE that the voltage and current variable names are output in SPICE style syntax.

2.9 Qucs-S circuit simulation components

Qucs is released with a good selection of passive and active component models. This selection includes both fundamental circuit components R, C, L and collections of devices for a given circuit design sector, like the RF microstrip component models. All the original Qucs component and device models were written to work with Qucs and there is **NO Guarantee** that they will be work with Ngspice, Xyce or SPICE OPUS. For circuit simulation packages which take advantage of simulation multi-engines this can be a serious problem, particularly for the less experienced user. To help reduce problems to a minimum, Qucs-S uses a policy of **blacklisting** those models which do not work with the chosen circuit simulation engine. This policy works in the following way:- when a specific simulator is chosen by a Qucs-S user, on running the selected simulator, **ONLY** those models which work with each simulator become available for drawing circuit schematics and simulation. The same approach applies to the components held in the Qucs-S libraries.

2.10 More basic simulation examples

2.10.1 DC Example 1: Calculating circuit input resistance and power dissipation in a resistor

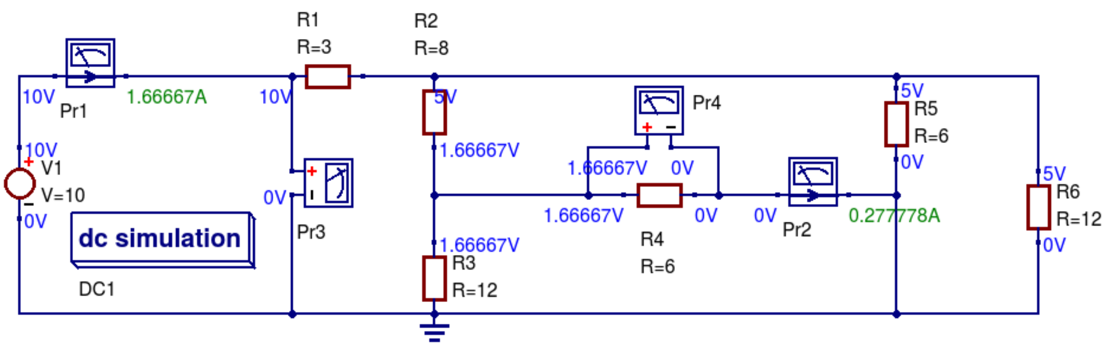


Figure 2.12: DC resistive test network.

- Draw the circuit diagram shown in Figure 2.12,
- Select simulator Ngspice,
- Press key F8 (computes DC bias),
- Determine Rin = V(Pr3)/I(Pr1) = 10/1.66667 = 6 Ohm,
- Calculate the power dissipated in R4 = V(Pr4)*I(Pr2) = 1.66667*0.277778 = 0.463 W.

2.10.2 DC Example 2: Variation of power dissipation with varying DC input voltage

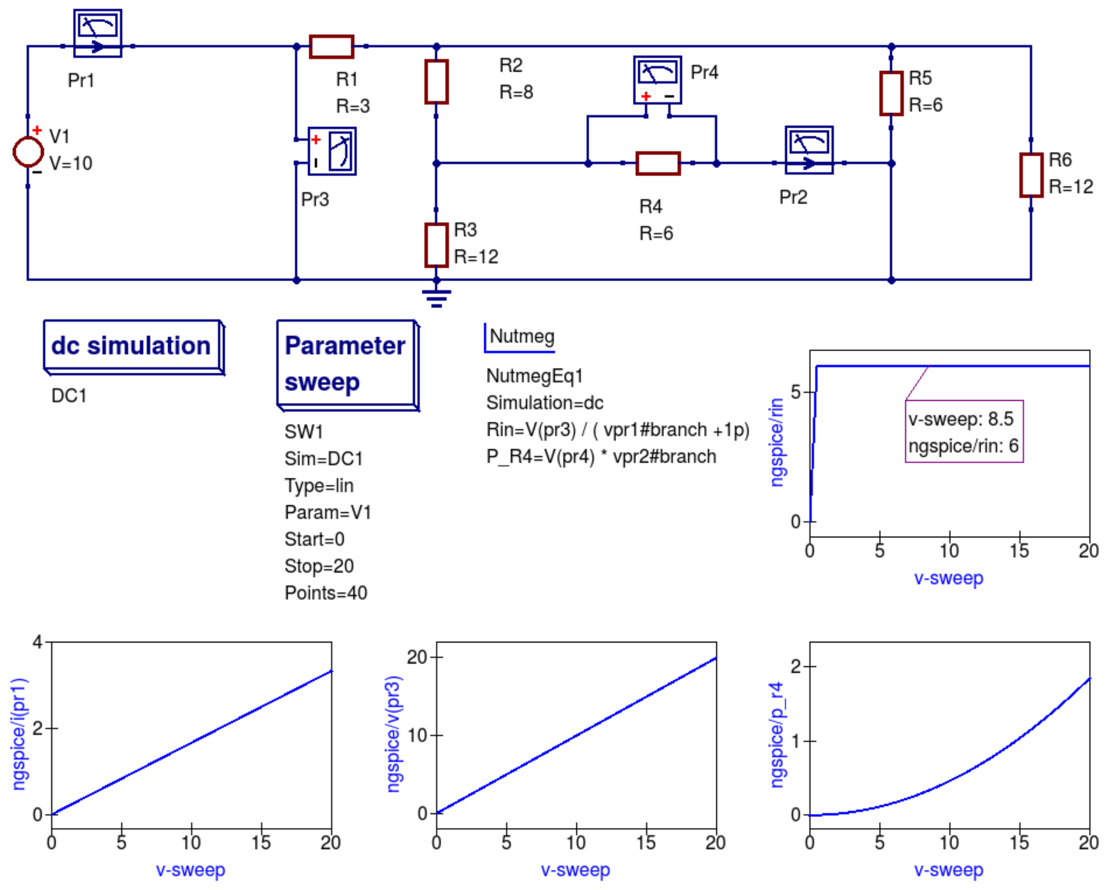


Figure 2.13: DC example 1 with varying input voltage: demonstrating the use of a DC sweep simulation.

- Draw the circuit diagram shown in Figure 2.13,
- Add the dc simulation, Parameter sweep and Nutmeg component icons to the drawn schematic,
- Press the F2 to simulate the circuit,
- Plot the graphs illustrated in Figure 2.13,
- Check that your results are the same - if not or the simulation fails check your schematic for errors and re-simulate.
- `Note 1: Current probe values are represented by vpr1#branch and $vpr2#branch$`
- `Note 2: There is a discontinuity in Rin when the $vpr1#branch$ current is zero; hence the need for the dummy 1pA in the Nutmeg equation for Rin.`