PULSONIX

Pulsonix Spice Simulator

User's Guide

Copyright Notice

Pulsonix, a division of WestDev Ltd.

Printed in the UK.   Issue date: 26/06/15 Iss 9

Pulsonix
20 Miller Court
Severn Drive
Tewkesbury Business Park
Tewkesbury
Glos, GL20 8DN
United Kingdom

Phone	+44 (0)1684 296 551
Fax	+44 (0)1684 296 515
Email	info@pulsonix.com
Support	support@pulsonix.com

Contents

Chapter 1. Introduction

What Is Pulsonix Spice

Pulsonix Spice is a mixed-signal circuit simulator designed for ease and speed of use. It is based on two publicly available programs namely **SPICE** developed by the CAD/IC group at the department of Electrical Engineering and Computer Sciences, University of California and **XSPICE** developed by the Computer Science and Information Technology Laboratory, Georgia Tech Research Institute, Georgia Institute of Technology. **XSPICE** itself was developed from SPICE3C1.

Although originally derived from these programs Pulsonix Spice includes front and back-end features that dramatically enhance their overall capability. Further, many improvements to the underlying simulation algorithms and device models have been made to the original code to increase accuracy and convergence reliability.

Features:

- Closely coupled direct matrix analog and event driven digital simulator.
- Enhanced transient analysis and dc operating point algorithms to virtually eliminate convergence problems.
- Cross Probing of voltage, current and device power from the Schematic.
- Monte Carlo analysis including support for both lot and device tolerances.
- Full featured scripting language allowing customised waveform analysis and automated simulation.
- Incremental graphing of results as simulation proceeds.
- Real time noise source allowing investigation of noise effects with transient analysis.
- Functional modelling with arbitrary non-linear source and arbitrary linear s-domain transfer function.
- Arbitrary logic block for definition of any digital device using descriptive language. Supports synchronous, asynchronous and combinational logic as well as ROMS and RAMS.
- Models for saturable magnetic components including support for air-gaps.
- Built in vertical MOSFET model with non-linear gate-drain capacitance and temperature dependent drain-source resistance.

Why Simulate ?

Integrated circuit designers have been using analog simulation software for two decades. The difficulty of bread-boarding and high production engineering costs have made the use of such software essential.

For discrete designers the case has not been so clear cut. For them prototyping is straightforward, inexpensive and generally provides an accurate assessment of how the final production version of a circuit will behave. By contrast computer simulation has been seen as slow and prone to inaccuracies stemming from imperfect models.

In recent years, however, the simulation of discrete analog circuits has become more viable. This has come about because of the almost relentless advances in CPU power, the increased availability of device models from their manufacturers and the introduction of easy to use and affordable simulation tools such as Pulsonix Spice.

The pressure to reduce product development timescales has meant that for many projects the traditional bread-boarding phase is skipped altogether - with or without simulation - and circuit development is carried out on the first revisions of PCB. The use of simulation on a circuit or parts of a circuit can help to eliminate errors in a circuit design prior to this stage and reduce the number of PCB revisions required before the final production version is reached. Of course, to be useful, the simulation process must therefore not be too time consuming.

Computer simulation, does however, have many more uses. There are some things you can do with a simulator which cannot be achieved with practical approaches. You can remove parasitic elements, you can make non-invasive measurements that are impossible in real-life or you can run components outside of their safe operating area. These abilities make simulation a valuable tool for finding out why a particular design does not behave as expected. If the problem can be reproduced on a simulator then its cause can be much more easily identified. Even if a problem cannot be reproduced then this gives some clues. It means that it is caused by something that is not modelled, a wiring parasitic perhaps.

Simulation is extremely useful for testing ideas at the system level. Sometimes it is not easy to test a concept because the hardware to implement it is very costly or time consuming to build. It may even be that you don't know how to implement the idea in hardware at all. The alternative is to design a model and simulate it with a computer. Once it has been established that the concept is viable then attention can be given to its implementation. If it proves not to be viable, then a great deal of time will have been saved.

What's In The Product ?

Pulsonix Spice has three main elements:

1. The Schematic capture stage. This allows you to graphically enter the circuit using Parts, which have Spice parameters already attached to them. You can also apply stimulus Symbols, Probes and functional symbols to complete the circuit ready for simulation.
   Because of the unique way in which Pulsonix Spice has been developed, the circuit entered is suitable for simulation and can be run directly into Pulsonix PCB design editor with no further interaction of the Parts!
2. The simulator. This is a ‘number cruncher’ that takes a netlist as it's input and generates a - sometimes very large - data file containing the results. The netlist contains a description of your circuit along with commands to instruct the simulator exactly what analysis to perform. The data file contains all the voltages and currents in your circuit for later plotting or analysis.

The simulator is based on SPICE3 developed by the University of California at Berkeley and XSPICE developed by the Georgia Technical Research Institute. Although based on these programs, Pulsonix Spice has received extensive further development to improve its overall performance.

3. The waveform viewer. This part of the program plots the results of the simulation using the data file created by the simulator. It can do this while the simulator is running or on demand after the simulation has completed. As well as plotting facilities, the waveform viewer can also perform measurements on the results such as RMS, peak-peak, frequency etc.

To run Pulsonix Spice you will also require the Pulsonix Schematics package. This is used to hold the Spice model properties, create the circuit and to generate the netlist, which is simulated.

System Requirements

To run Pulsonix Spice your PC hardware will need to conform to the following requirements:

- A Pentium class processor (1.8Ghz or better recommended)
- Microsoft Windows Vista/Windows 7/Windows 8 *
- 1024Mb of RAM or more recommended
- 1.4Gb of hard disk space for the program files and libraries
- SVGA 1024x768 graphics or better
- CD-ROM or compatible drive to install the program
- USB port to attach the security device to
- A mouse which is supported and runs under Windows (a 3 button wheel mouse is also recommended, refer to the following section for Supported Mouse Types)

** It is highly recommended that the most up-to-date Windows operating systems is always used. See www.pulsonix.com for further information about operating system support.*

*Windows 3.1x and Windows 95/98/NT/2000 are not supported.

Installation

Pulsonix Spice is usually installed along with the main Pulsonix product. From the setup program during installation, on the Programs page, you can select Spice and Spice libraries as options to select for installation. If you opted not to install these with the main Pulsonix product, you can do so at any time. Simply insert the Pulsonix CD and wait a short time while the *autorun* feature starts. Follow the instructions.

You will need a Pulsonix Spice feature within the license feature to run this cost option. This will be provided by our sales office or your local Pulsonix representative.

Pulsonix Spice and SPICE

Pulsonix Spice is developed from SPICE version 3e.2 and subsequently upgraded to include 3f.5 enhancements. The simulator core remains substantially the same as SPICE but we have developed it further to improve convergence and to remove bugs and ‘memory leaks’.

However, we have modified some of the devices and not implemented others. We have also not implemented all analysis modes available with SPICE 3. Details follow.

Devices

Capacitors:

Pulsonix Spice:	Allows two temperature coefficients of capacitance and two voltage coefficients on device line and model. (TC1, TC2, VC1, VC2).
SPICE 2	Allows polynomial voltage coefficients. Temperature coefficients not supported.
SPICE 3	Has capacitor model allowing specification in terms of process parameters. Does not support voltage or temperature coefficients.

Inductors:

Pulsonix Spice	Inductance value only
SPICE 2	Allows polynomial current coefficients.
SPICE 3	Inductance value only

MOSFETs:

Pulsonix Spice	Supports levels 1, 2 & 3 and new proprietary level 7 designed for vertical devices. Supports specification for L and W as model parameters for compatibility with PSPICE®.
SPICE 2	Supports levels 1, 2 & 3
SPICE 3	Supports levels 1-6. Levels 4 and 5 are BSIM1 and BSIM2 respectively. Level 6 is a new empirical model

Resistors:

Pulsonix Spice	Supports 2 temperature coefficients on device line and in model.
SPICE 2	Supports 2 temperature coefficients on device line only.
SPICE 3	Supports 2 temperature coefficients in model only. Also allows specification of resistance in terms of process parameters.

Voltage Controlled Switches:

Pulsonix Spice	Switch specified in terms of model describing on resistance (RON), off resistance (ROFF), turn on voltage (VON) and turn off voltage (VOFF). Resistance between VON and VOFF varies continuously following a cubic law. Switch model is compatible with PSPICE®
SPICE 2	Switches not implemented.
SPICE 3	Switch specified in terms of on resistance, off resistance, threshold voltage and hysteresis voltage. Device switches abruptly at appropriate voltage.

Additional model parameters have been added to some devices for compatibility with other commercial simulators. Devices affected are:

Diodes

BJT's

JFET's

MOSFET's

The following devices are implemented in SPICE 3 but are not available with Pulsonix Spice.

Current controlled switch

Uniform Distributed RC Line

Analysis Modes

Two analysis modes available in standard SPICE 3 have not been implemented in Pulsonix Spice. Details follow:

Distortion analysis. (.DISTO) This has not been implemented primarily because the usefulness of this mode is limited. The analysis only provides 2nd and 3rd harmonics which would is inadequate for the majority of applications. Pulsonix Spice is supplied instead with an FFT post processing function which provides a detailed spectral analysis of any signal.

Chapter 2. Quick Start

Tutorials - Overview

This chapter cover a number of tutorials that will help you get started with Pulsonix Spice.

Tutorial 1 is designed for total novices and so if you are experienced with other simulation products and/or EDA tools, you may find its pace a little slow. You can skip to tutorial 2 which makes more assumptions about your knowledge.

Tutorial 2 assumes you have grasped the basics of adding SPICE devices using the Pulsonix Schematic editor. You don't have to worry about setting up analyses or the characteristics of any input stimulus such as V2 in tutorial 1; these procedures will be explained.

If you are an experienced circuit designer but have never used a circuit simulator before, we recommend you read the following notes. This will familiarise you with a few concepts about simulation that may be alien to you if you are used to traditional methods of evaluating circuits.

Simulation for the Novice

When measuring a real circuit, you would probably connect up a power source - bench power supply perhaps - maybe also some signal source, then switch it on and take measurements using - probably - an oscilloscope. You can also make adjustments to the circuit and see the effects straight away.

For simulation, you have a choice of analysis modes and not all of them have an exact "real life" equivalent. The analysis mode that most closely resembles the method of bench testing a circuit described above is ‘Transient Analysis’. This performs an analysis over a specified (by you) time period and saves all the results - i.e. all the voltages and currents in the circuit - to your hard disk. The difference between real life testing and simulation is that the simulation is not running all the time. If you want to see the effects of changing a component value, you have to change it then re-run the simulation.

In order to solve the circuit, the simulator has to calculate the voltage at every node at every time point. Disk space is cheap and plentiful so Pulsonix Spice saves all these values as well as the device currents. Not all simulators do this, some require you to state in advance what you want saved.

This makes it possible to plot voltages and currents after the simulation is complete. You can also place fixed probes on the circuit before running the analysis which will cause the waveform at that point of the circuit to be automatically be displayed while the simulation is running or optionally after its completion.

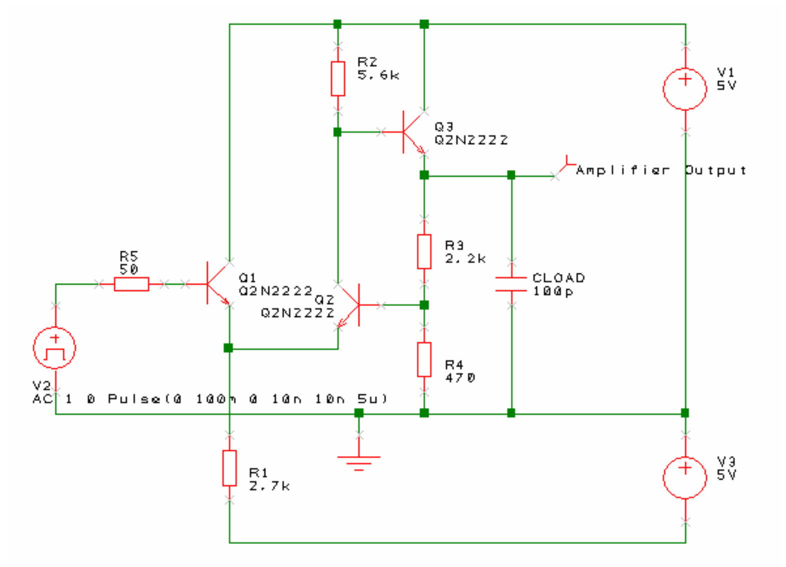
Some of the other analysis modes are: AC analysis which performs a frequency sweep, DC sweep which ramps a voltage or current source and noise analysis which calculates total noise at a specified point and which components are responsible for that noise.

Tutorial 1. A simple ready to run circuit

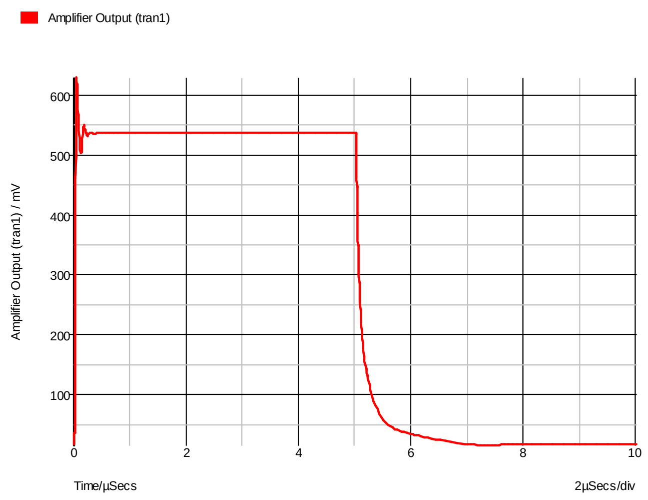
This tutorial demonstrates a basic simulation on a ready to run to circuit. All you need to do is load the circuit and then run it. We will then make a few changes to the circuit and make some extra plots.

▶ Simulating Tutorial 1

1. Click on the **File** menu, click on **Open**. Change folder to **…\Pulsonix\Pulsonix\Examples\Spice\Tutorials**
2. Select **Tutorial1** and click on the **Open** button
   A Schematic window will open with the following circuit:

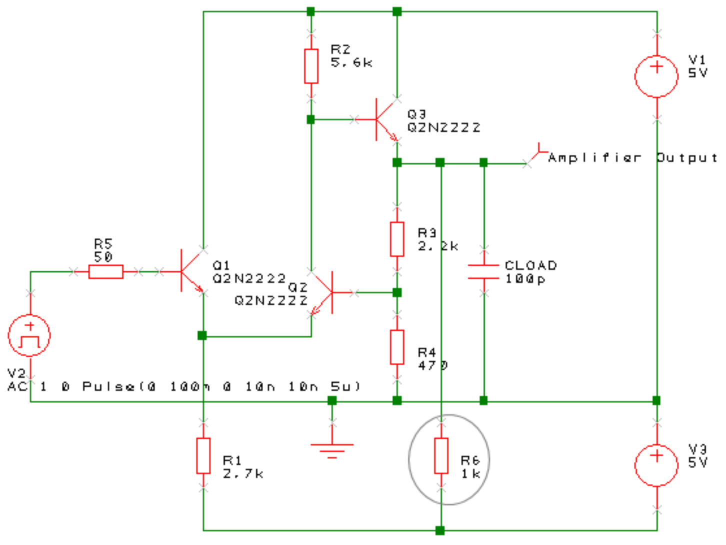


3. This is a simple feedback amplifier designed to amplify a 100mV pulse up to around 500mV. The basic requirement of the design is that the pulse shape should be preserved, DC precision is important but is not critical. The above is our first attempt at a design but has not yet been optimised.
4. This example circuit has been setup to be ‘ready to run’.
5. To start the simulation, from the **Simulation** menu, select **Simulate Simulation** or press the **F9** key on the keyboard. Simulation will not take long, on most machines less than a second.
6. A window will open showing a graph of the output voltage:



7. As you can see, the amplifier doesn't work all that well. There are two problems.
   a) There is substantial ringing on the rising edge, probably caused by the capacitative load.
   b) The falling edge is somewhat sluggish.

The sluggish falling edge is caused by the absence of any standing current in the output emitter follower, Q3. To rectify this, we will place a resistor from the emitter to the -5V rail. The resulting schematic is shown below (R6 has been added and is circled):

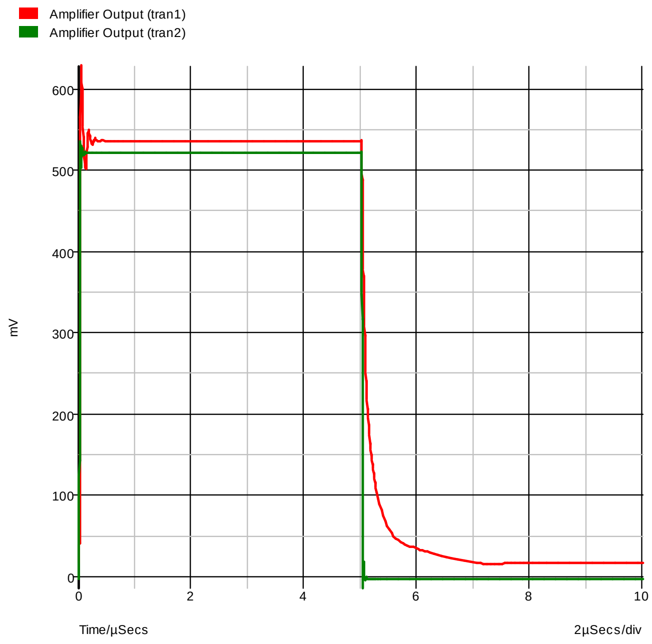


▶ To make the modifications

1. Ensure that the Parts popup toolbar is displayed, if not right click on one of the toolbars and select **Parts** from the shortcut menu. From this menu you can also switch on the Simulation toolbar if not already displayed.

2. From the **Parts** toolbar, select the **Passives** button to popup the **Passives** toolbar. Select the **Resistor (Box Shape)** button or if you prefer, the **Resistor (Z Shape)** button.
3. A resistor symbol will appear. Move the resistor into the location shown in the diagram above. Left click to drop it in place.
4. Another resistor symbol will appear. Cancel this by pressing the Escape (ESC) key on the keyboard.
5. The resistor will be automatically named to R6, the next available number in the'R' sequence.
6. Now connect the resistor.
7. Drag a connection from each of the terminals on the resistor to connect it into the circuit. To be sure you have made the connection, look out for the finish cursor when the connection is over a legal connection point.
8. Alternatively, use the Insert Connection option from the menu or toolbar to make the connections.
9. Re run the simulation by pressing **F9**. The graph will now be updated to look like this:

The graph will now be updated to:



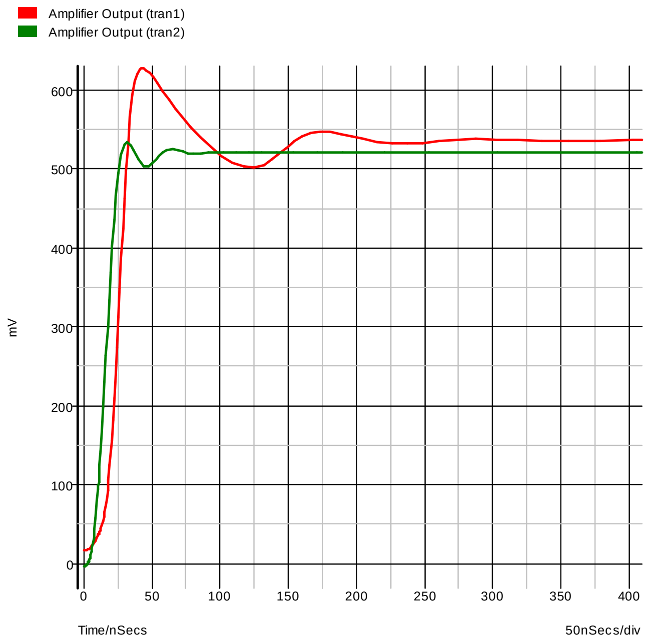
The green trace(the cleaner of the two) is the output after the modification. As you can see, the problem with the trailing edge has been fixed and the ringing is much improved.

Now let's have a look at the ringing in more detail. To do this, we need to zoom in the graph by adjusting the limits of the axes. There are two ways of doing this. The quickest is to simply drag the mouse over the region of interest. The other method is to manually enter the limits using the **edit axis** dialog.

▶ To zoom with the mouse

1. Make sure that the graph window is selected by clicking in its title bar.
2. Place the cursor at the top left of the region of interest i.e. to the left of the y-axis and above the top of the red curve.
3. Press the left mouse key and while holding it down, drag the mouse to the bottom right of the area you wish to zoom in. You should see a rectangle appear as you drag the mouse.

Release the mouse key, you should see the following:

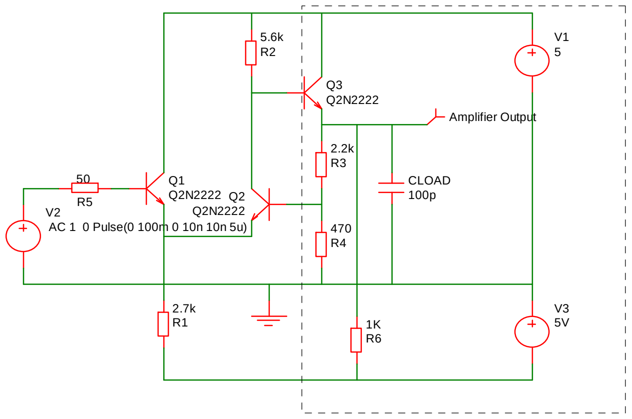


If you don't get it quite right, press the **Undo** (Zoom) button on the toolbar to return to the previous view.

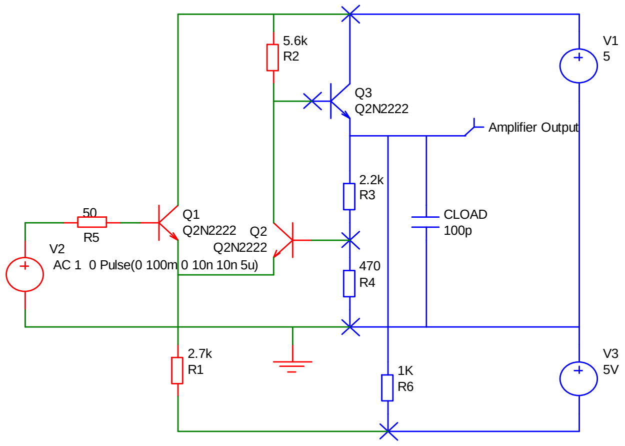
You can improve the ringing by adding a small phase lead in the feedback loop. This can be done by connecting a small capacitor between the emitter of Q3 and the base of Q2. There isn't room to add this tidily at present, so first, we will move a few components to make some space.

▶ To make the space required

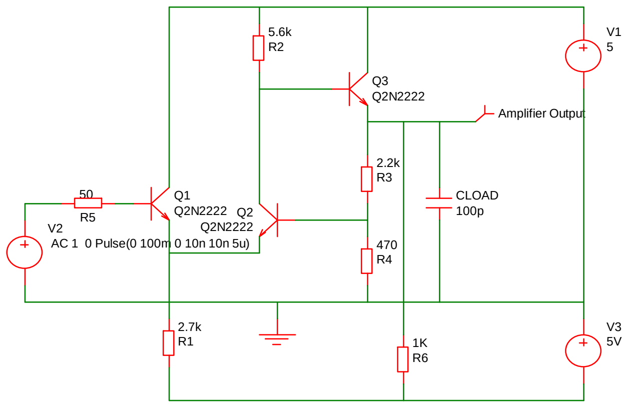
1. On the Schematic design, drag a window over the region shown in dotted lines on the diagram below:



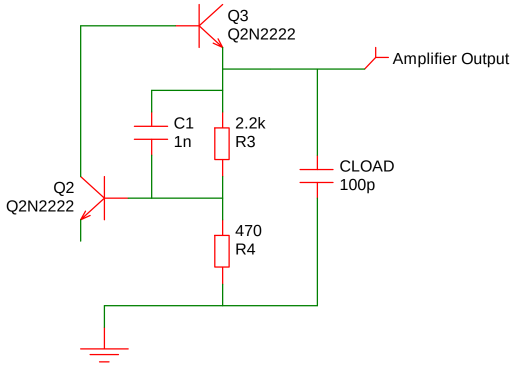
2. As you drag the mouse, a rectangle will appear.
3. Release the mouse. The area enclosed will change colour to the selected colour. If extra connection segments are selected they can be removed from the selection using Ctrl-Click.



4. The selected connections and Components can be moved by dragging the selection. Move it to the right to give you space on the right of Q2.
5. Move it by a number of grid spaces and release it.
6. Click in ‘free’ space to deselect the group. You will now see:



7. Connect in the capacitor C1 as shown below using a similar procedure as for the resistor R6. Select the Capacitor from the **Parts** popup toolbar.

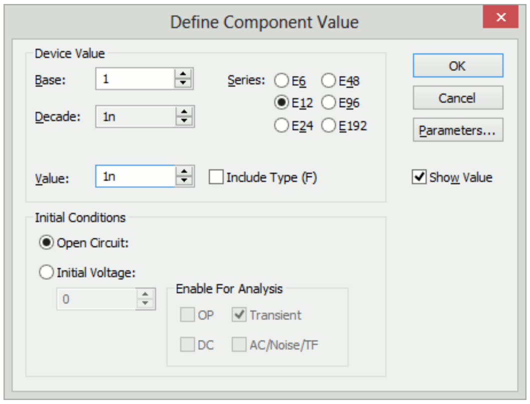


8. 1nF is obviously far too high a value so we will try 2.2pF.

▶ To change the Capacitor value

1. To change the component's value first select C1. To do this, left click on it.

2. Either press **F7** or right-click on the Component C1 and from the shortcut menu, select **Edit Spice Value/Model...**. You will see the following dialog appear:


Define Component Value

Device Value

Base: 1

Decade: 1n

Series: ○ E6 ○ E48 ● E12 ○ E96 ○ E24 ○ E192

Value: 1n

☐ Include Type (F)

Initial Conditions

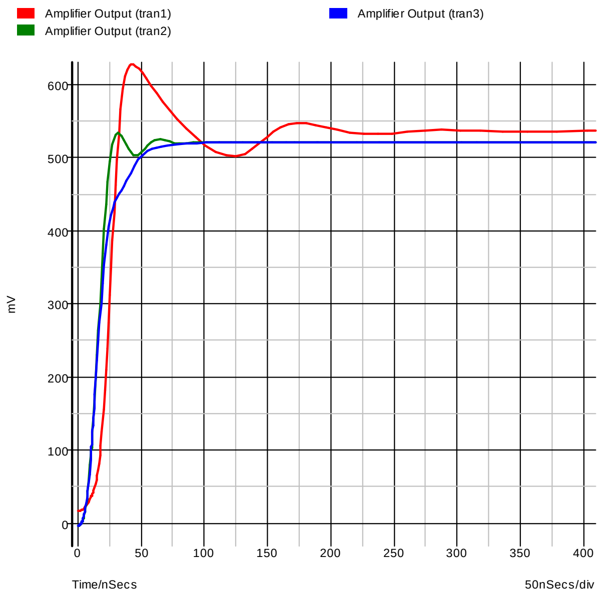
● Open Circuit:

○ Initial Voltage: 0

Enable For Analysis

☐ OP ☑ Transient ☐ DC ☐ AC/Noise/TF

3. You can type the new value in directly in the **Value** box or you can select a value using the mouse alone with the up and down arrow buttons.
4. Change the value to 2.2pf and press **OK**.
5. Now re-run the simulation by pressing <**F9**>.
6. This is the result you should see:



The blue curve is the latest result. This is now a big improvement on our first attempt.

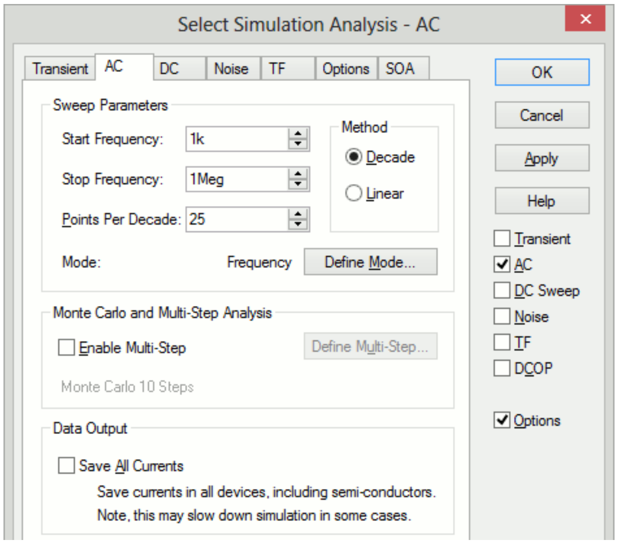
We will now round off tutorial 1 by introducing AC analysis.

AC analysis performs a frequency sweep over a specified frequency range.

Introducing AC Analysis

▶ To make the modifications

1. In the Schematic window, select the **Simulation** menu and select **Simulation Parameters**.



Select Simulation Analysis - AC

Transient | AC | DC | Noise | TF | Options | SOA

Sweep Parameters

Start Frequency: 1k

Stop Frequency: 1Meg

Points Per Decade: 25

Method: [✓ Decade] Linear

Mode: Frequency

Define Mode...

Monte Carlo and Multi-Step Analysis

☐ Enable Multi-Step

Define Multi-Step...

Monte Carlo 10 Steps

Data Output

☐ Save All Currents

Save currents in all devices, including semi-conductors. Note, this may slow down simulation in some cases.

OK | Cancel | Apply | Help

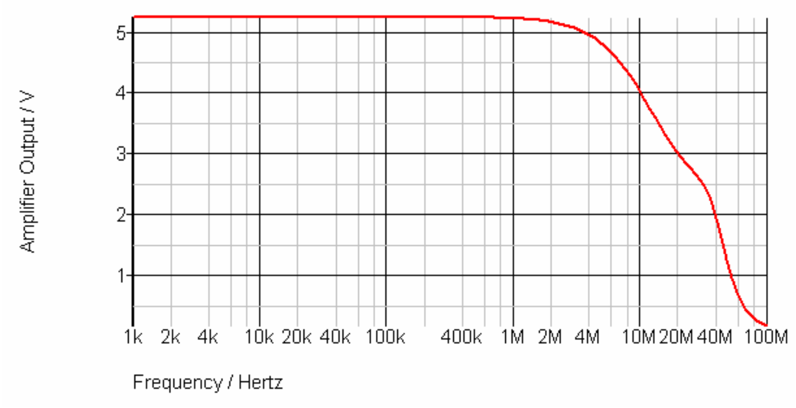
☐ Transient
☑ AC
☐ DC Sweep
☐ Noise
☐ TF
☐ DCOP

☑ Options

2. Select the AC tab, ensure that the **Start Frequency** is set to **1K**, the **Stop Frequency** is set to **1Meg** and that **Points Per Decade** is set to **25**. The **Decade** radio button should be selected.

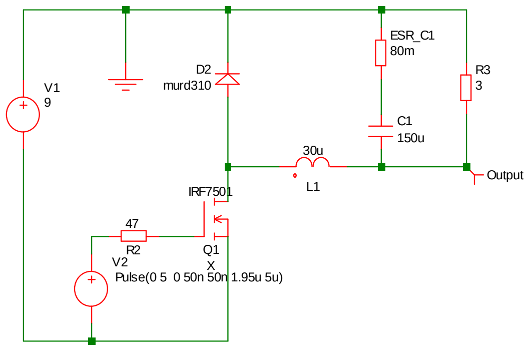
3. Click **OK** to close the dialog and accept the values.

Now run the simulation again by pressing <**F9**>. The following graph appears.



Tutorial 2. A Simple SMPS Circuit

In this tutorial we will simulate a simple SMPS switching stage to demonstrate some of the more advanced graphing and waveform analysis facilities available with Pulsonix Spice.

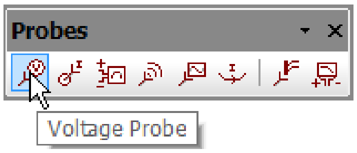


You can either load this circuit from **…Pulsonix\Examples\Spice\Tutorials\Tutorial2** or alternatively you can enter it from scratch. The latter approach is a useful exercise in using the schematic editor to add SPICE components.

▶ To simulate

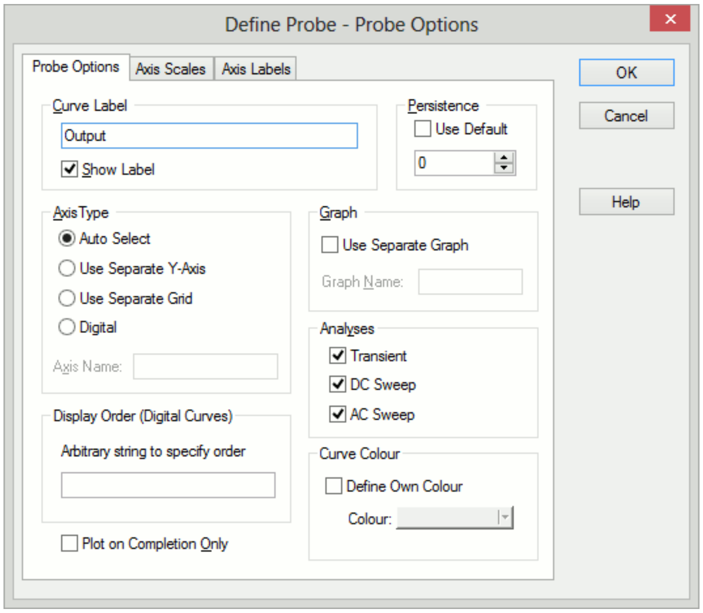
1. If creating the design from scratch, place the Components and connections as shown above.

2. The "Output" probe (on the right side of the design) can be selected from either the **Parts** toolbar and **Probes** popup toolbar using the **Voltage Probe** button**,** or from the **Simulation** menu and **Insert Fixed Probe, Voltage Probe.**



3. The voltage probe uses a single pin symbol, so you can release it on top of the connection required and it will automatically ‘weld’ itself to this net making itself fully connected.

4. After placing the output probe, select it, then press **F7** to edit its label. Enter the name **Output** to identify the curve when simulated. Enter the text in the **Curved Label** box. All the other options may be left at their defaults.


Define Probe - Probe Options

Probe Options | Axis Scales | Axis Labels

Curve Label: Output
[✓ Show Label]

Persistence: Use Default; 0

Axis Type: [✓ Auto Select] Use Separate Y-Axis; Use Separate Grid; Digital
Axis Name:

Graph: Use Separate Graph
Graph Name:

Analyses: [✓ Transient] [✓ DC Sweep] [✓ AC Sweep]

Display Order (Digital Curves)
Arbitrary string to specify order

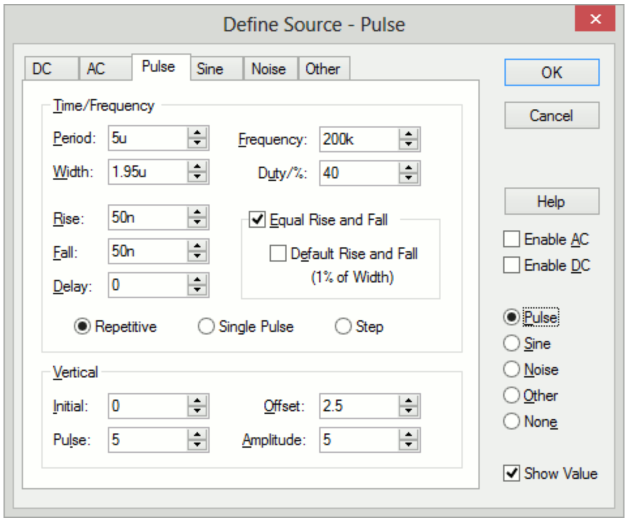
Curve Colour: Define Own Colour
Colour:

Plot on Completion Only

5. For the pulse source **V2**, you should use a **Universal Source**.

6. From the **Sources** popup toolbar, select the **Universal Source** button.

7. When added, select V2 and press **F7**. Edit the settings as shown below:


Define Source - Pulse

DC | AC | Pulse | Sine | Noise | Other

Time/Frequency
Period: 5u
Frequency: 200k
Width: 1.95u
Duty/%: 40
Rise: 50n
[✓ Equal Rise and Fall]
Fall: 50n
Default Rise and Fall (1% of Width)
Delay: 0
[✓ Repetitive] Single Pulse Step

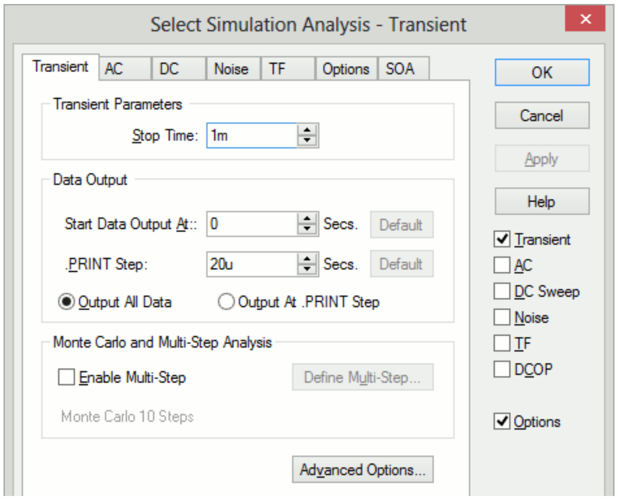
Vertical
Initial: 0
Offset: 2.5
Pulse: 5
Amplitude: 5

Enable AC
Enable DC
[✓ Pulse] Sine Noise Other None
[✓ Show Value]

8. This sets up a 200kHz 5V pulse source with 40% duty cycle and 50nS rise time.

9. Set up the simulation parameters by selecting the **Simulation** menu and **Simulation Parameters** option.

10. Select the **Transient** tab on the dialog. In **Transient Parameters**, set the **Stop Time** to **1m**.


Select Simulation Analysis - Transient

Transient | AC | DC | Noise | TF | Options | SOA

Transient Parameters
Stop Time: 1m

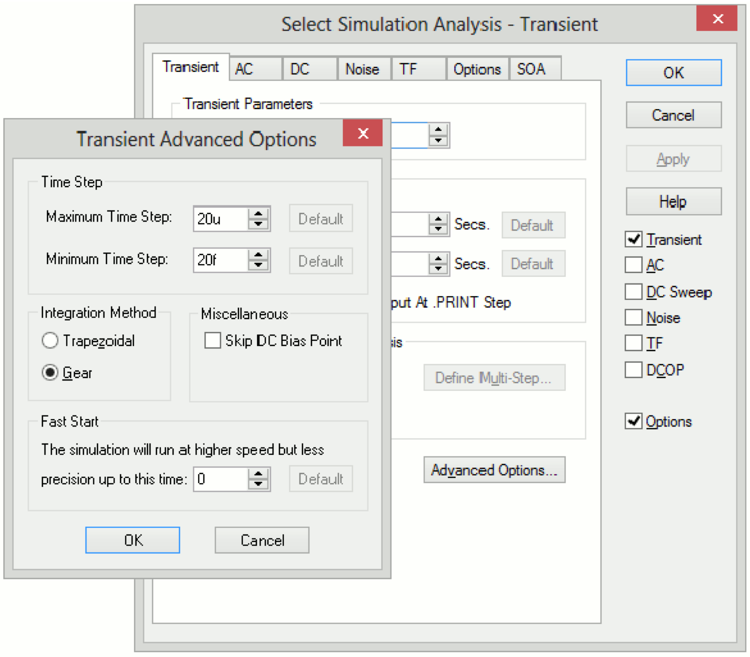
Data Output
Start Data Output At:: 0 Secs.
.PRINT Step: 20u Secs.
[● Output All Data] ○ Output At .PRINT Step

Monte Carlo and Multi-Step Analysis
☐ Enable Multi-Step
Monte Carlo 10 Steps

[✓ Transient] ☐ AC ☐ DC Sweep ☐ Noise ☐ TF ☐ DCOP [✓ Options]

11. Select the **Advanced Options** button.

12. In the **Integration Method** section, check the radio button **Gear**. This improves the simulation results for this type of circuit. You will still get sensible results without checking this option, they will just be a little better with it.


Transient Advanced Options

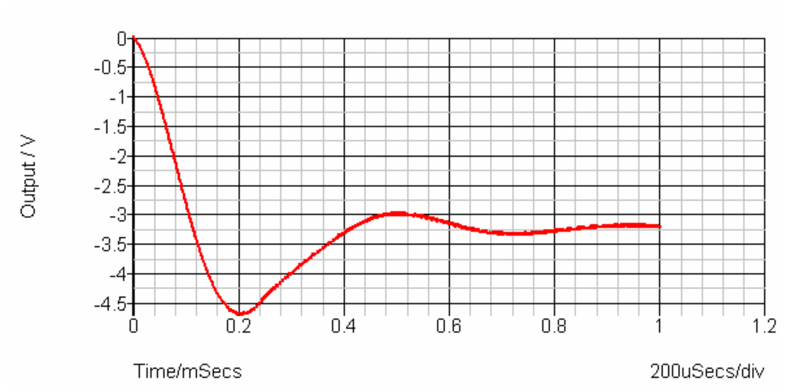
Time Step
Maximum Time Step: 20u
Minimum Time Step: 20f

Integration Method
○ Trapezoidal
[● Gear]

Miscellaneous
☐ Skip DC Bias Point

Fast Start
The simulation will run at higher speed but less precision up to this time: 0

13. If you have any Pulsonix Spice graph windows open, you should now close them. Once you have loaded or entered the circuit, press **Run Simulation** from the **Simulation** menu to start the simulation. This is the graph you will see

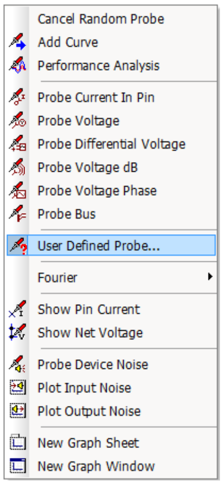


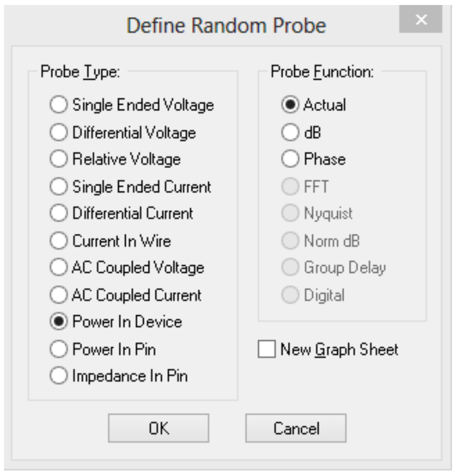
The circuit is the switching stage of a simple step-down (buck) regulator designed to provide a 3.3 V supply from a 9V battery. The circuit has been stripped of everything except bare essentials in order to investigate power dissipation and current flow. Currently, it is a little over simplified as the inductor is ideal. More of this later. We will now make a few measurements.

Let’s have a look at the inductor current. We could have viewed the inductor current by placing a current probe on the pin of the device before starting the simulation. Here, instead, we will plot the device current *after* the completion of the simulation.

▶ To measure power dissipation

1. From the Schematic design, click on the **Simulation** menu and **Random Probe**. With the Random Probe cursor shown in design, right click the mouse.
2. From the shortcut menu, select **User Defined Probe**.
3. From the dialog, select the **Probe Type** to be **Power In Device** and the **Probe Function** to be **Actual**.




Define Random Probe

Probe Type:

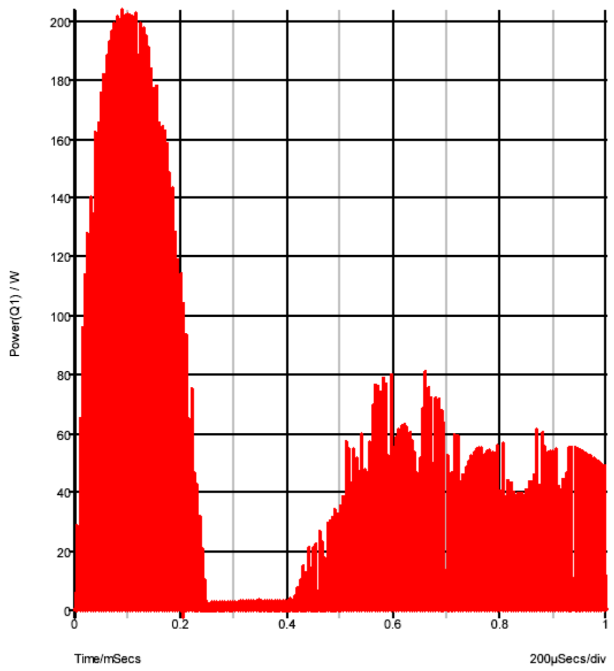
- Single Ended Voltage
- Differential Voltage
- Relative Voltage
- Single Ended Current
- Differential Current
- Current In Wire
- AC Coupled Voltage
- AC Coupled Current
- [✓ Power In Device]
- Power In Pin
- Impedance In Pin

Probe Function:

- [✓ Actual]
- dB
- Phase
- FFT
- Nyquist
- Norm dB
- Group Delay
- Digital

New Graph Sheet

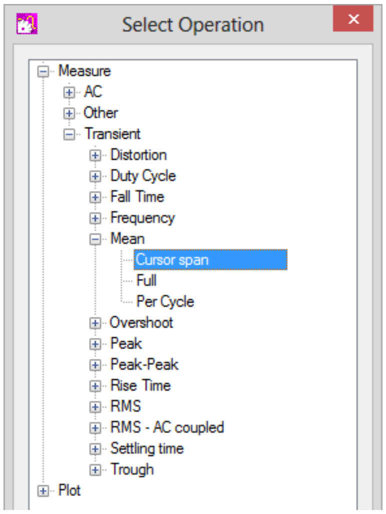
4. Select the **New Graph Sheet** check box.
5. Click **OK** to confirm the selection.
6. Left click on Q1 to use the probe.
7. This is what you should see:



8. This shows a peak power dissipation of 200W although you are probably more interested in the average power dissipation over a specified time.

▶ To display the average power dissipation over the analysis period

1. In the graph window, use menu Measure and Mean.
2. This should display a value of 517 mW. This is the average power over the whole analysis period of 1mS. You can also make this measurement over any period you select using the cursors.
3. Zoom in the graph at a point around 100uS, i.e. where the power dissipation is at a peak.
4. Switch on graph cursors using the Cursors and Toggle On/Off option on the right mouse menu. There are two cursors represented by cross-hairs. One uses a long dash and is referred to as the main or measurement cursor. When first switched on the reference cursor is positioned to the left of the graph and the main to the right.
5. Position the cursors to span a complete switching cycle. There are various ways of moving the cursors. To start with the simplest is to drag the vertical hairline left to right. As you bring the mouse cursor close to the vertical line you will notice the cursor shape change. See “Graph Cursors” for other ways of moving cursors.
6. Press F3 or select analysis menu (right click in legend panel) More Functions... From the tree list select **Measure** and **Transient** and **Mean** and **Cursor Span**:



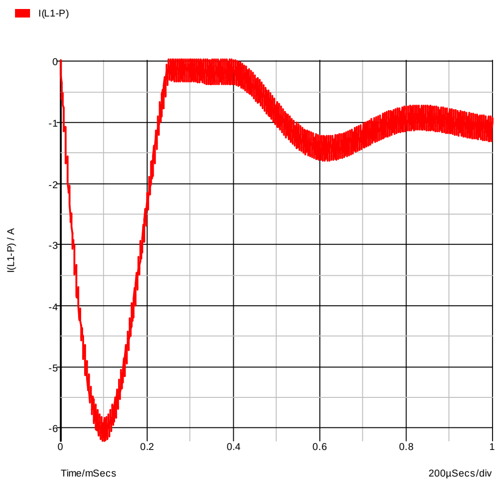
You should see a value of about 2.8W displayed. This is somewhat more than the 517mW average but is still well within the safe operating area of the device. However, as we noted earlier, the inductor is ideal and does not saturate. Let’s have a look at the inductor current.

▶ To use the Random Current Pin Probe

1. From within the Pulsonix Schematic window, select the **Simulation** menu.
2. Select **Random Probe**. Right click, and from the shortcut menu select **New Graph Sheet**. A new simulation graph will appear.

3. Right mouse click again to display the same shortcut menu, this time select **Probe Current In Pin**.

4. Click on the left pin of the inductor L1 (marked pin ‘P’). This is what you will see in the graph window:



This shows that the operating current is less than 1.5A but peaks at over 6A. In practice you would want to use an inductor with a maximum current of around 2A in this application; an inductor with a 6A rating would not be cost-effective. We will now replace the ideal component, with something closer to a real inductor.

▶ To replace inductor L1

1. Press the <**Esc**> key to exit **Random Probe** mode.

2. Left click to select the inductor component L1 in the schematic design.

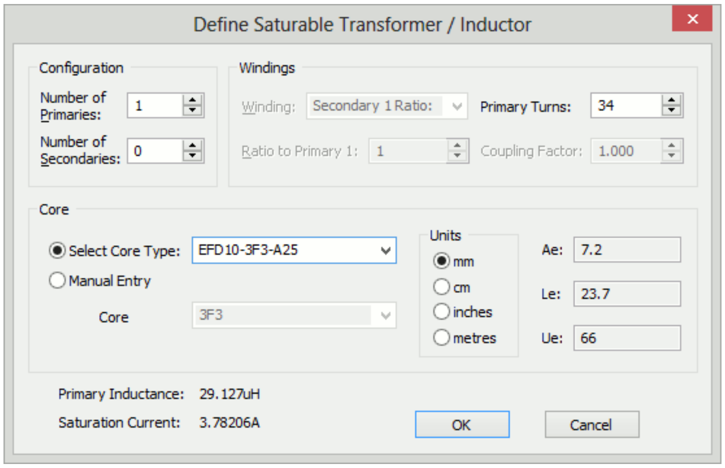
3. Click the **Del** key on the keyboard.

4. The two connections which were previously attached to L1 will now be left ‘hanging’ in free space.

5. If the **Part Browser** is not already running, select it from the **View | Dockable Windows** menu. Make sure that Spice category is displayed, if not use **Change Part Category** from the right hand menu and add Spice category to the list.

6. From the Spice Category list, select **Passives**, then **Saturable Transformer**. Drag off the Part Browser view to add this device to the design.

7. You will be presented with the Define Saturable Transformer/Inductor dialog:


Define Saturable Transformer / Inductor

Configuration
Number of Primaries: 1
Number of Secondaries: 0

Windings
Winding: Secondary 1 Ratio:
Primary Turns: 34
Ratio to Primary 1: 1
Coupling Factor: 1.000

Core
(●) Select Core Type: EFD10-3F3-A25
( ) Manual Entry
Core: 3F3

Units
(●) mm
( ) cm
( ) inches
( ) metres

Ae: 7.2
Le: 23.7
Ue: 66

Primary Inductance: 29.127uH
Saturation Current: 3.78206A

8. Select number of Primaries to 1 and number of Secondary's to 0 for and inductor.
9. Select **EFD10-3F3-A25** in the **Select Core Type:** drop down list. This is part number for a Philips ferrite core. Enter 34 in the turns edit box. Units should be mm. All other information will be automatically calculated.
10. Drag it into the space previously occupied by L1, rotate and mirror as necessary to make sure pin 1 is on the left.
11. Press **Esc** to cancel the addition of a second copy of this component.
12. If you placed the inductor in exactly the same location as L1 it will 'auto weld' attaching the connections back to the two terminals of the new component. You will now not see pin crosses in the design.
13. Click **OK** to confirm the values and press **F9** to simulate again.
14. Select the graph sheet that displayed the inductor current by clicking on its tab at the top of the graph window. Now select the simulation menu in the Pulsonix Schematic editor.
15. Select Random Probe. From the shortcut menu select Probe Current In Pin.
16. Click on the left hand inductor pin again. You will notice the peak current is now in excess of 45A. This is of course because the inductor is saturating. You can also measure the peak power over 1 cycle.

▶ To measure the peak power

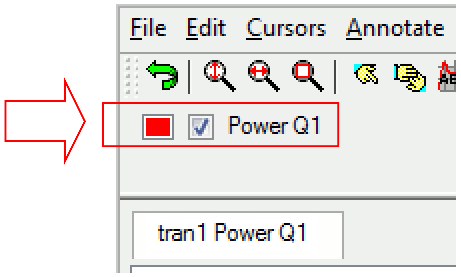
1. In the graph window, select the graph sheet tab with the power plot.
2. In the schematics window use **Random Probe** and, from the shortcut menu, select **User Defined Probe**.
3. From the dialog, select the Probe Type to be Power In Device, the Probe Function to be Actual and uncheck the New Graph Sheet box.
4. Left click on the Q1 component.

5. In the graph window, zoom out to see the full graph using the **Fit Window** button from the toolbar.

6. Zoom in on the peak power.

7. Position the cursors to span a full cycle. (The cursors are currently tracking the first power curve. This doesn't actually matter here as we are only interested in the x-axis values. If you want to make the cursors track the green curve, you can simply pick up the cursor at its intersection with the mouse and drag it to the other curve).

8. We now have two curves on the graph so we must select which one with which we wish to make the measurement. To do this check the box.



9. As before, press <F3> in the graph window, then select Measure|Transient|Mean|Cursor Span. A value of about 11-12W should be displayed, rather substantially higher than before.

Tutorial 3. Installing 3rd Party Models

In this tutorial, we will install a device model library. For this exercise, we have supplied a model library file - TUTORIAL3.MOD - with just two devices. These are:

SXN1001 - an NPN bipolar transistor

SXOA1000 - an Opamp.

Both are totally fictitious.

You will find this file in the tutorial folder **Examples|Spice|Tutorials|Tutorial3.mod**. There are two aspects to installing a model. Pulsonix Spice needs to know where within your file system, the model is located. If the model is to be listed in the Pulsonix Schematics system using the **Insert Part By Model** option, then Pulsonix Spice also needs to know what part to use for it in the Schematic and what category it comes under.

▶ To Insert Part By Model

1. Open windows explorer or click on My Computer or open other file manager of your choice.
2. Locate TUTORIAL3.MOD in *EXAMPLES*\Tutorials (see “Examples and Tutorials – Where are They?” Pick the file up and drop into the Pulsonix Spice command shell. That is, drop it in the window where Pulsonix Spice messages are displayed. If you can’t see the command shell because it is obscured, select any Pulsonix Spice window then press the space bar.

A message box will appear asking you to confirm you wish to install the file. Click OK.

The message “Making device catalog. This may take some time, please wait….” will be displayed.

At this stage, Pulsonix Spice knows where to find our fictitious models. You will find that it also knows about the NPN transistor as the following demonstrates:

In the Schematics window, open a new Schematic design.

From the Simulation menu use **Insert Part Using Model.**

From the dialog select **NPN** from the left hand list.

In the model list search for our new transistor, SXN1001. Using the filter string “SX*” will help.

Select the device and press **OK** to place it on the Schematic sheet.

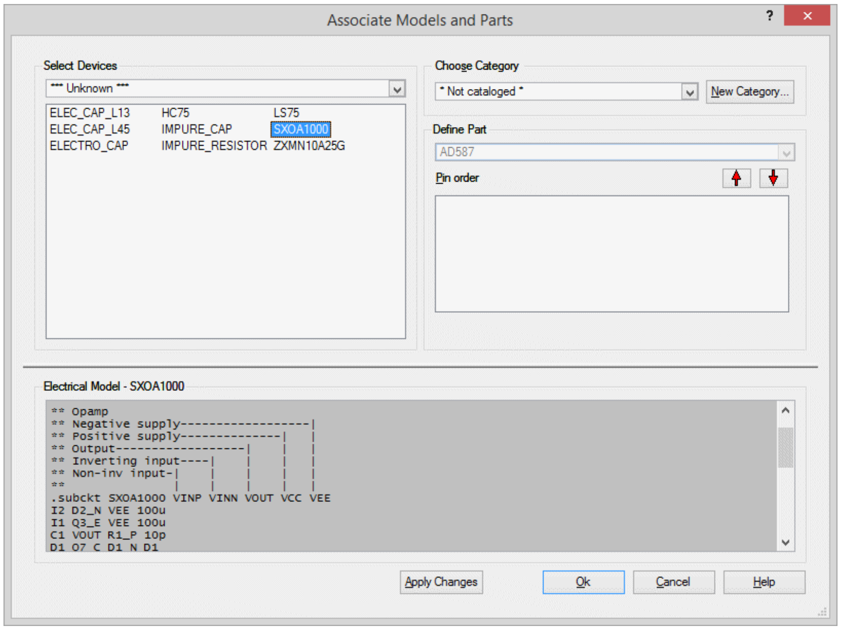
Without you having to tell it, Pulsonix Spice already knew that the SXN1001 is an NPN transistor. This is because it is a *primitive* device defined using a .MODEL control. Such devices are built in to the simulator and what the device is can be determined from the model contained in TUTORIAL3.MOD.

The other device in the model library is an op-amp and is defined as a *sub-circuit*. This is a module made up of other components, in this case BJT's, diodes, resistors and current sources. If you repeat the above procedure for the opamp device - SXOA1000 - by looking under the Op-amps category, you will find it is not there.

In order to arrange for the device to be listed in the **Insert Part Using Model**, you need to associate the model with a Pulsonix Part.

▶ To associate Models with Parts

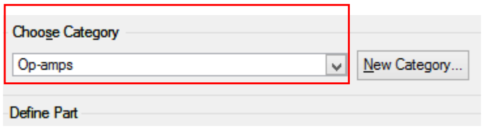
1. In the simulator **command shell**, from the **File** menu, select **Model Library**> and **Associate Models and Parts...**
2. You should see the opamp - **SXOA1000** - listed in the **Select Devices** box on the left, select it. This is what you will see:



Electrical Model - SXOA1000

```
** Opamp
** Negative supply------------------|
** Positive supply--------------|   |
** Output------------------|    |   |
** Inverting input----|    |    |   |
** Non-inv input-|    |    |    |   |
**               |    |    |    |   |
.subckt SXOA1000 VINP VINN VOUT VCC VEE
I2 D2_N VEE 100u
I1 Q3_E VEE 100u
C1 VOUT R1_P 10p
D1 O7 C D1 N D1
```

3. You must now supply a category and a Part:
4. From the list box immediately below the **Choose Category** label, select **Op-amps**
5. From the list box below the **Define Symbol** label, select **Opamp** from the list of Parts with the same number of pins as your model.



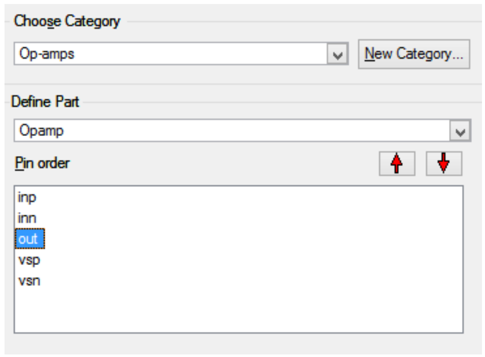
We have not quite finished yet. Our new op-amp has the wrong pin out for the Schematic Part. The pin order for the Part is shown in the third box down on the right and is as follows:

Pin Name	**Function**
inp	*Non-inverting input*
inn	*Inverting input*
vsp	*Positive supply*
vsn	*Negative supply*
out	*Output*

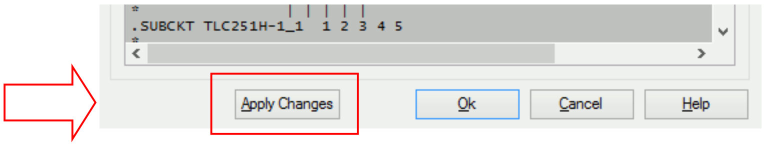
The text at the bottom of the dialog box shows the actual sub-circuit definition. Fortunately, it has been annotated with the function of each of the sub-circuit's terminals. (This is in fact usually the case with third-party models). As you can see, the output terminal is in the wrong place. We can change the pin order using the **Pin order** up and down buttons:

1. Select "out" in the pin order box.
2. Click twice on the ‘up’ button so that it is positioned between "inn" and "vsp"

This is what you will now see:



3. To finish, press the **Apply Changes** button at the bottom of the dialog. If you simply press OK, your changes will not be saved.



```
.SUBCKT TLC251H-1_1  1 2 3 4 5
```

4. Now press **OK** to exit.

You will now find the op-amp listed under the **Op-amps** category in the Schematic **Insert Part Using Model** dialog.

Notes: You will not always need to execute the above procedure to associate models and parts even for sub-circuit devices. Pulsonix Spice is supplied with a data base of over 30000 devices that are already associated. These are devices for which SPICE models are known to be available from third party sources. This database is in the file ALL.CAT which you will find in the Pulsonix Spice installation support folder. The information you enter in the associate models dialog is stored in a file called USER_V2.CAT which you will find in the application data directory folder.

Finally, there is a method of embedding association information within the model itself, and such models will not require manual association. The embedding method is described in the section Embedded Association.

Chapter 3. Getting Started

This chapter describes the basic operation of Pulsonix Spice and is aimed primarily at novice users.

The basic steps to simulate a circuit are as follows:

1. Enter the circuit using the schematic editor. Refer to Pulsonix manual.
2. Prepare circuit for simulation by defining stimuli (e.g. pulse, sine. See below) and what type of analysis to perform. (Transient, AC frequency sweep, DC sweep etc. see below)
3. Run simulator
4. Graph results

The following paragraphs briefly describe these steps. More details are given in other sections.

Circuit Stimulus

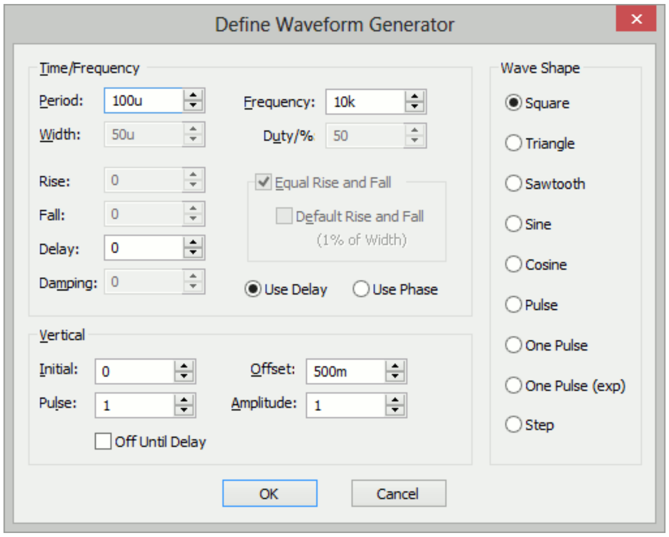
Most analysis modes require some kind of signal source or *stimulus*. For analog circuits these are nearly always provided by either a voltage source or a current source such as a pulse or sine wave. Digital signals can be created using the digital pulse device. Such signals - or stimuli - are specified using a special source component which is placed on the schematic in the usual way. There are generic source components **Fixed Current Source** and **Fixed Voltage Source** provided in library **SPICE.cml**.

The **Digital Pulse** device and the analog **Pulse Generator** device perform a similar task. You should use the **Pulse Generator** to drive analog inputs and **Digital Pulse** to drive digital inputs. They can be interchanged but the simulation will be less efficient.

Waveform Generator

This is used to create a time domain signal for transient analysis. To place one of these devices, select the Simulation menu |Insert Source | Waveform Generator for a voltage source or Simulation | Insert Source | Current Waveform Generator for a current source.

Once the Source has been added to the design, you can specify the signal for it. Select the source component, then from the shortcut menu select **Edit Value/Model...** or simply click the <**F7**> key. For Analog sources, this will bring up the dialog shown below:



Define Waveform Generator

Time/Frequency

Period: 100u
Frequency: 10k
Width: 50u
Duty/%: 50
Rise: 0
Fall: 0
Delay: 0
Damping: 0
[✓] Equal Rise and Fall
[ ] Default Rise and Fall (1% of Width)
[✓ Use Delay] Use Phase

Vertical

Initial: 0
Offset: 500m
Pulse: 1
Amplitude: 1
[ ] Off Until Delay

Wave Shape

[✓ Square] Triangle Sawtooth Sine Cosine Pulse One Pulse One Pulse (exp) Step

Select the wave shape on the right hand side then enter the parameters as appropriate. The following notes provide details on some of the controls.

- Damping describes an exponential decay factor for sinusoidal wave-shapes. The decay is governed by the expression:

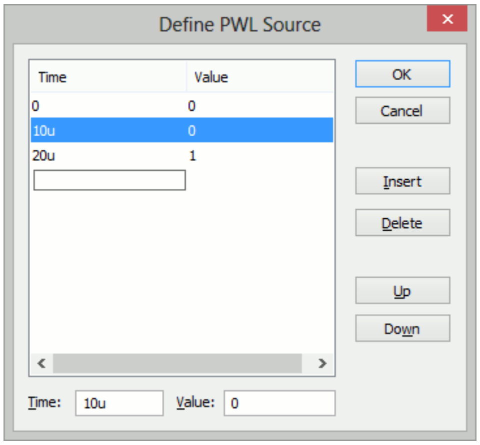
  $e^{-damping*t}$

- Off until delay if checked specifies that the signal will be at the Initial value until the delay period has elapsed.
- Note that some parameters can be specified in more than one way. For example both frequency and period edit controls are supplied. Changing one will cause the other to be updated appropriately. The same applies to duty and width and the vertical controls in the lower half.

PWL Source

This device can be used to describe a piece wise linear source. A PWL source can describe any arbitrary wave shape in terms of time-voltage or time-current pairs. To place a PWL source select the Simulation|Insert Source menu and PWL Source or PWL Current Source.

To edit the device, select it and press <**F7**>. This will open the Edit PWL Dialog which allows you to enter time and voltage/current values.



Time	Value
0	0
10u	0
20u	1

Time: 10u Value: 0

Power Supply/Fixed Current Source

Select the Simulation|Insert Source menu Power Supply or DC Current Source to place a fixed voltage or current source.

This selects the value used for the DC bias point analysis. Note that if a transient source is specified (pulse, sine, noise, pwl, pwlfile, sffm or exp) this value will be ignored and the value at time=0 will be used instead. The push buttons are convenient way of selecting commonly used power supply voltages.

AC Source

This specifies magnitude and phase for ACs Weep and Transfer Function analysis. Note that noise analysis also requires at least one source with an AC specification.

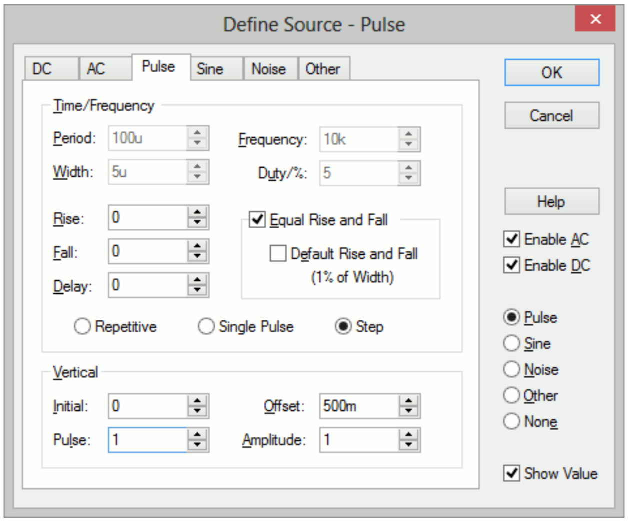
To place an AC Source select menu Simulation|Insert Source and use AC Source or AC Current Source.

Universal Source

There is also a Universal source which provides the function of transient, AC and DC sources all in one device. In addition, the Universal source may be used to create a random noise source.

To place a universal source, select menu Simulation|Insert Source|Universal Source or Simulation|Insert Source|Universal Current Source.

To edit a universal source, select the device and press <**F7>** or popup menu **Edit Part**…. This will display the following dialog box:


Define Source - Pulse

Tabs: DC, AC, Pulse, Sine, Noise, Other

Time/Frequency
Period: 100u
Frequency: 10k
Width: 5u
Duty/%: 5
Rise: 0
Fall: 0
Delay: 0
[✓] Equal Rise and Fall
[ ] Default Rise and Fall (1% of Width)
○ Repetitive ○ Single Pulse [● Step]

Vertical
Initial: 0
Offset: 500m
Pulse: 1
Amplitude: 1

OK
Cancel
Help
[✓] Enable AC
[✓] Enable DC
[● Pulse] ○ Sine ○ Noise ○ Other ○ None
[✓] Show Value

Pulse, sine, noise and DC waveforms may be specified using the tab sheets of the same name. You can also specify a Piece wise linear source, exponential source or a single frequency FM source. To enter one of these select the Text tab and enter the appropriate syntax for the source. Please refer to Voltage source in the *Simulator Reference Manual* for more information on these sources. The AC sheet is for AC analysis only.

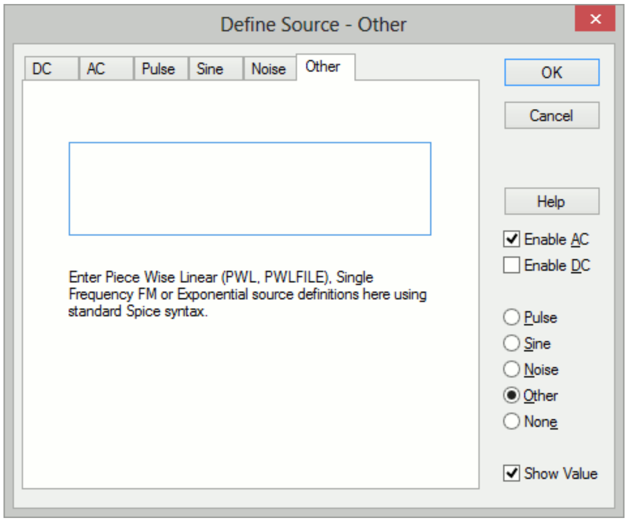
With the universal source, you can specify transient, AC and DC specifications simultaneously. This is not possible with any of the other sources.

Digital Pulse Source

This specifies the component to be XPSICE device and mode specifically a digital pulse generator. Use the <**F7**> key to bring up the dialog where values can be entered.

Other Tab on Universal Sources

The **Other** tab is used to specify the source device as another of the SPICE transient source types.



Enter Piece Wise Linear (PWL, PWLFILE), Single Frequency FM or Exponential source definitions here using standard Spice syntax.

Enable AC [✓]
Enable DC [ ]
[✓ Other]
Show Value [✓]

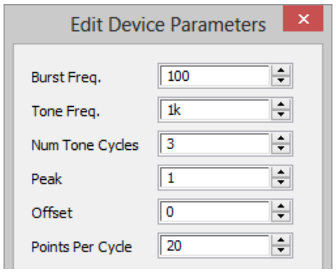
Enter the correct SPICE netlist format for the source device. This includes - Piece wise linear source, Exponential and Single frequency FM sources.

Other Sources

Sine Tone Burst

Generates a sequence of sinusoidal bursts with a user defined number of cycles per burst, burst frequency and tone frequency.

Use **Add Component** to add the Part named **Sine Wave Burst Voltage Source**. Editing the device using the <**F7**> key will display a dialog with 6 parameters:



Burst Freq.	100
Tone Freq.	1k
Num Tone Cycles	3
Peak	1
Offset	0
Points Per Cycle	20

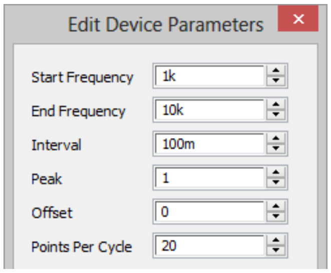
Parameter	**Description**
Burst Freq.	Burst frequency
Tone Freq.	Frequency of the sinusoidal tone
Num Tone Cycles	Number of sinusoidal cycles in each burst
Peak	Peak voltage

Offset	Offset voltage
Points Per Cycle	Minimum number of time steps in each sinusoidal cycle. Increasing this number will improve the accuracy of the simulation at the expense of simulation speed

Swept Sinusoid

Generate a sinusoidal signal with linearly increasing frequency.

Use **Add Component** to add the Part named **Sine Wave Sweep Voltage Source**. Editing the device using the <**F7**> key will display a dialog with 6 parameters:



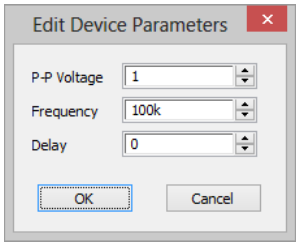
Start Frequency	1k
End Frequency	10k
Interval	100m
Peak	1
Offset	0
Points Per Cycle	20

Parameter	**Description**
Start Frequency	Starting frequency
End Frequency	Frequency at the end of the ramp Interval Time taken to ramp from start frequency to end frequency.
Peak	Peak voltage
Offset	Offset voltage
Points Per Cycle	Minimum number of time steps in each sinusoidal cycle. Increasing this number will improve the accuracy of the simulation at the expense of simulation speed.

Bidirectional Pulse

Generates a symmetrical bidirectional pulse waveform.

Use **Add Component** to add the Part named **PP Voltage Source**. Editing the device using the <**F7**> key will bring up a dialog with 3 parameters:

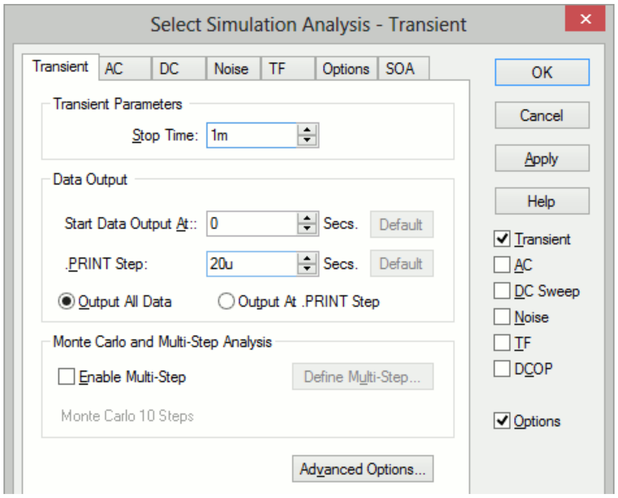


P-P Voltage	1
Frequency	100k
Delay	0

Parameter	Description
P-P Voltage	Peak to peak voltage
Frequency	Pulse frequency
Delay	Delay after start

Analysis Modes

Analysis mode is setup by selecting the **Simulation** menu and **Simulation Parameters**.



Transient Parameters
Stop Time: 1m

Data Output
Start Data Output At:: 0 Secs.
.PRINT Step: 20u Secs.
[✓ Output All Data] Output At .PRINT Step

Monte Carlo and Multi-Step Analysis
☐ Enable Multi-Step
Monte Carlo 10 Steps

☑ Transient
☐ AC
☐ DC Sweep
☐ Noise
☐ TF
☐ DCOP
☑ Options

To set up the analysis, first check the box on the right side of the dialog to select the analysis mode which you wish to perform. You can select more than one mode.

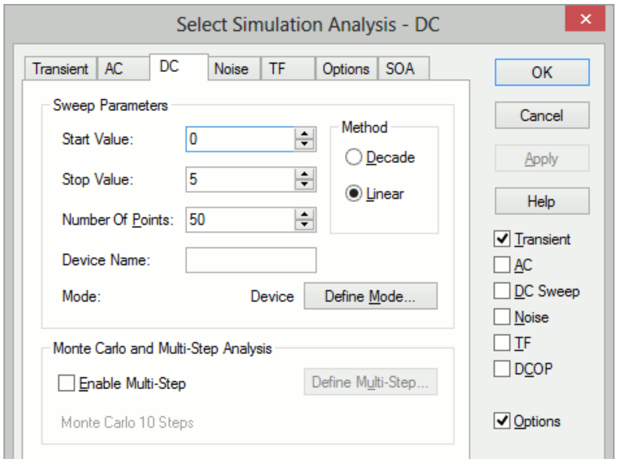
Transient

Select to use transient analysis, the most useful and general mode. First the bias point is found. Then the circuit is simulated over a fixed time interval in steps of varying size according to circuit activity. The circuit may contain any number of time varying voltage and current sources to simulate external signals, test generators etc.
Normally you only need to specify the **Stop Time** specified at the top of the dialog box, but additional control parameters exist.

DC Device Sweep

A DC device sweep will sweep a specified device over a defined range and compute the DC operating point of the circuit at each point. This allows, for example, the DC transfer function of the circuit to be plotted. Note that all reactive elements are ignored in DC sweep.

To set up a DC Sweep, select the **DC Sweep** check box at the right and the DC tab at the top. You will need to enter some values in the Sweep Parameters section:



Sweep Parameters

Start Value: 0

Stop Value: 5

Number Of Points: 50

Device Name:

Mode: Device

Method: ( ) Decade [✓ Linear]

[✓] Transient
[ ] AC
[ ] DC Sweep
[ ] Noise
[ ] TF
[ ] DCOP
[✓] Options

Monte Carlo and Multi-Step Analysis

[ ] Enable Multi-Step

Monte Carlo 10 Steps

The analysis will sweep the device you specify in the **Device** name box over the range specified by Start value, Stop value and Number of points or Points per decade if you select a decade sweep.

The entry in the Device name box is the component reference of the device to be swept and for DC sweep would usually be a voltage source, a current source or a resistor.

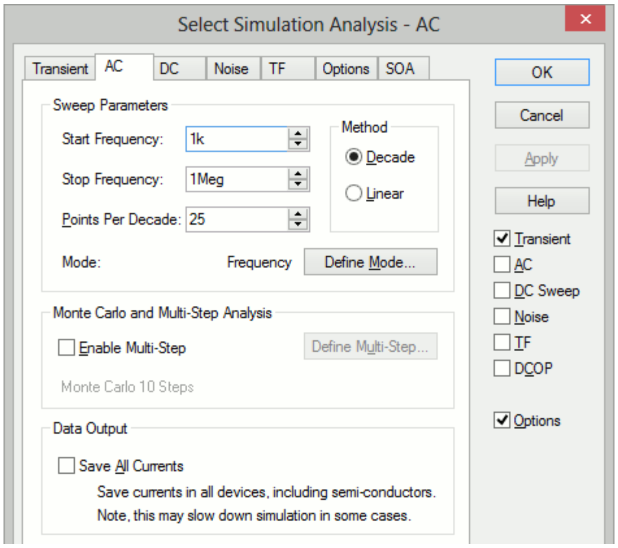
Device sweep is just 1 of 5 modes available with DC sweep. The Define Mode... button allows you to specify one of the others.

AC Frequency Sweep

An AC frequency sweep calculates the small signal response of a circuit to any number of user defined inputs over a given frequency range. The small signal response is computed by treating the circuit as linear about its DC operating point.

There must be at least one input source for AC analysis for the results to be meaningful. Connect a voltage or current source to the circuit, select it then press F7. In the dialog box select the Enable AC check box.

To set up an AC Frequency Sweep, select the **AC** check box at the right and the AC tab at the top. You will need to enter some values in the Sweep Parameters section:


Select Simulation Analysis - AC

Sweep Parameters

Start Frequency: 1k

Stop Frequency: 1Meg

Points Per Decade: 25

Mode: Frequency

Method: [✓ Decade] Linear

Monte Carlo and Multi-Step Analysis

☐ Enable Multi-Step

Monte Carlo 10 Steps

Data Output

☐ Save All Currents

Save currents in all devices, including semi-conductors. Note, this may slow down simulation in some cases.

☑ Transient
☐ AC
☐ DC Sweep
☐ Noise
☐ TF
☐ DCOP
☑ Options

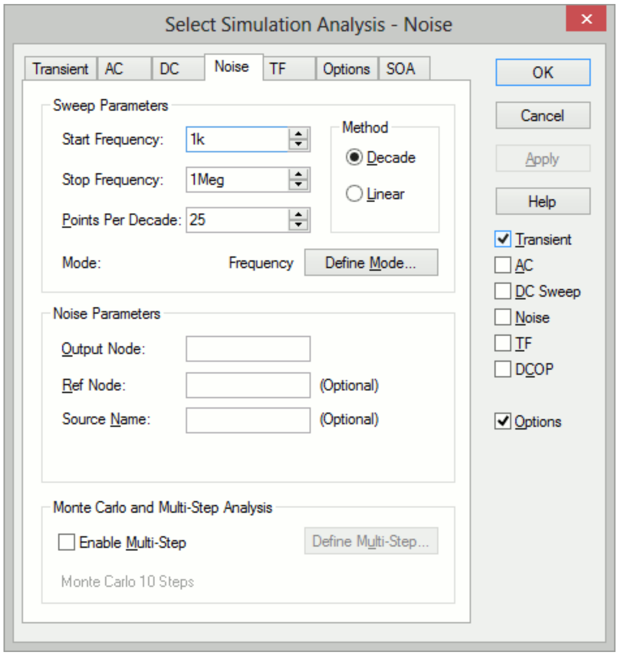
The analysis will sweep the frequency over the range specified by Start frequency, Stop frequency and Number of points or Points per decade if you select a decade sweep. Frequency sweep is just 1 of 6 modes available with AC sweep. The Define Mode... button allows you to specify one of the others.

Noise Analysis

Like AC analysis, AC Noise analysis is a small signal mode. The circuit is treated as linear about its DC operating point and the contribution of all noisy devices to a designated output is computed. The total noise at that output is also calculated and optionally the noise referred back to an input source may also be computed.

First the bias point is found then performs a frequency response analysis over a given frequency range.

Note that it is not necessary to apply an AC specification to any source - including the optional input referred source - as it is with standard SPICE and many (if not all) of its derivatives.



Sweep Parameters
Start Frequency: 1k
Stop Frequency: 1Meg
Points Per Decade: 25
Method: [✓ Decade] Linear
Mode: Frequency

Noise Parameters
Output Node:
Ref Node: (Optional)
Source Name: (Optional)

Monte Carlo and Multi-Step Analysis
☐ Enable Multi-Step
Monte Carlo 10 Steps

☑ Transient
☐ AC
☐ DC Sweep
☐ Noise
☐ TF
☐ DCOP
☑ Options

The analysis will sweep the frequency over the range specified by Start frequency, Stop frequency and Number of points or Points per decade if you select a decade sweep. You will also need to enter some additional Noise Parameters.

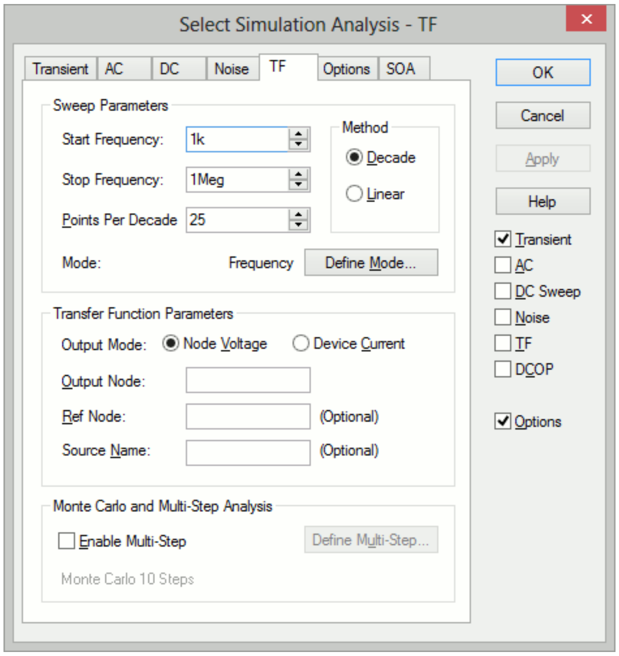
An entry in the Output node box is compulsory. It is the name of the circuit node as it appears in the netlist. We recommend that when running a noise analysis that you assign a user defined net name to your designated output node.

An entry in the Ref node box is optional. It is the node to which the output node is referenced. If omitted it is assumed to be ground. An entry in the Source name box is optional. If specified the noise referred back to it will be calculated. Enter the component reference of the voltage or current source that is used as the input to your circuit.

Frequency sweep is just 1 of 6 modes available with **Noise Analysis**. The **Define Mode...** button allows you to specify one of the others.

TF (Transfer Function)

This analysis is similar to **AC analysis** in that it performs a swept small signal analysis. However, whereas AC analysis calculates the response at any circuit node from a (usually) single input source, **Transfer Function** analysis calculates the individual responses from each source in the circuit to a single specified output node. This allows, for example, the series mode gain, common mode gain and power supply rejection of an amplifier to be measured in one analysis. The same measurements could be performed using AC analysis but several of them would need to be run. Transfer function mode also calculates output impedance or admittance and, if an input source is specified, input impedance.


Select Simulation Analysis - TF

Sweep Parameters

Start Frequency: 1k

Stop Frequency: 1Meg

Points Per Decade 25

Method: [✓ Decade] Linear

Mode: Frequency

Transfer Function Parameters

Output Mode: [✓ Node Voltage] Device Current

Output Node:

Ref Node: (Optional)

Source Name: (Optional)

Monte Carlo and Multi-Step Analysis

☐ Enable Multi-Step

Monte Carlo 10 Steps

☑ Transient
☐ AC
☐ DC Sweep
☐ Noise
☐ TF
☐ DCOP

☑ Options

Transfer Function Parameters

Output Mode - Specify whether the output is a node voltage or device current.

Output Node/Output Source - This is compulsory. If voltage mode is selected it is the name of the circuit node to which the gain of all circuit sources will be calculated. It is the node name as it appears in the netlist. We recommend that when running a transfer function analysis that you assign a user defined name to your designated output node net.

If current mode is selected it is the name of a voltage source through which the output current is measured. The simulation will calculate the gain for every circuit source to this current.

Reference Node - Optional and only available in voltage mode. Output voltage is referred to this node. This is assumed to be ground if it is omitted.

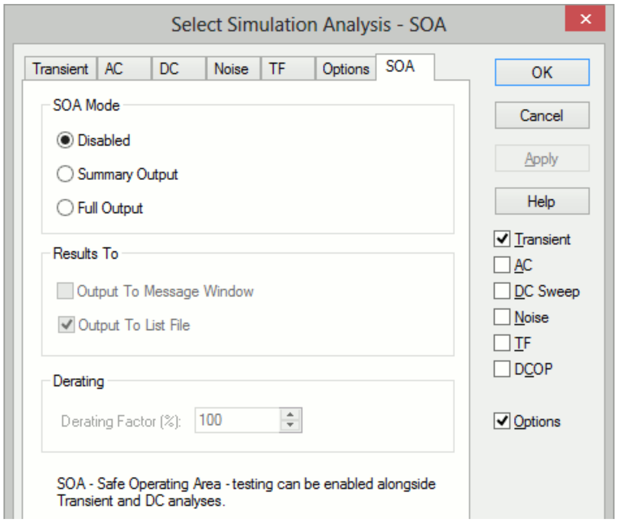
Source Name - Optional. Input impedance to this source device will be calculated if specified.

Monte Carlo and Multi-step Analysis

Transfer Function analysis can be setup to automatically repeat while varying some circuit parameter. See Multi-Step Analyses for details.

SOA (Safe Operating Area) Testing

Safe Operating Area (**SOA**) testing is a feature that can be used with **DC** or **Transient** analyses. With SOA testing, you can set maximum and minimum limits for any simulation quantity and the simulator will display when those limits are violated. It is intended to check that semiconductor devices are operating within the manufacturer's design limits.



SOA Mode
- [✓ Disabled]
- Summary Output
- Full Output

Results To
- Output To Message Window
- [✓] Output To List File

Derating
Derating Factor (%): 100

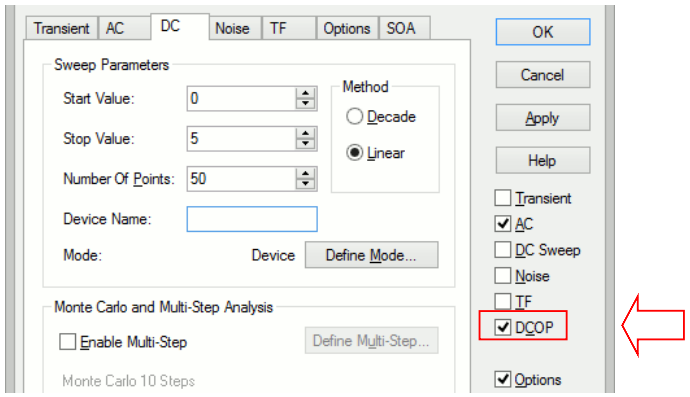
- [✓] Transient
- AC
- DC Sweep
- Noise
- TF
- DCOP
- [✓] Options

SOA - Safe Operating Area - testing can be enabled alongside Transient and DC analyses.

Parts with built-in functions are available to support SOA testing: **Watch - Current**, **Watch - Voltage** and **Watch - Differential Voltage**. These are used to create Watch parts for simple SOA limit testing.

DC Operating Point (DCOP)

To specify a DC operating point analysis, check the **DCOP** box. The DC operating point is calculated automatically for all the other analysis modes described above although for noise analysis the results are not stored. After a DC operating point has been completed, you can annotate your schematic with markers to display the voltages at each node. Select the menu Simulation|Bias Annotation|Auto Place Markers to automatically place markers on all nodes.



Sweep Parameters
Start Value: 0
Stop Value: 5
Number Of Points: 50
Device Name:
Mode: Device
Method: ○ Decade [✓ ● Linear]
Monte Carlo and Multi-Step Analysis
☐ Enable Multi-Step
Monte Carlo 10 Steps
☐ Transient
☑ AC
☐ DC Sweep
☐ Noise
☐ TF
☑ DCOP
☑ Options

Other Analysis Modes:

The following additional modes are available:

Sensitivity	Calculates the sensitivity of a specified output to device and model parameters.
Pole-zero	Calculates the poles and zeros of the circuit.
Multi-step	Analyses Transient, AC, DC, Noise and Transfer Function analyses can be run in an auto-repeat mode while stepping a user-defined parameter.
Monte Carlo Analysis	Part of the multi-step analysis. See section later in this manual for more details.

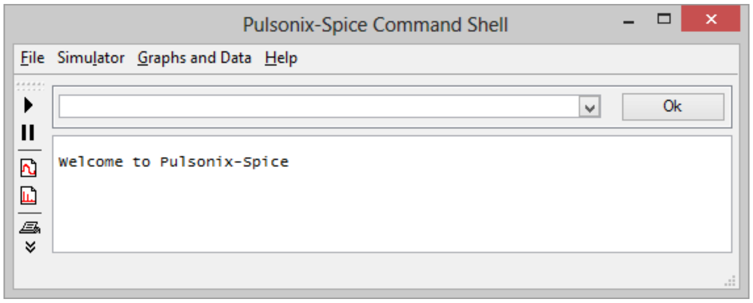
Manual Entry of Simulator Commands

You can enter your own analysis commands of simulation options using the **Extra Simulation Data** option within the Pulsonix Schematic **Simulation** menu.

Running the Simulator

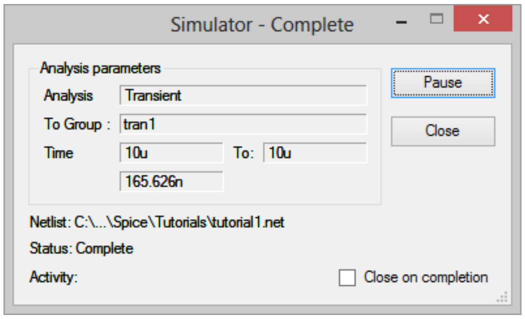
To run the simulator, select the **Simulator** menu and **Simulate Design**, or press **F9**. A report will open showing the status of the simulation. This can be switched off by unchecking the **View SPICE Netlist Report on Simulation** option on the **Simulator Parameters** dialog.

If the Pulsonix Spice simulator is not already running it will be launched at this point which may take a few seconds. A **Pulsonix Spice Command Shell** window will appear which will report any errors found interpreting the SPICE netlist.


Pulsonix-Spice Command Shell
File Simulator Graphs and Data Help
Ok

```
Welcome to Pulsonix-Spice
```

The Command Shell contains information about the simulation being run and its current status. Once the netlist has been successfully read the simulation of the circuit will be performed. A **Simulator Running** dialog will open showing the status of the simulation. From this you can pause the simulation by selecting the **Pause** button. To restart select the **Resume** button (the **Pause** button changes name when simulation pauses) or the Command Shell's **Simulator** and **Resume** menu item.



Analysis parameters
Analysis: Transient
To Group : tran1
Time: 10u To: 10u
165.626n
Netlist: C:\...\Spice\Tutorials\tutorial1.net
Status: Complete
Activity:
Close on completion [ ]

There is no obligation to resume a simulation that has been paused. If you start a new run after having paused the previous one, you will be asked whether you wish to abandon the pending simulation run. The **OK** button will run the current netlist again and does not close the Command Shell window. To **close** this window, click on the small **X** in the top right hand side of the window.

Notes

Using the procedure described above, the simulation results are stored to disc to a file along with the design. These reside in the TEMPDATA directory (The location of this directory is specified in the **File Locations** tab of the **Options Dialog box**.

There is no need to specify in advance of the simulation what voltage or current you wish to look at. By default all top level (i.e. not internal to subcircuits) voltages and currents are stored during the simulation. It is possible to change this, for example to store internal subcircuit signals as well or to restrict further the data stored in order to save disc space. This is done using the .keep control.

It is recommended that any schematics are saved before a run is commenced especially if the run is expected to take a long time.

Plotting Simulation Results

Overview

Pulsonix Spice provides three methods of creating plots of simulated results.

The first approach is to fix voltage or current probes to the schematic before a run. Pulsonix Spice will then generate graphs of the selected voltages and/or currents automatically. The probes have a wide range of options which allow you to specify - for example - how the graphs are organised and when and how often they are updated.

The second approach is to randomly probe the schematic circuit after the run is complete. (You can also do this during a run by pausing first). With this approach, the graph will be created as you point the probe but will not be updated on a new run.

You do not need to make any decisions on how you wish to probe your circuit before starting the run. You can enter a circuit without any fixed probes, run it, then randomly probe afterwards. Alternatively, you can place - say - a single fixed probe on an obvious point of interest, then randomly probe to investigate the detailed behaviour of your circuit.

The third approach is to use the **Define Curve** dialog accessed via the command shell menu **"Graphs and Data|Add Curve..."** menu. This menu opens a dialog box from which you can plot any voltage or current after the simulation is complete. You can also use this feature to plot arithmetic expressions of any number of voltages and/or currents perhaps to plot the power in a device.

The first two approaches are somewhat quicker and simpler so it worth giving some thought to what you wish to view before starting the simulation and then placing probes as appropriate.

Fixed schematic probes are limited to single ended voltages and currents and differential voltages. The second method described above allows you to plot anything you like including device power, FFT's, arbitrary expressions of simulation results and X-Y plots such as Nyquist diagrams. It is possible to set up fixed probes to plot arbitrary expressions of signals but this requires manually entering the underlying simulator command, the .GRAPH control. There is no direct schematic support for this.

Adding Fixed Probes

There are six types of fixed probe:

1. Voltage. Plots the voltage on a net.
2. Current. Plots the current in a device pin.
3. Differential voltage. Plots the voltage difference between two points.
4. dB. Plots the signal in dB (decibels).
5. Phase. Plots the phase of a signal in degrees.
6. Bus. Plots representing all signals on the bus.
7. Keep.

8. Watch.

These are simply schematic components with special attributes. When you add a fixed probe on the schematic, the voltage or current at the point where you place the probe will be plotted each time you run the simulation. The probes have a wide range of options which can be set by selecting the probe and using the shortcut menu to edit the component's Spice parameters value using the **F7** function.

Fixed Voltage Probes

You can place these on a Schematic with the **Insert Fixed Probe** option from the **Simulation** menu and using **Voltage Probe.**

Drop the probe directly onto a connection. Plots voltage on a net.

Fixed Current Probes

You can place these on a Schematic with the **Insert Fixed Probe** option from the **Simulation** menu and using **Current Probe**.

Drop the probe directly over a Component pin. They will have no function if they are not. Plots the current on the device pin.

Fixed Differential Voltage Probes

These can be placed using the **Simulation** menu, **Insert fixed Probe** option and **Differential Voltage.**

Drop the probe into the design and connect its pins to the two nets you wish to probe. Plots the voltage difference between two points.

Fixed dB Probes

These can be placed using the **Simulation** menu, **Insert fixed Probe** option and **Voltage (dB) Probe**.

Drop the probe directly onto a connection. Plots the dB of a signal attached to the probe. Will display the unit of the Y Axis (Yunit) to be decibels (dB).

Fixed Phase Probes

These can be placed using the **Simulation** menu, **Insert fixed Probe** option and **Voltage (Phase) Probe**.

Drop the probe directly onto a connection. Plots the phase of a signal attached to the probe. Will display the unit of the Y Axis (Yunit) to be degrees.

Fixed Bus Probe

These can be placed using the **Simulation** menu, **Insert fixed Probe** option and **Bus Probe**. Drop the probe directly onto a bus. A plot representing all signals on the bus will be created.

Keep Probe

These can be placed using the **Simulation** menu, **Insert fixed Probe** option and **Keep Probe**. Drop the probe directly onto a connection.

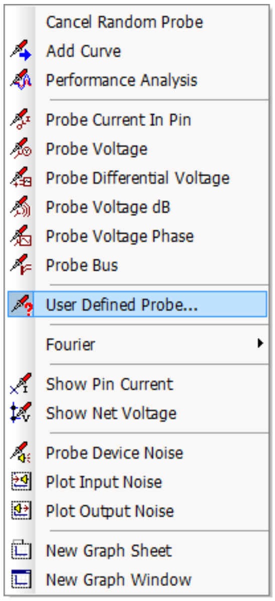
Watch Probe

These can be placed using the **Simulation** menu, **Insert fixed Probe** option and **Watch Probe**. Drop the probe directly onto a connection. This is specifically used during SOA Testing.

Random Probes

Pulsonix Spice has the ability to randomly probe the design using a variety of probe types. Once the simulation has been run, you may select **Random Probe** from the **Simulation** menu.

You can probe, voltage, current, differential voltage, device power, dB, phase, Nyquist diagrams and much more. Once the **Random Probe** mode has been entered, you will see the probe cursor and you can right click and select a probe type from the context menu:



You can also plot arbitrary expressions of any circuit signal. Just a few of the possibilities to get you started are explained below.

Random Voltage Probing

1. Select the **Simulation** menu item **Random Probe**
2. Right click. From the menu, select the **Probe Voltage**.
3. Using the mouse, place the cursor over the net on the circuit you wish to plot. The net under the cursor will be selected and its name presented on the Status Bar.
4. Press the left mouse button. A graph of the voltage at that point will be created. The new curve will be added to any existing graph if the X-axis has the same units. Otherwise, a new graph sheet will be created.

Random Voltage Probing – On New Graph Sheet

1. Select the Simulation menu item Random Probe
2. Right click and select the **New Graph Sheet**. A new graph will be created on subsequent plots.
3. Right click again and select **Probe Voltage**.
4. Press the left mouse button. A graph of the voltage at that point will be created on a new graph sheet.

Random Current Probing

1. Select the **Simulation** menu and select **Random Probe**
2. Right click and select **Probe Current in Pin**
3. Using the mouse, place the cursor over the device pin whose current you wish to plot. The pin will be selected and its name presented on the Status Bar.
4. Press the left mouse button. A graph of the current at that point will be created. The new curve will be added to any existing graph if the X-axis has the same units. Otherwise, a new graph sheet will be created.

Probing db and Phase for AC Analysis

In AC analysis you will probably want to plot signals in dB and you may also want to plot the phase of a signal.

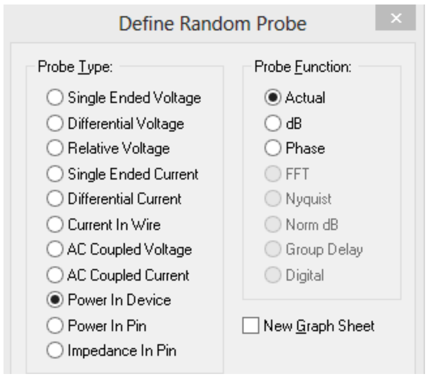
1. Select the **Simulation** menu and **Random Probe**.
2. Select **Probe Voltage dB** for dB or **Probe Voltage Phase** for phase.
3. Using the mouse, place the cursor over the net on the circuit you wish to plot the voltage of.
4. Press the left mouse button. The new curve will be added to any existing graph if the X-axis has the same units. Otherwise, a new graph sheet will be created.

Differential Voltage Probing

1. Select the **Simulation** menu and **Random Probe**.
2. Right click and select **Probe Differential Voltage**.
3. Using the mouse, left click on the two nets you wish to plot the voltage difference between. The first is the "signal node" and the second the "reference node".
4. A graph of the of the voltage difference will be created.

Advanced Probing

From the **Simulation** menu using the **Random Probe** option will allow you to get to more probe functions by using the right mouse shortcut menu. From here you can select **User Defined Probe…**This provides many more probing functions selectable from a dialog:



Probe Type:
- ○ Single Ended Voltage
- ○ Differential Voltage
- ○ Relative Voltage
- ○ Single Ended Current
- ○ Differential Current
- ○ Current In Wire
- ○ AC Coupled Voltage
- ○ AC Coupled Current
- ● Power In Device
- ○ Power In Pin
- ○ Impedance In Pin

Probe Function:
- ● Actual
- ○ dB
- ○ Phase
- ○ FFT
- ○ Nyquist
- ○ Norm dB
- ○ Group Delay
- ○ Digital

☐ New Graph Sheet

Choose the required probe type and function, and whether you require to plot on a new graph sheet. Press OK and you will be prompted on the Status Bar to select the relevant item to create the plot.

More advanced plotting can be achieved with the shortcut option **Add Curve.** These open a dialog box allowing you to enter any expression and which also provides a range of options on how you wish the graph to be plotted.

Saving Simulation Results

Saving the Most Recent Simulation

1. Select **Simulator Command Shell** option, from the **File** menu, select **Data> Save**
2. Type in or select a filename.
3. Select **OK**.

Saving an Earlier Simulation

1. Select **Simulator Command Shell** option, from the **File** menu, select **Data> Sa**ve
2. Select the group (i.e. results of earlier simulation run).
   Groups are named according to the type of analysis appended with a number which is incremented for each new run of the same type. For example, the first transient analysis is called "tran1". AC analyses produce two groups as a DC operating point is carried out first, one will be called "dcn" and the other "opm" where n and m are numbers.
3. Select **OK**.
4. Select **Simulator Command Shell** option, from the **File** menu, select **Data> Cha**nge **Data Group**
5. Type in or select a filename.
6. Select **OK**.

If you wish to continue plotting results from the *latest* simulation run you should restore the current group using the **Graphs and Data** and **Change Data Group**

Restoring Saved Data

1. Select **Simulator Command Shell** option, from the **File** menu, select **Data> Load**
2. Select file to restore.
3. Select **OK.**

Note that all simulation data is saved to a file anyway. The file is treated as temporary and will be deleted or overwritten in subsequent Pulsonix Spice sessions but it is possible to copy or move the file to a non-temporary location as desired. The file will have the same name as the group but with the extension .sxdat and will reside in the TEMPDATA directory. So if the group name is tran2 the file will be tran2.sxdat. For Monte Carlo analysis, the collection name will be used, e.g. mc2.sxdat

Chapter 4. Circuit Definition

Overview

This chapter covers how to define your circuit for simulation. Usually this would be done with the Pulsonix schematic editor which automatically creates a netlist for the simulator. The general operation of the schematic editor is described in the Pulsonix manual. Here we describe how to use the schematic editor to create a design for simulation. Adding circuit stimulus and probes has been dealt with in the *Getting Started* chapter, so this chapter will deal with adding specific types of devices and how particular component values help with this.

There are some who prefer to enter a netlist manually and bypass the schematic editor altogether. This is explained in section *Direct Netlist Entry*.

Using Pulsonix Schematic Entry

In short, the method for using Pulsonix Spice is to use information on parts to indicate what SPICE device they represent, the SPICE model to use during the simulation, and how to output them to a SPICE netlist. These parts can represent circuit stimuli (like voltage and current sources), basic SPICE devices (analog and digital), sub-circuits, special SPICE functional models and Simulator commands (like Probes and Initial Conditions).

You can set up a part’s SPICE information by editing it using the library manager and using the **Edit Spice** button from the **Details** page.

SPICE libraries are provided for use within Pulsonix containing these parts to help you quickly generate circuits to simulate. These libraries can also be used as a guideline to be referenced when creating your own libraries. The parts can be found in the **Spice** parts libraries, and it’s corresponding symbol library. **Spice** contains special parts for functional modeling, circuit stimuli, probes along with hundreds of parts that reference the Pulsonix Spice Simulator’s models and subcircuits.

Note: All components in these libraries are Schematics only. Add your own PCB symbols to them if you wish to translate the simulated circuit to a PCB design.

Add the required parts from these libraries to your schematic and connect up to create the circuit to be simulated. Insert circuit stimuli by adding voltage and/or current source parts to the circuit. Add fixed voltage and current probe components to the nets and pins that you wish to plot a simulation curve. Then use the **Simulation Parameters** option from the **Simulation** menu to choose the analysis mode and simulation options.

The **Simulate Design** option can be found on the **Simulation** menu. All pages of the schematic will be collated and the entire schematic simulated. To simulate only parts of a schematic design, use the **Simulate Current Page** and **Simulate Selected Items** options available from the Simulation toolbar.

Also you can add special components to the circuit to represent initial conditions for a device prior to simulation. The "Initial Condition" and "Nodeset" components are described later.

Analog Semiconductor Devices

This includes all analog semiconductor components both discrete devices and integrated circuits.

The **SpiceModels** library has a number of parts representing analog devices such as transistors and opamps. Each part has a default model name, but you are not restricted to just that model. Pulsonix Spice is supplied with about 6500 electrical models that can be used with these parts.

If you know the model name, then use **Insert Part Using Model** from the **Simulation** menu to add these parts.

If you know it is an Opamp, for example, but don't know the model name then use the **Part Browser** from the **View** menu to add an Opamp part and then use <**F7**> on it to see which models are available for this part.

If the model has a different pin order to the part then you have two options to sort this out:-

1. Create a specific part by copying the part using the library manager. Edit the new part using the **Pin Properties** button on the **Define Spice Type** dialog to change the spice pin order on the part to match the model.
2. Keep using the generic Opamp part and use the Simulator's **Associate Model/Part** option to place a pin mapping on the model to match the part.

Digital Devices

All digital devices supported are defined in the Digital Simulation Manual. These are mainly XSPICE devices, but many digital devices are modelled using subcircuits in the digital model libraries.

In addition to the digital models, many of the Pulsonix library 74HCxxx and 74LSxxx components can have simple mapping subcircuits defined for them in model library Pulsonix.lb. These subcircuits have been added to map these Pulsonix parts to their corresponding Pulsonix Spice models.

For example, component SN74LS00.

```
.subckt SN74LS00 In1 In2 Out Vcc Gnd

A1 [ In1 In2 ] Out LS00

.ends
```

This subcircuit has the same name as the part. If you don't define the spice type of a part, its part name is used instead to reference the Pulsonix Spice subcircuit. The part has Vcc and Gnd pins that the LS00 digital model does not have, so this mapping subcircuit defines them but does not pass them on to the model.

So by adding these simple mapping subcircuits with a text editor you don't need to change your Pulsonix libraries. Note: this will only work if the part name is different from the actual model name. For example, many parts in the 4000 series have the same names as their models.

Note, that the spice information for the part in this example will be:

Spice Device Type = ”Subcircuit”

Spice Model = ”SN74LS00”

Built-in Digital Components

There are a number of built-in digital devices that have special parts in the **spice.pal** library. These parts are preset to use XSPICE models and many of them have specific edit dialogs for easily setting up their device parameters. Add these parts to a circuit using the **Digital** popup toolbar (found on the Parts toolbar), select them and press <**F7**> to edit their parameters.

These include:

Digital Capacitor

Digital Delay

Digital Frequency Divider

Digital Ground

Digital Initial Condition

Digital Pull-Down

Digital Pull-Up

Digital Pulse

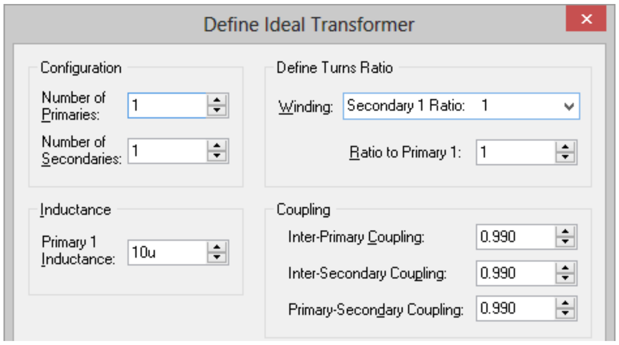
Digital Resistor

Digital VCC

Ideal Transformers

To insert an ideal transformer, use one of the following methods:

1. Using the **Part Browser** view, from the **Passives** category drag part **Ideal Transformer** into the design.
2. Using the **Parts toolbar**, select the **Magnetics** popup button, and from the **Magnetics** popup toolbar click on the **Ideal Transformer** icon.
   This will open the following dialog:



Number of Primaries: 1
Number of Secondaries: 1
Winding: Secondary 1 Ratio: 1
Ratio to Primary 1: 1
Primary 1 Inductance: 10u
Inter-Primary Coupling: 0.990
Inter-Secondary Coupling: 0.990
Primary-Secondary Coupling: 0.990

Configuration		Specify the number of primaries and secondaries. You can specify up to ten of each.
Define Turns Ratio		
	Select Winding	Lists all windings except primary 1.
	Ratio to Primary 1	Enter the turns ratio wrt primary 1.
Inductance		
	Primary 1 Inductance	Self-explanatory
Coupling		
	Inter-primary coupling	Coupling factor between primaries.
	Inter secondary coupling	Coupling factor between secondaries
	Primary-secondary coupling	Coupling factor from each primary to each secondary

This method of implementing an ideal transformer is not totally general purpose as you cannot arbitrarily define inter winding coupling factors. If you need a configuration not supported by the above method, you can define any ideal transformer using ideal inductors and the "Mutual Inductance" device.

Saturable Magnetic Components

Pulsonix Spice is supplied with a number of models for inductors and transformers that correctly model saturation and, for most models, hysteresis. As these components are nearly always custom designed there is no catalogue of manufacturers parts as there is for semiconductor components. Consequently a little more information is needed to specify one of these devices. This section describes the facilities available and a description of the models available.

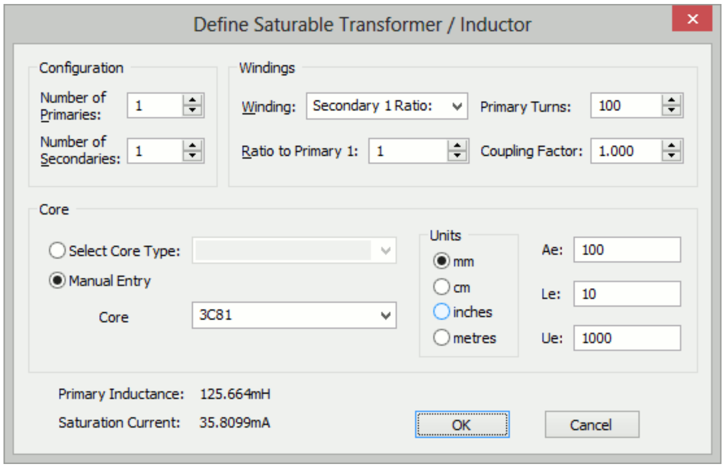
Core Materials

The available models cover a range of ferrite and MPP core materials for inductors and transformers with any number of windings. The complete simulation model based on a library core model is generated by the user interface according to the winding specification entered.

Placing and Specifying Components

To insert a saturable device use one of the following methods:

1. Using the **Part Browser** view, from the **Passives** category drag part **Saturable Transformer (1P 1S)** into the design.
2. Or, using the **Parts** toolbar, select the **Magnetics** popup button, and from the popup toolbar click on the **SaturableTransformer (1P 1S)** icon. Place the component into the design.
3. The following dialog will be displayed.


Define Saturable Transformer / Inductor

Configuration
Number of Primaries: 1
Number of Secondaries: 1

Windings
Winding: Secondary 1 Ratio:
Primary Turns: 100
Ratio to Primary 1: 1
Coupling Factor: 1.000

Core
Select Core Type:
Manual Entry
Core 3C81

Units
mm
cm
inches
metres

Ae: 100
Le: 10
Ue: 1000

Primary Inductance: 125.664mH
Saturation Current: 35.8099mA

4. Specify the number of windings required for primary and secondary in the Configuration section. If you just want a single inductor, set primary turns to 1 and secondaries to 0.

5. Specify turns rations in the **Define Windings** section. You can select the winding to define using the **Select Winding** drop down box then enter the required ratio to primary 1 in the edit box below it.

6. Specify the number of turns for the primary and coupling factor. The coupling factor is the same for all windings. You can define different coupling factors for each winding by adding ideal inductors in series with one or more windings. In some instances it may be necessary to add coupled inductors in series. This is explained in more detail in “Coupling Factor.

7. Specify the core characteristics in the **Define Core** section. A number of standard core sets are pre-programmed and can be selected from the **Select Core Type** list at the top. If the part you wish to use is not in the list or if you wish to use a variant with a – say – different air gap, you can manually enter the characteristics by clicking on the **Manual Entry** check box.

The values you need to enter are

Ae	Effective Area
Le	Effective Length
Ue	Relative Permeability
Core Material	

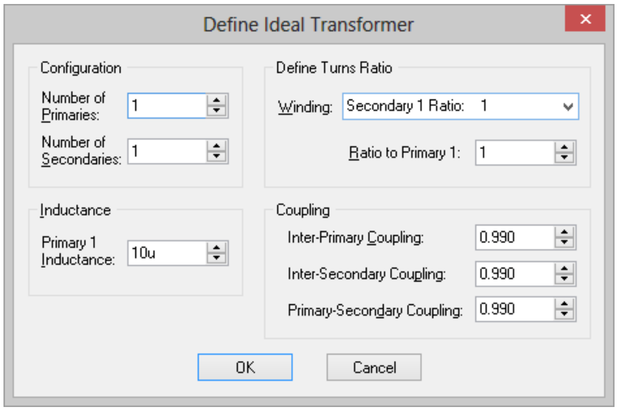
Model Details

The models for saturable components can be found in the file cores.1b. Most of the models are based on the Jiles-Atherton magnetic model which includes hysteresis effects. The MPP models use a simple model which does not include hysteresis. These models only define a single inductor. To derive a transformer model, the user interface generates a subcircuit model that constructs a non-magnetic transformer using controlled sources. The inductive element is added to the core which then gives the model its inductive characteristics.

The model does not currently handle other core characteristics such as eddy current losses nor does it handle winding artefacts such as resistive losses, skin effect, inter-winding capacitance or proximity effect.

Ideal Transformers

To define an ideal transformer, use the **Part Browser**, from the **Passives** category drag part **Ideal Transformer** into the design, this will open the following dialog box:



Configuration

Specify the number of primaries and secondaries. You can specify up to ten of each.

Define turns ratio

Select Winding	Lists all windings except primary 1.
Radio to Primary 1	Enter the turns ratio with respect ratio primary 1.

Inductance and coupling

Primary 1 Inductance	Self-explanatory
Inter-primary coupling	Coupling factor between primaries
Inter secondary coupling	Coupling factor between secondaries
Primary-secondary coupling	Coupling factor from each primary to each secondary

This method of implementing an ideal transformer is not totally general purpose as you cannot arbitrarily define inter winding coupling factors. If you need a configuration not supported by the above method, you can define any ideal transformer using ideal inductors and the Mutual Inductance device.

Coupling Factor

The standard user interface for both saturable and ideal transformers provide only limited flexibility to specify inter-winding coupling factor. In the majority of applications, coupling factor is not an important issue and so the standard model will suffice.

In some applications, however, the relative coupling factors of different windings can be important. An example is in a flyback switched mode supply where the output voltage is sensed by an auxiliary winding. In this instance, best performance is achieved if the sense winding is strongly coupled to the secondary. Such a transformer is likely to have a different coupling factor for the various windings.

You can use external leakage inductances to model coupling factor and this will provide some additional flexibility. One approach is to set the user interface coupling factor to unity and model all non-ideal coupling using external inductors. In some cases it may be necessary to couple the leakage inductors. Consider for example an E-core with 4 windings, one on each outer leg and two on the inner leg. Each winding would have the same leakage inductance. But the two windings on the centre leg would be more closely coupled to each other than to the other windings. To model this, the leakage inductances for the centre windings could be coupled to each other using the mutual inductor method described in the next section.

Mutual Inductors

You can specify coupling between any number of ideal inductors, using the mutual inductor device. There is no menu or schematic symbol for this. It is defined by a line of text that must be added to the netlist. (See “Manual Entry of Simulator Commands”) The format for the mutual inductance line is:

`Kxxxx` ***`inductor_1 inductor_2 coupling_factor`***

Where:

inductor_1	Component reference of the first inductor to be coupled
inductor_2	Component reference of the second inductor to be coupled
coupling_factor	Value between 0 and 1 which defines strength of coupling

Note

If more than 2 inductors are to be coupled, there must be a K device to define every possible pair.

Examples

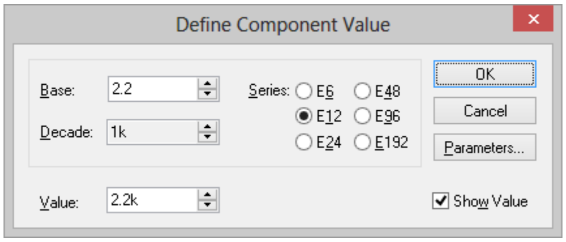
```
** Couple L1 and L2 together
K12 L1 L2 0.98
** Couple L1, L2 and L3
K12 L1 L2 0.98
K23 L2 L3 0.98
K13 L1 L3 0.98
```

Resistors, Capacitors and Inductors

Resistors

To add a resistor, use the **Part Browser**, from the **Passives** category drag part **Resistor (Box Shape)** [or your preferred shaped resistor] into the design.

To edit value use <**F7**>. This will display the following dialog for resistors.



Base: 2.2
Decade: 1k
Series: E6 [✓ E12] E24 E48 E96 E192
Value: 2.2k
[✓] Show Value

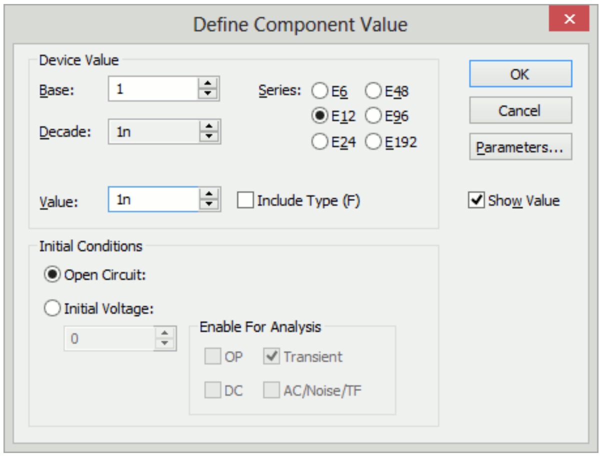
You can enter the value directly in the **Result** box or use the **Base** and **Decade** up/down controls.

Additional Parameters

Press **Parameters**… button to edit additional parameter associated with the device such as temperature coefficients (TC1, TC2). Refer to device in the *Simulator Reference Manual* for details of all device parameters.

Capacitors and Inductors

The following dialog will be displayed when you edit a capacitor or inductor using <**F7**>:



Device Value
Base: 1
Decade: 1n
Series: E6 [✓ E12] E24 E48 E96 E192
Value: 1n
[ ] Include Type (F)
[✓] Show Value

Initial Conditions
[✓ Open Circuit:] Initial Voltage: 0

Enable For Analysis
[ ] OP [✓] Transient
[ ] DC [ ] AC/Noise/TF

The device value is edited in the same manner as for resistors. You can also supply an initial condition which defines how the device behaves while a DC operating point is calculated. For capacitors you can either specify that the device is open circuit or alternatively you can specify a fixed voltage. For inductors, the device can be treated as a short circuit or you can define a constant current.

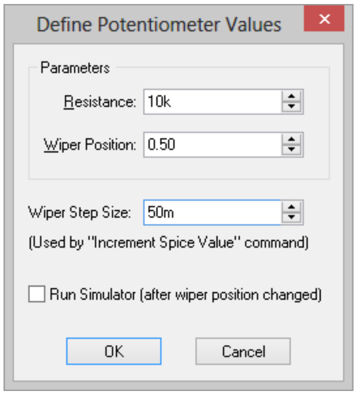
Important note to experienced Pulsonix Spice users
The initial condition values above do not require the ‘UIC (or Skip DC bias point) option to be set. This implementation of initial condition is a new feature not found in standard Pulsonix Spice. If an initial condition for a capacitor is defined, it will behave like a voltage source during the DC operating point calculation. Similarly an inductor will behave like a current source if it has an initial condition defined.

Potentiometer

A potentiometer device is supplied with Pulsonix and can be inserted by using one of the following methods:

1. Using the **Part Browser** view, from the **Passives** category drag part **Potentiometer** into the design.
2. Using the **Parts** toolbar, select the **Passives** popup button, and select **Potentiometer**.

This can be edited in the usual manner with the **<F7>** key to change its wiper position and resistance.


Define Potentiometer Values

Parameters

Resistance: 10k

Wiper Position: 0.50

Wiper Step Size: 50m

(Used by "Increment Spice Value" command)

☐ Run Simulator (after wiper position changed)

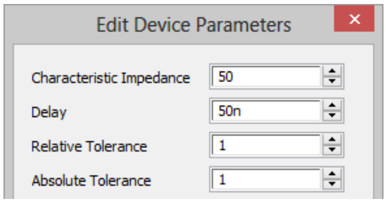
Enter **Resistance** and **Wiper position** as required.

Check **Run simulation** after position change if you wish a new simulation to be run immediately after the wiper position changes.

The potentiometer’s wiper position may also be altered using the shift-up and shift-down keys while the device is selected. Edit Inc/dec step size to alter the step size used for this feature.

Lossless Transmission Line

To add this component, use the **Part Browser**, from the **Passives** category drag part **Transmission Line (Lossless)** into the design. Pressing <**F7**> will display the following dialog:



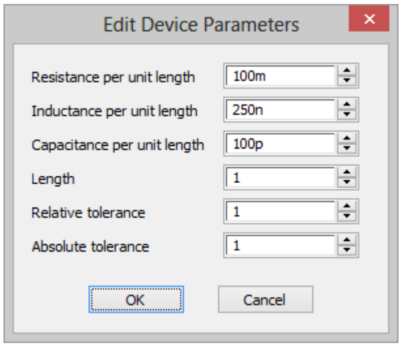
Edit Device Parameters

Characteristic Impedance	50
Delay	50n
Relative Tolerance	1
Absolute Tolerance	1

Enter the **Characteristic Impedance** (Z0) and **Delay** as indicated.

Lossy Transmission Line

To add this component, use the **Part Browser**, from the **Passives** category drag part **Transmission Line (Lossy RLC)** into the design. Pressing <**F7**> will display the following dialog:



Edit Device Parameters

Resistance per unit length	100m
Inductance per unit length	250n
Capacitance per unit length	100p
Length	1
Relative tolerance	1
Absolute tolerance	1

Lossy lines must be defined in terms of their per unit length impedance characteristics. Currently only series losses are supported.

Enter parameters as indicated. The absolute tolerance and relative tolerance parameters control the accuracy/speed trade-off for the model. Reduce these values for greater accuracy.

Fixed Voltage and Current Sources

See “Circuit Stimulus”

Controlled Sources

There are four types which can be found using the **Part Browser** and the **Sources** category:

Voltage controlled voltage source or VCVS

Voltage controlled current source or VCCS

Current controlled voltage source or CCVS

Current controlled current source or CCCS

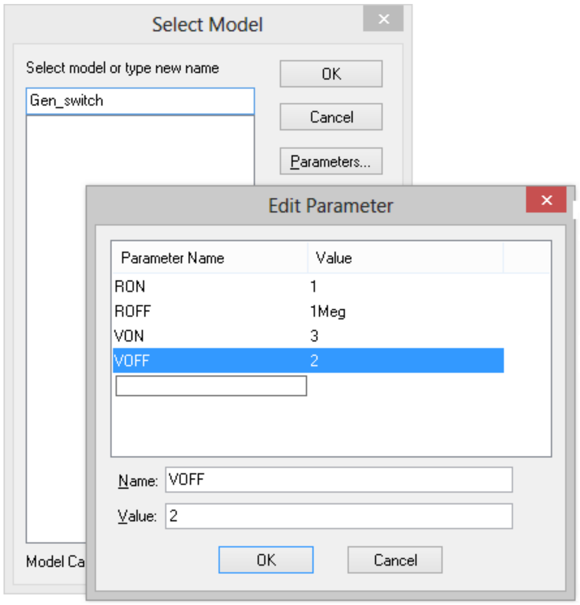
These have a variety of uses. A VCVS can implement an ideal Opamp; current controlled devices can monitor current; voltage controlled devices can convert a differential signal to single ended.

They require just one value to define them which is their gain. Edit value using <**F7**> and you will be presented with a dialog similar to that used for resistors, capacitors and inductors but without the **Parameters**… button

Voltage Controlled Switch

This is essentially a voltage controlled resistor with two terminals for the resistance and two control terminals.

To add this component, use the **Part Browser**, from the **Analog Functions** category, from the **Switches**, drag part **Voltage Controlled Switch** into the design. Pressing <**F7**> will display the following dialog:



Gen_switch

Parameter Name	Value
RON	1
ROFF	1Meg
VON	3
VOFF	2

Name: VOFF

Value: 2

```
If On Voltage > Off Voltage
    If control voltage > On Voltage
        Resistance += On Resistance
    else if control voltage < Off Voltage
        Resistance = Off Resistance

If Off Voltage > On Voltage
    If control voltage > Off Voltage
        Resistance = Off Resistance
```

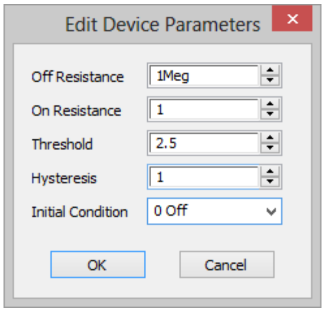
```
else if control voltage < On Voltage
    Resistance = On Resistance
```

If the control voltage lies between the On Voltage and Off Voltage the resistance will be somewhere between the on and off resistances using a law that assures a smooth transition between the on and off states.

Switch with Hysteresis

An alternative switch device is available which abruptly switches between states rather than following a continuous V-I characteristic. The switching thresholds are governed by an hysteresis law and, when used with the simulator, the state change is controlled to occur over a fixed time period (currently 10nS).

To add this component, use the **Part Browser**, from the **Analog** category **Switches,** drag part **Switch with Hysteresis** into the design. Pressing <**F7**> will display the following dialog:


Edit Device Parameters

Off Resistance	1Meg
On Resistance	1
Threshold	2.5
Hysteresis	1
Initial Condition	0 Off

Simulation Command Devices

These parts are included in the schematic design to give information to the Simulator.

Initial Conditions

An "Initial Condition" is a special device that fixes the voltage at the net to which it is connected while the DC bias point is calculated. When a subsequent transient analysis is started, it is released.

An Initial Condition device is supplied with Pulsonix and can be inserted by using one of the following methods:-

1. Using the **Part Browser** view, from the **Miscellaneous** category drag part **Initial Condition** into the design and drop onto a connection.
2. Using the **Parts** toolbar, select the **Miscellaneous** popup button, and from the **Miscellaneous** popup toolbar click on the **Initial Condition** icon.

Note that initial conditions are applied with a driving resistance of 1Ω but this can be changed.

Nodesets

The "Nodeset" is a special device which is used to influence the solution of the DC bias point. Nodesets have two uses:

1. To force a particular solution in circuits that have more than one stable state.
2. To help difficult to converge circuits. With Pulsonix Spice it is rare that nodesets will be needed for this purpose. For more information on the use of nodesets to aid convergence see "Convergence".

A Nodeset device is supplied with Pulsonix and can be inserted by using one of the following methods:-

1. Using the **Part Browser** view, from the **Miscellaneous** category drag part **Nodeset** into the design and drop onto a connection.
2. Using the **Parts** toolbar, select the **Miscellaneous** popup button, and from the **Miscellaneous** popup toolbar click on the **Nodeset** icon..

Keep Voltage/Current

"Keep Voltage" and “Keep Current” are a special devices that instruct the simulator what values to store during simulation.

These devices are supplied with Pulsonix and can be inserted by using the **Part Browser** view. From the **Miscellaneous** category drag part **Keep Voltage** or **Keep Current** into the design. Drop **Keep Current** onto a device pin, and **Keep Voltage** onto a connection.

Functional Blocks

Functional modelling is the activity of defining a device in terms of its function or behaviour rather than specifying a device part number for which a model exists. Functional Blocks have a number of uses. Here are two examples:

1. System level simulation. You are investigating the viability or characteristics of a complete system before actually considering its implementation detail. Your system may consist of a number of interconnected blocks each with an easily defined function.
2. Device model implementation. Functional models can be used to actually create device models. Suppose you wish to use an op-amp for which no model is available. The only characteristic that affects the performance of your circuit is its gain bandwidth product. So instead of creating a detailed model you simply use a differential voltage amplifier with the appropriate GBW.

Pulsonix Spice provides functional modelling at both the Schematic and Simulator levels. The Schematic provides a convenient user interface to the functional devices provided by the Simulator. This is accessed via the **Analog Behavioural** popup toolbar (on the **Parts** toolbar), or using the **Analog/Behavioural** category in the **Part Browser** view.

The simulator provides three devices that can be defined in a completely arbitrary manner.

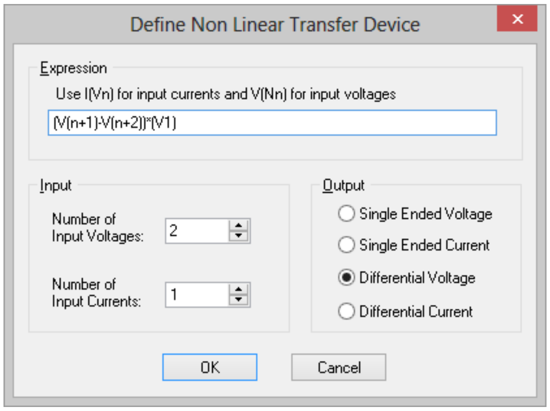
Device	**Description**
Arbitrary non-linear source or “B” device	Analog non-linear device. Can express single voltage or current in terms of any number of circuit voltages and currents including its own output.
S-domain Transfer Function	Linear block with single input and output each of which may be single ended or differential, voltage or current. Specified in terms of its S-domain or Laplace transfer function
Arbitrary Logic Block	Digital device. Implements any digital device, combinational, synchronous or asynchronous using a descriptive language.

Pulsonix functional devices currently provided are:

Menu	**Description**
Non-linear Transfer Function...	Based on the arbitrary non-linear source. This will create a Schematic symbol with your specified inputs and outputs. You enter the equation to relate them.
Laplace Transfer Function...	Based on the S-domain transfer function block. This will create a Schematic symbol with specified input and output. You enter an s-domain transfer function.
Arbitrary Non-linear	Non-linear passive components based on non-linear source. You enter component's value in terms of its voltage (R and C) or current (L).

Non-Linear Transfer Function

To add this component, use the **Part Browser**, from the **Analog** and **Behavioural** category drag part **Non-Linear Transfer** into the design. Pressing <**F7**> will display the following dialog:

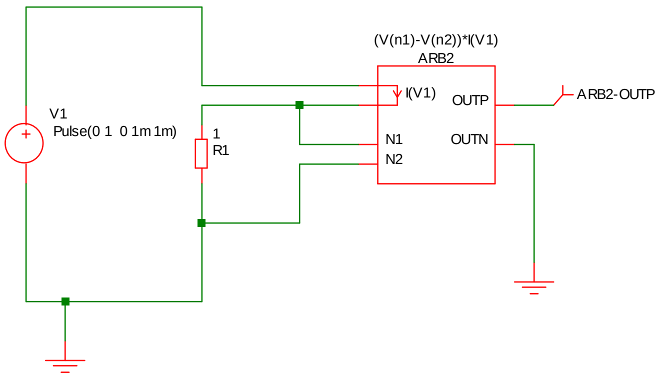

Define Non Linear Transfer Device

Expression
Use I(Vn) for input currents and V(Nn) for input voltages
(V(n+1)-V(n+2))*(V1)

Input
Number of Input Voltages: 2
Number of Input Currents: 1

Output
( ) Single Ended Voltage
( ) Single Ended Current
(●) Differential Voltage
( ) Differential Current

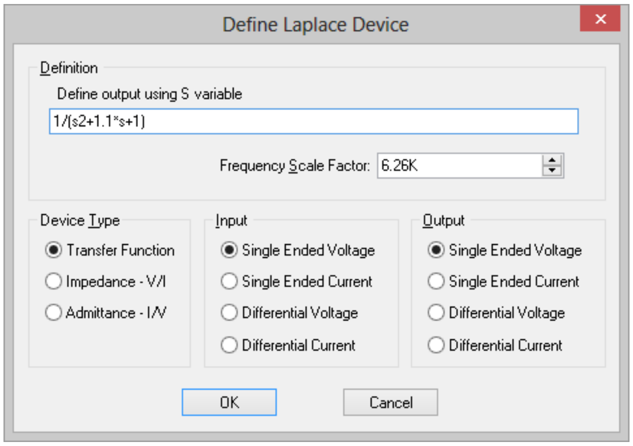
Specify the number of input voltages and currents you require. All voltage inputs are single-ended and all input currents are differential. In the expression box, you must specify an equation relating the output to the inputs. In the equation, currents are referred to by the label "I(Vn)" where n identifies the actual current input. 1 is the first (top most on the symbol), 2 is the second and so on. Voltages are referred to by the label "V(Nn)". Again n identifies the actual voltage input in the same manner as for current. In the example above, the expression shown - (V(n1)-V(n2))*I(V1) - multiplies a voltage and current together. This could be used to monitor the power in a two terminal device as shown in the following Schematic:



In the above, ARB2 is the device created from the Non-linear Transfer Function menu. ARB2-OUTP will carry a voltage equal to the power dissipation in R1.

Laplace Transfer Function

To add this component, use the **Part Browser**, from the **Analog** and **Behavioural** category drag part **Laplace Transfer** into the design. Pressing <**F7**> will display the following dialog:


Define Laplace Device

Definition

Define output using S variable: 1/(s2+1.1*s+1)

Frequency Scale Factor: 6.26K

Device Type: [● Transfer Function] ○ Impedance - V/I ○ Admittance - I/V

Input: [● Single Ended Voltage] ○ Single Ended Current ○ Differential Voltage ○ Differential Current

Output: [● Single Ended Voltage] ○ Single Ended Current ○ Differential Voltage ○ Differential Current

The operation of the various controls is described below.

Definition	Enter an expression using the 'S' variable to define the frequency domain transfer function. The above shows the example of a second order response.
Frequency scale factor	Multiplier for frequency
Device type:	
Transfer function	Expression defines output/input

Impedance V/I	Two terminal device, expression defines voltage/current
Admittance I/V	Two terminal device, expression defines current/voltage
Input	Input configuration for transfer function
Output	Output configuration for transfer function

When you close the box, an appropriate part will be used according to the selections you make for device type, input and output.

Laplace Expression

When you close the box, a symbol will be created according to the selections you make for device type, input and output.

As seen in the above examples, the transfer function of the device is defined by the model parameter LAPLACE. This is a text string and must be enclosed in double quotation marks. This may be any arithmetic expression containing the following elements:

Operators

+-*/^
^ means raise to power. Only integral powers may be specified.

Constants

Any decimal number following normal rules. Spice style engineering suffixes are accepted.

S Variable

This can be raised to a power with '^' or by simply placing a constant directly after it (with no spaces). E.G. s^2 is the same as s2.

Filter response functions

These are described in the following table:

Function Syntax	**Filter Response**
BesselLP(*order, cut-off)*	Bessel low-pass
BesselHP(*order, cut-off)*	Bessel high-pass
ButterworthLP(*order, cut-off)*	Butterworth low-pass
ButterworthHP(*order, cut-off)*	Butterworth high-pass
ChebyshevLP(*order, cut-off, passband_ripple)*	Chebyshev low-pass
ChebyshevHP(*order, cut-off, passband_ripple)*	Chebyshev high-pass

Where:

Order	Integer specifying order of filter. There is no maximum limit but in practice orders larger than about 50 tend to give accuracy problems.

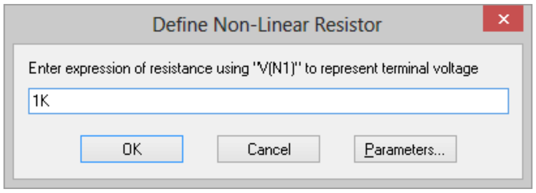
Cut-off	-3dB Frequency in Hertz
Passband_ripple	Chebyshev only. Passband ripple spec. in dB

Arbitrary Non-linear Passives

Non-linear Resistor

Insert this into the design by selecting the device named **Resistor (Non-Linear)** from the **Functions** popup toolbar or from the **Passives** category in the **Part Browser**.

Once added to the design, select the device and click **F7** to edit or modify the Resistor's value. The following dialog is displayed:


Define Non-Linear Resistor

Enter expression of resistance using "V(N1)" to represent terminal voltage

1K

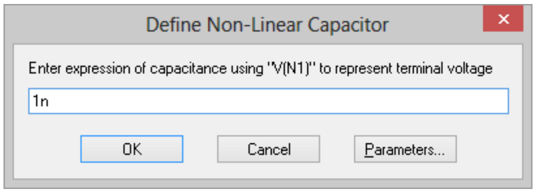
Resistance is expressed in terms of V(n1) which represents the terminal voltage. For example the following defines a resistor whose resistance is 1k * (1 + v2/10)

1K*(1+v(n1)^2/10)

Non-linear Capacitor

Insert this into the design by selecting the device named **Capacitor (Non-Linear)** from the **Functions** popup toolbar or from the **Passives** category in the **Part Browser**.

Once added to the design, select the device and click **F7** to edit or modify the Capacitor's value. The following dialog is displayed:


Define Non-Linear Capacitor

Enter expression of capacitance using "V(N1)" to represent terminal voltage

1n

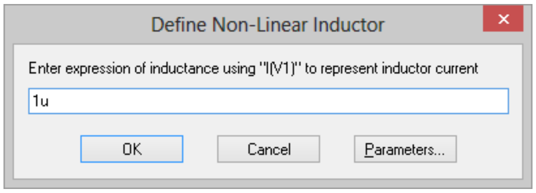
Capacitance is expressed in terms of V(n1) which represents the terminal voltage. For example the following defines a capacitor whose capacitance is 1n * (1e-3 * v2 + 1):

1n*(v(n1)^2*1e-3+1)

Non-linear Inductors

Insert this into the design by selecting the device named **Inductor (Non-Linear)** from the **Functions** popup toolbar or from the **Passives** category in the **Part Browser**.

Once added to the design, select the device and click **F7** to edit or modify the Inductor's value. The following dialog is displayed:


Define Non-Linear Inductor

Enter expression of inductance using "I(V1)" to represent inductor current

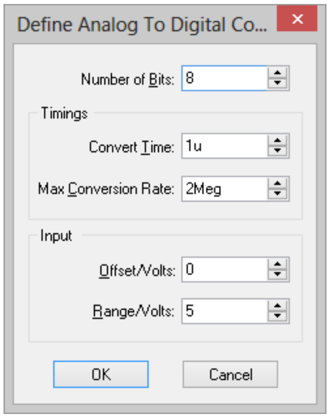
1u

Value L should be set to inductance expressed in terms of I(V1) which represents the current in the device. For example the following defines an inductor whose inductance is 1u / (1 + i4):

L = 1u/(1+I(v1)^4)

Generic ADC's and DAC's

To add these components use the **Part Browser**, from the **Digital** category **Generic,** drag part **A-D Converter** or **D-A Converter** into the design. Pressing <**F7**> will display the following dialog:


Define Analog To Digital Co...

Number of Bits: 8

Timings

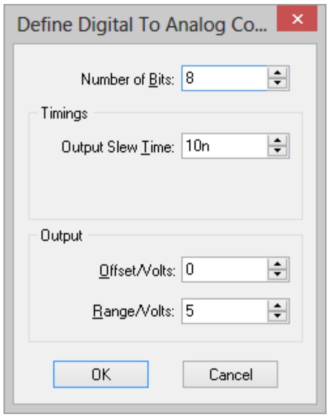
Convert Time: 1u

Max Conversion Rate: 2Meg

Input

Offset/Volts: 0

Range/Volts: 5


Define Digital To Analog Co...

Number of Bits: 8

Timings

Output Slew Time: 10n

Output

Offset/Volts: 0

Range/Volts: 5

The above dialogs are opened by selecting the device named **A-D Converter** or **D-A Converter** from the **Digital Generic** popup toolbar or from the **Digital/Generic** category in the **Part Browser**.

These allow the specification of generic data conversion devices. They are implemented using the simulator's ADC and DAC models. For details refer to the *Digital Simulation Manual*.

The controls in these boxes are explained below.

Number of bits	Resolution of converter. Values from 1 to32
Convert time (ADC)	Time from start convert active (rising edge) to data becoming available
Max conversion rate (ADC)	Max frequency of start convert. Period (1/f) must be less than or equal to convert time.

Output slew time (DAC)	Whenever the input code changes, the output is set on a trajectory to reach the target value in the time specified by this value.
Offset voltage	Self-explanatory
Range	Full scale range in volts

Generic Digital Devices

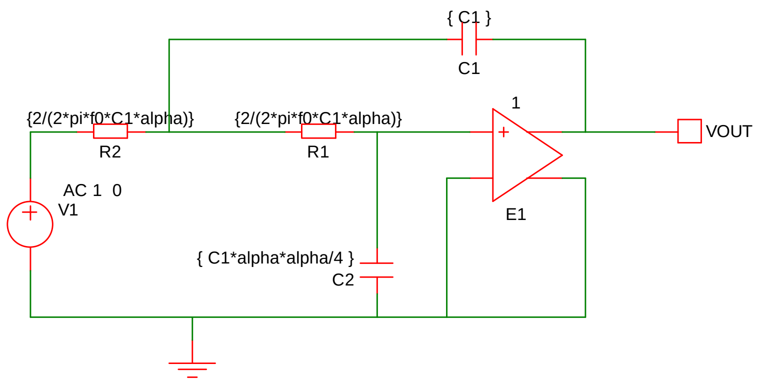
A number of generic digital devices are provided on the **Digital Generic** popup toolbar and on the **Digital/Generic** category in the **Part Browser**.

Each will automatically insert a Part using a basic spec provided by your entries to a dialog box. Functions provided are, counter, shift register, AND, OR, NAND and NOR gates, and bus register.

Parameters and Expressions

You can specify both device and model parameters using an arithmetic expression containing user defined variables. The variables may be defined using the .PARAM simulator control, which must be placed in the netlist, or globally in a script using the Let command. A variable may also be swept using the parameter sweep mode for the swept analyses and stepped for multi-step analyses. Complete documentation on this subject can be found in the "Simulator Devices" chapter of the *Simulator Reference Manual*. Below are brief details of how to use expressions with a schematic based design. We explain this with an example.

Example



The above circuit is that of a two pole low-pass filter. C1 is fixed and R1=R2. The design equations are:

R1=R2=2/(2*pi*f0*C1*alpha)

C2=C1*alpha*alpha/4

where freq is the cut off frequency and alpha is the damping factor.

The complete netlist for the above circuit is:

V1 V1_P 0 AC 1 0

C2 0 R1_P {C1*alpha*alpha/4}

C1 VOUT R1_N {C1}

E1 VOUT 0 R1_P 0 1

R1 R1_P R1_N {2/(2*pi*f0*C1*alpha)}

R2 R1_N V1_P {2/(2*pi*f0*C1*alpha)}

Before running the above circuit you must assign values to the variables. This can be done by one of three methods:

1. With the .PARAM control placed in the netlist.
2. With Let command from the command line or from a script. (If using a script you must prefix the parameter names with "global:")
3. By sweeping the value using the parameter mode of a swept analysis or multi-step analysis.

Expressions for device values must be entered enclosed in curly braces ('{' and '}'). This can be done by selecting the device and using **Shift F7** to enter the value as a text string.

Suppose we wish a 1kHz roll off for the above filter.

Using the .PARAM control, add these lines to the netlist using the **Extra Simulation Data** option.

```
.PARAM f0 1k
.PARAM alpha 1
.PARAM C1 10n
```

For more information on .PARAM

Using the Let command, you would type:

```
Let f0=1k
Let alpha=1
Let C1=10n
```

If you then wanted to alter the damping factor to 0.8 you only need to type in its new value:

```
Let alpha=0.8
```

then re-run the simulator.

To execute the Let commands from within a script, prefix the parameter names with "global:". E.g. "Let global:f0=1k"

In many cases the .PARAM approach is more convenient as the values can be stored with the Schematic using the **Extra Simulation Data** option on the **Simulation** menu.

Subcircuits

Overview

Subcircuits are a method of defining a circuit block which can be referenced any number of times by a single netlist line or Schematic device. Subcircuits are the method used to define many device models such as op-amps.

You don't need to know anything about subcircuits unless you wish to define you own device models, perhaps to build up a library of tested blocks for general distribution.

This section explains how to create a subcircuit from a schematics and how to reference it in a netlist or circuit.

Creating a Sub-circuit from a Schematic

Sub-circuits must be defined in text form as a spice netlist. However the schematic editor can be used to generate the netlist. To create a sub-circuit from a schematic, you need to identify which nodes are to be connected externally. This is done using the special **Subcircuit Pin** part from the **Spice** parts library.

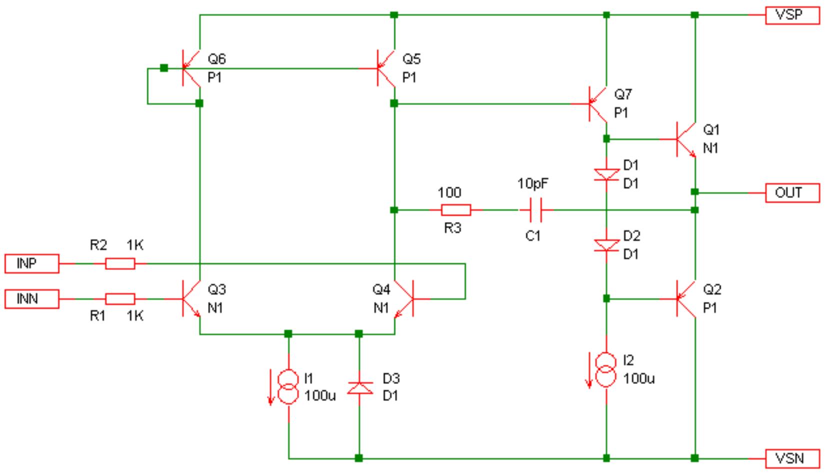
The procedure for defining a sub-circuit is as follows:-

1. Draw the circuit using the schematic editor including Subcircuit Pin components to identify external connections.
2. Use the **Parts Browser** and **Miscellaneous** category to add **Subcircuit Pins**.
3. Create spice netlist for sub-circuit from schematic using the **Simulation** menu, **Create SPICE Netlist File** sub-menu, **As Subcircuit** option.

To describe the procedure we will use an example.

Stage 1 – Draw Schematic

This is the circuit of a simple op-amp. In fact it is the circuit of our fictitious SXOA1000 op-amp used in Tutorial 3.



The five rectangular **Subcircuit Pin** components are the connections to the outside world.

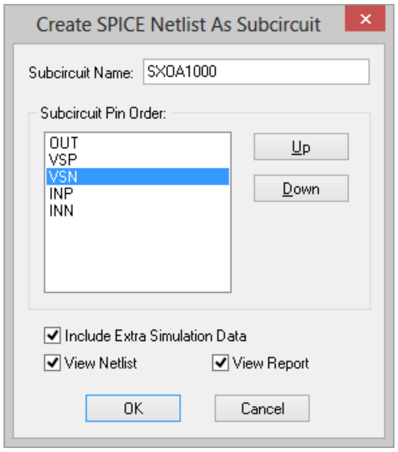
After adding, use <**F7**> on the subcircuit pins to set the required subcircuit pin name.

It is recommended that any model definitions are included in the subcircuit definition. This makes the subcircuit self-contained. If you have referenced models in the device library you can import them into the schematic automatically using the **Import Models** button on the **Extra Simulation Data** dialog accessed from the **Simulation** menu. Alternatively you can enter them into the **Extra Simulation Data** manually.

Stage 2 - Netlist creation

To create a subcircuit netlist, from the **Simulation** menu use the **Create SPICE Netlist File** sub-menu and click on the **As Subcircuit** option.

The following dialog will be displayed to allow you to define the subcircuit name and set the pin order. The name will also be used for the filename with extension .MOD.



Subcircuit Name: SXOA1000

Subcircuit Pin Order:
OUT
VSP
VSN
INP
INN

☑ Include Extra Simulation Data
☑ View Netlist ☑ View Report

Type the required name and adjust the pin order to match the part that you will create to use for the subcircuit instance. Press OK to create the subcircuit netlist. If **View Netlist** was checked the spice netlist will be displayed.

Subcircuit Spice Definition

Alternatively, you can create a sub-circuit manually using a text editor. Subcircuits begin with the .SUBCKT control and end with .ENDS. A subcircuit definition is of the form:

.SUBCKT *subcircuit_name nodelist* [params: *default_parameter_list*]
definition_lines
.ENDS

subcircuit_name is the name of the subcircuit and is used to reference it in the main netlist.

nodelist may be any number of node names and are used for external connections. The subcircuit call (using an "X" device) would use a matching number of nodes in the same order.

default_parameter_list is a list of parameters and their default values.

definition_lines is a list of valid device and model lines. In addition, .NODESET, .IC and .KEEP lines may also be placed in a subcircuit.

Where to Place Subcircuit Definition

Subcircuit definitions may be placed in a number of locations.

1. Directly in the netlist. This is the best place if the subcircuit is specific to a particular design. If you are entering the circuit using the Schematic editor, you will need to add additional lines to the netlist.
2. Put in a separate file and pull in to the Schematic with .INC control placed in the netlist.
3. Put in a library file and reference in Schematic with .LIB control placed in the netlist. Similar to 2. but more efficient if library has many models not used in the Schematic. Only the devices required will be read in.
4. Put in a library file and make global reference using Add/Remove Libraries. This will make the device globally available to all Schematics and netlists. You can also associate the subcircuit with a Pulsonix part so that you can use the Pulsonix Insert Part Using Model simulation option. These topics are covered in Tutorial 3 previous.

Subcircuit Instance

Once a subcircuit has been defined, any number of instances of it may be created. These can be created in the schematic design or directly into an existing netlist.

Schematic Entry

To call a sub-circuit in a schematic, you must choose an existing part to use or create a new part and symbol for it. The symbol must have the same number of pins and, ideally, it would also have the same pin order. In other words the order of the nodes in the .SUBCKT line would be the same as the pin order of the symbol. The matching of this order is not absolutely essential, but it makes things much easier. If they are not the same there is a way of overcoming the problem on the **Part** using the **Pin Properties** dialog accessed from the **Edit Spice** button on the part’s **Details** page.

The part must have spice type “Subcircuit” and have its spice model set to the appropriate subcircuit name (SXOA1000 in our example).

Netlist Entry

Subcircuit calls are of the form:

```
Xxxxx nodelist sub_circuitname [: parameters]
```

Each node in *nodelist* will connect to its corresponding node in the subcircuit's definition. The number of nodes in the instance must exactly match the number of nodes in the definition.

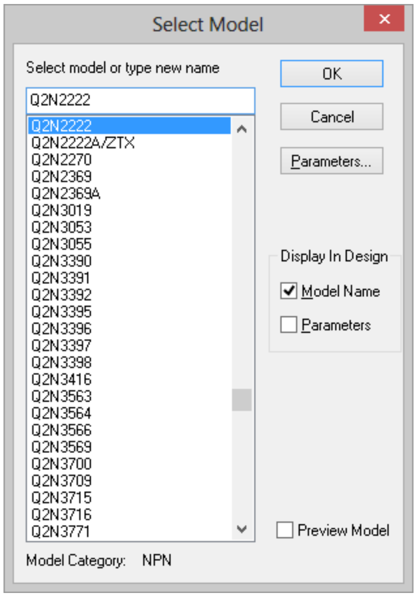
sub_circuitname is the name of the subcircuit definition.

parameters is a list of parameter names and their values which may be referenced in the subcircuit definition. Subcircuit parameters are explained below.

Passing Parameters to Subcircuits.

You can pass parameters to a subcircuit. To specify the parameters for a sub-circuit device in a schematic, you must use the following procedure:

1. Select the sub-circuit device and use **<F7>** to use the **Select Model** dialog.



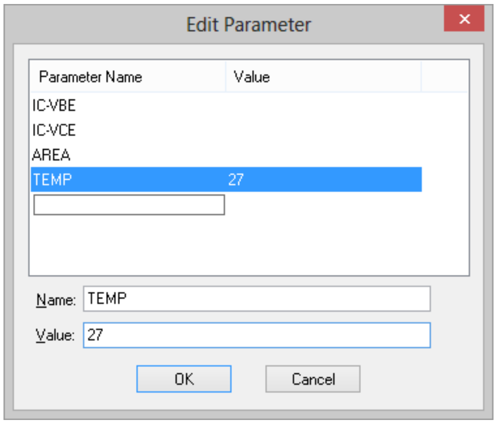
Select model or type new name: Q2N2222

Q2N2222
Q2N2222A/ZTX
Q2N2270
Q2N2369
Q2N2369A
Q2N3019
Q2N3053
Q2N3055
Q2N3390
Q2N3391
Q2N3392
Q2N3395
Q2N3396
Q2N3397
Q2N3398
Q2N3416
Q2N3563
Q2N3564
Q2N3566
Q2N3569
Q2N3700
Q2N3709
Q2N3715
Q2N3716
Q2N3771

Model name [✓], Parameters [ ], Preview Model [ ]

Model Category: NPN

2. Select the **Parameters** button to use the **Edit Parameters** dialog.



Parameter Name	Value
IC-VBE	
IC-VCE	
AREA	
TEMP	27

Name: TEMP

Value: 27

3. Click on the parameter you wish to change, or click on the blank line at the end of the parameter list to add a new parameter.

4. Press **OK** to both dialogs to accept the parameters.

When the circuit is output as a netlist, the parameters will be included at the end of the subcircuit call with a ":" character separating them from the subcircuit name.

Example:

Consider the filter example provided in above. Supposing we wanted to define several filters with different characteristics. We could use a subcircuit to define the filter but the values of the components in the filter need to be different for each instance. This can be achieved by passing the parameter values to each instance of the subcircuit.

So:

```
** Definition
.SUBCKT Filter IN OUT params: C1=1n alpha=1 f0=1k
C2 0 R1_P {C1*alpha*alpha/4}
C1 OUT R1_N {C1}
E1 OUT 0 R1_P 0 1
R1 R1_P R1_N {2/(2*pi*f0*C1*alpha)}
R2 R1_N IN {2/(2*pi*f0*C1*alpha)}
.ENDS
** Subcircuit instance
X1 V1_P VOUT Filter : C1=10n alpha=1 f0=10k
** AC source
V1 V1_P 0  AC 1  0
```

In the above example the parameters after "params:" in the .subckt line define default values should any parameters be omitted from the subcircuit instance line. It is not compulsory to define defaults but is generally recommended.

Nesting Subcircuits

Subcircuit definitions may contain both calls to other subcircuits and local subcircuit definitions.

If a subcircuit definition is placed within another subcircuit definition, it becomes local. That is, it is only available to its "host" subcircuit.

Calls to subcircuits may not be recursive. A subcircuit may not directly or indirectly call its own definition.

Global Nodes

Sometimes it is desirable to refer to a node at the circuit's top level from within a subcircuit without having to explicitly pass it. This is sometimes useful for supply rails.

Pulsonix Spice provides two methods.

1. '#' prefix. Any node within a subcircuit prefixed with '#' will connect to a top level node of the same name without the '#' prefix.
2. '$g_' prefix. Any node in the circuit prefixed '$g_' will be treated as global

The second approach is compatible with PSPICE®.

Note: the two approaches are subtly different. In the second approach the '$g_' prefix must be applied to all connected nodes, whereas in the first approach the '#' prefix must be applied only to subcircuit nodes.

Using Third-party Models

Most semiconductor manufacturers now provide a wide range of SPICE models at their web sites with which Pulsonix Spice is compatible. To use one of these models you must do the following:

1. Install electrical model in simulator's model library.
2. Associate model with Schematic symbol. (You do not always need to this.)

Examples

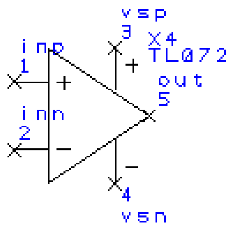
Example 1 - Using a Third Party Op-Amp

Suppose you wished to use a model for an AD847 operational amplifier. Such a device is not supplied but can be downloaded from the Analog Devices web site. An abbreviated version of the text of this particular model is:

```
* Node assignments
*               non-inverting input
*               | inverting input
*               | | positive supply
*               | | |  negative supply
*               | | |  |  output
*               | | |  |  |
*               | | |  |  |
.SUBCKT AD847   1 2 99 50 30
** Device Lines
.ENDS
```

The first 8 lines of the above are just comments but are important as they tell us the function of each terminal. The model proper starts with the line containing ".SUBCKT".

The use of .SUBCKT means that this device is a "subcircuit" and as a consequence the **SpiceDevice** component value must be set to 'X'. In fact all opamps are subcircuits but some other devices such as diodes are not. This is explained in more detail later. As this is an opamp, we will want to use one of the opamp components. Let’s have a look at the standard opamp component:



This has 5 pins in the following sequence:

inp	Non-inverting input
inn	Inverting input
vsp	Positive supply
vsn	Negative supply
out	Output

It also has the component values:

SpiceDevice	X
SpiceSubcircuit	TL072

You will notice that the symbol's 5 pins are in the same order as the terminals of the subcircuit definition. This is good news as it means we can use this symbol without any modification. If it had a different number of pins, we would not be able to use it at all whereas if the pins were in a different order, we would need to add a **SpicePinOrder** value. The only thing we need to change is the SpiceSubcircuit value. This must be set to the same as the name immediately after ".SUBCKT" in the text of the model. This is of course "AD847".

So in summary, to use an Analog Devices AD847 you must first install it. Then place a standard opamp on the schematic and use <**F7**> to change its model name to "AD847".

For the vast majority of opamps, this is all you will need to do. Most opamp models have 5 pins and follow the order shown above. There are, however, some exceptions. Some have more than 5 pins and some do not follow the standard pin configuration.

Example 2 - Opamp with Non-standard Pin Order

First, it is important to note that by "pin order" we are referring to the pin order of the electrical model. This has no connection with the pin out of the physical package. The standard pin order is described in example 1 above. Here we show the changes that are required to handle a different pin order. Here is the text of a model with a non-standard pin order:

```
* Connections:    Non-Inverting
*                 |  Inverting
*                 |  |  Output
*                 |  |  |  +Vcc
*                 |  |  |  |  -Vcc
*                 |  |  |  |  |
.SUBCKT LM6317/NS 3  2  6  7  4
** Device lines
.ENDS
```

This model is in fact in the standard model library.

If you compare this with the model in example 1, you will notice that the output is in a different position. The schematic editor needs to know about this when it creates the netlist for the simulator. To do this we must provide a Spice pin order using the **Pin Properties** option. For this device the correct value is:

1 2 5 3 4

It tells the schematic's netlist generator to output the connections to the symbols pins in the order given i.e.

1	inp = Non-inverting input
2	inn = Inverting input
5	out = Output
3	vsp = Positive supply
4	vsn = Negative supply

Example 3 - Bipolar Transistor

Here is the first two lines of the text of an On-Semiconductor MJ15023 which is a power PNP device:

.MODEL Qmj15023 pnp

```
+IS=2.16e-14 BF=69.6949 NF=0.903189 VAF=10
```

This device uses a ".MODEL" statement. This affects the value given for the **Spice Value** of the schematic component.

We would want to use the standard PNP transistor symbol for this device. For this we find that the standard values are:

Device Type	Bipolar transistor
Device Letter	Q
Model/Value	Q2N2222

The standard value for **Spice Device Type** is in fact correct. This would always be 'Q' when the second word after ".MODEL" is either "npn" or "pnp". (see below).

We will need to change **Spice Model/Value** to “Qmj15023”.

Pin order is not an issue for devices defined using .MODEL.

Example 4 - Bipolar Transistor Implemented as Subcircuit

Here is the first lines of text for the BFS17/PS transistor model

```
* 1: COLLECTOR; 2: BASE; 3: EMITTER;

.SUBCKT BFS17/PS 1 2 3

Q1 6 5 7 7 BFW92

...
```

This is implemented as a subcircuit and therefore, as we saw in examples 1 and 2, the **Spice Device Type** value must be set to 'Subcircuit'. However, the standard symbol for an NPN transistor has **Spice Device Type** set to 'Bipolar transistor'. So this must be changed to 'Subcircuit' using the **Edit Spice Type** option from the Simulation menu.. You will also need to change the **Spice Value/Model** to "BFS17/PS".

Defining/Modifying a Component for an Electrical Model

If there is a suitable Part in the library and it has the correct number of pins, then it can be used for a new electrical model. Note, that you can use any component, not just those from the Spice libraries. What is important is that the component has the right number of pins and the appropriate Spice information. If a suitable Part or symbol is not available, you can create them. See the Pulsonix manual for information on creating symbols and Parts.

If the device is a subcircuit (starts with .SUBCKT) the symbol must have Spice **Device Type** set to ‘Subcircuit’ and Device letter set to X. Otherwise set it according to the *device_type* which can be found from the .MODEL statement as follows:

.MODEL statements are always of the form:

`.MODEL` *`device_name device_type parameters`*

So we can find the SpiceDevice value from:

device_type	**Description**	**SpiceDevice**
C	Capacitor	C
D	Diode	D
LTRA	Lossy Transmission Line	O
NIGBT	IGBT	Z
NJF	N-channel JFET	J
NMF	N-channel GaAsFET	Z
NMOS	N-channel MOSFET	M
NPN	NPN transistor	Q
PJF	P-channel JFET	J
PMF	P-channel GaAsFET	Z
PMOS	P-channel MOSFET	M
PNP	PNP transistor	Q
R	Resistor	R
SRDIO	Soft Recovery Diode	D
SW	Voltage Controlled Switch	S

There are many others which are described but you will rarely see these in manufacturer's models.

In addition to the **Spice Device Type**, the symbol must also have a Spice **Value** which is set to the subcircuit or model name.

Finally, if a subcircuit's pins are not in the correct order, use a **Pin Properties** option to re-map them. See Example 2 for details on how to do this.

Schematic Topics

Displaying and Editing Net Names

The expressions and parameters on some devices need to access the names of nets in the schematic. The simplest way to find the name of a net in Pulsonix is to select the connection or pin and read the net name that is displayed on the status bar at the bottom of the schematic window. If you want to display the name in the schematic for quicker reference in the future, select a pin or junction on a net and right click to use the pop-up menu option called **Show Net Name**. Move the name to the required position. Use **Show Net Name** again on the same pin if you wish to hide the name.

Net names in Pulsonix are automatically allocated and tend to be of the form "N0001". You may wish to use a more descriptive name to simplify use in the simulator. To change the name of a net, select a connection or pin on the net and right click to use the shortcut menu option **Change Net**. You will be presented with a dialog in which you can type in your preferred net name. Be careful not to pick or type the name of an existing net, as this would force a merge of the two nets.

Net names can be any character string but may not contain spaces or the characters '#' or '.' In addition you should avoid names containing the characters:

\ " % & + - * / ^ < > [ ] ' . @ { }

These are legal but you will not be able to plot the voltage at net with names containing these characters as they conflict with expression operators and other special characters.

Adding the Ground Net

In a SPICE netlist the analog ground net must have the name "0". You can achieve this by changing all ground nets added in Pulsonix to net name "0" using the method mentioned in the previous section and merging the nets together.

This can be time consuming and lead to errors, so another quicker method has been provided. The ground net is defined as the net that uses a net class with a net type of Ground. Adding the special Documentation Symbol called “Ground” will automatically change any net it is dropped on to the ground net as defined above.

This Ground symbol can be added using the **Miscellaneous** popup toolbar accessed from the **Parts** toolbar. Any net that contains this symbol will be set to ground “0” in the netlist.

Adding Device Parameters

Many devices have additional parameters that may be specified on "per instance" basis that is they are applied to a single instance of a component. These are called "device parameters". This is opposed to "model parameters" that are specified on a "per model" basis which would apply to all components that use that model.

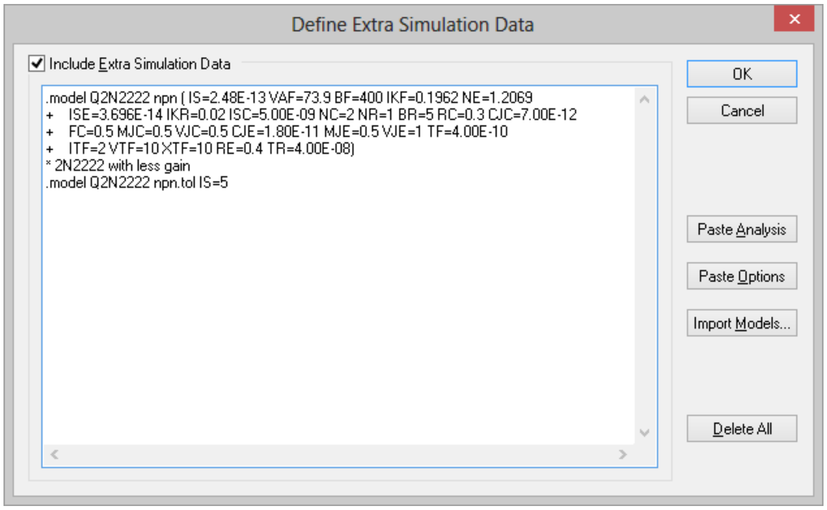
Resistors, for example, have three additional "device parameters" apart from resistance. These are TC1 (first order temperature coefficient), TC2 (second order temperature coefficient) and TEMP (local temperature).

In Pulsonix, you can add a device parameter of the form “Name=Value” simply by following the procedure shown above in the “**Passing Parameters to Subcircuits”** section. For some models there are parameters that are simple numbers or text strings following the device value or model name. In this case use **Shift F7** on the component to add these parameters after the value or model name.

For example, a Bipolar Junction Transistor has parameters “area” which is just a number and “OFF” which can be used prior to the normal “Name=Value” type parameters. **E.g.** MJ15024 25 OFF TEMP=85

Adding Extra Netlist Lines

In order to include simulator controls, local SPICE models and other user supplied SPICE statements at the end of the SPICE netlist, a facility has been provided on the **Simulation** menu under **Extra Simulation Data**.



```
.model Q2N2222 npn ( IS=2.48E-13 VAF=73.9 BF=400 IKF=0.1962 NE=1.2069
+    ISE=3.696E-14 IKR=0.02 ISC=5.00E-09 NC=2 NR=1 BR=5 RC=0.3 CJC=7.00E-12
+    FC=0.5 MJC=0.5 VJC=0.5 CJE=1.80E-11 MJE=0.5 VJE=1 TF=4.00E-10
+    ITF=2 VTF=10 XTF=10 RE=0.4 TR=4.00E-08)
* 2N2222 with less gain
.model Q2N2222 npn.tol IS=5
```

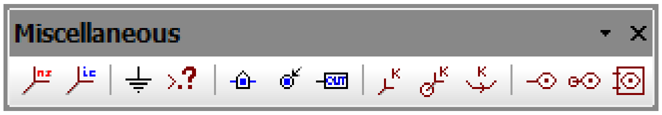
The data entered using this dialog is stored inside the schematic design.

Use **Import Models** to paste a copy of all models and sub-circuits used in the design into the data. The advantage of this is so they can have their parameters altered locally without changing the model library , and to make the schematic design complete. If the design is passed to another designer, or brought out of archive, it should still simulate.

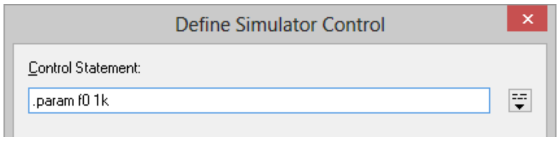
Simulator Control Part

A generic Part called **Simulator Control** is available to enable you to add control statements to the schematic rather than placing them in the **Extra Simulation Data** dialog. This way simulation parameters, like .param f0 1K for example, can be seen in the schematic design. The control statements will be added to the netlist when simulation is run.

The **Simulator Control** Part is available on the **Parts Browser** under **Miscellaneous**. It uses a **Simulator Control** spice function.



When this Part is edited using the **<F7>** key the **Define Simulator Control** dialog will be presented for you to enter one or more control statements.



Control Statement: .param f0 1k

Direct Netlist Entry

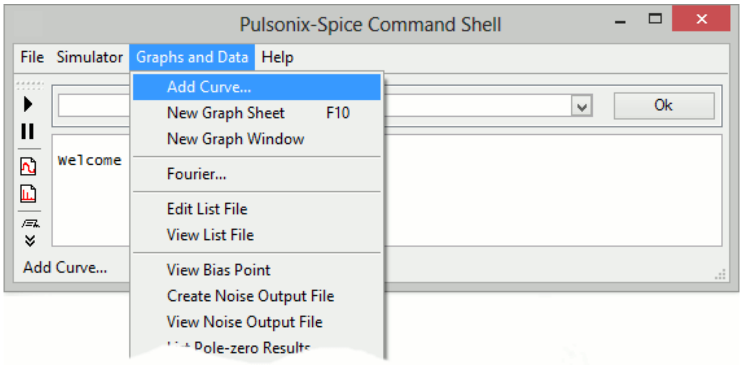
This is the traditional method of entering a circuit to the simulator. Generally entering via the schematic is a superior method as it is easier, less error prone, and is far easier to modify. However, they may be occasions when entering the circuit the "old-fashioned" way may be desirable. This section describes how to simulate a netlist direct and provides information on the netlist format.

Simulating an Existing Netlist

1. From the Pulsonix Spice **Command Shell** window, select the **Simulator** menu and **Run Netlist**
2. Select a netlist file using the dialog box. Netlist files are assumed to have the extension **.net**.
3. Select **Close**

Plotting Results from Netlist Simulation

Use the **Command Shell** menu **Graphs and Data** and the option **Add Curve**.



Netlist Format

The netlist is a text file consisting of a title line followed by a series of device lines and simulator controls. Device lines begin with a letter which signifies the type of device (e.g. Q is a BJT) followed by a list of nodes (or nets) to which that device is connected and some description of that device. Simulator controls always begin with a period (.) and are used to provide the simulator with other information such as what simulation to perform and model descriptions.

Node names can be any character string but may not contain spaces or the characters '#' or '.'. (But node names may begin with a '#' to signify a global node). In addition you should avoid names containing the characters:

\ " % & + - * / ^ < > [ ] ' . @ { }

These are legal but you will not be able to plot the voltage at node with names containing these characters as they conflict with expression operators and other special characters.

For details of device lines, each entry in the "SPICE Devices Reference" has a description titled "Netlist Entry" which gives the full format for each device.

Circuit Rules

The following design rules must be observed for the simulation to run correctly. Note that most circuits obey them anyway and they do not impose serious limitations on the capability of the simulator.

- There must always be at least one ground symbol on every circuit.
- Every node on the circuit must have a dc path to ground. If you do have a floating node, connect a high value resistor (e.g. 1G) between it and ground.
- There must not be any zero resistance loops constructed from voltage sources and or inductors. If you do have such loops. insert a low value resistor. It is best to make the resistance as low as is needed to have a negligible effect on your circuit but no lower.
- There should be at least two connections at every node.

Failure to observe the above usually leads to a **Singular Matrix** error.

Chapter 5. Device Library and Model Management

Overview

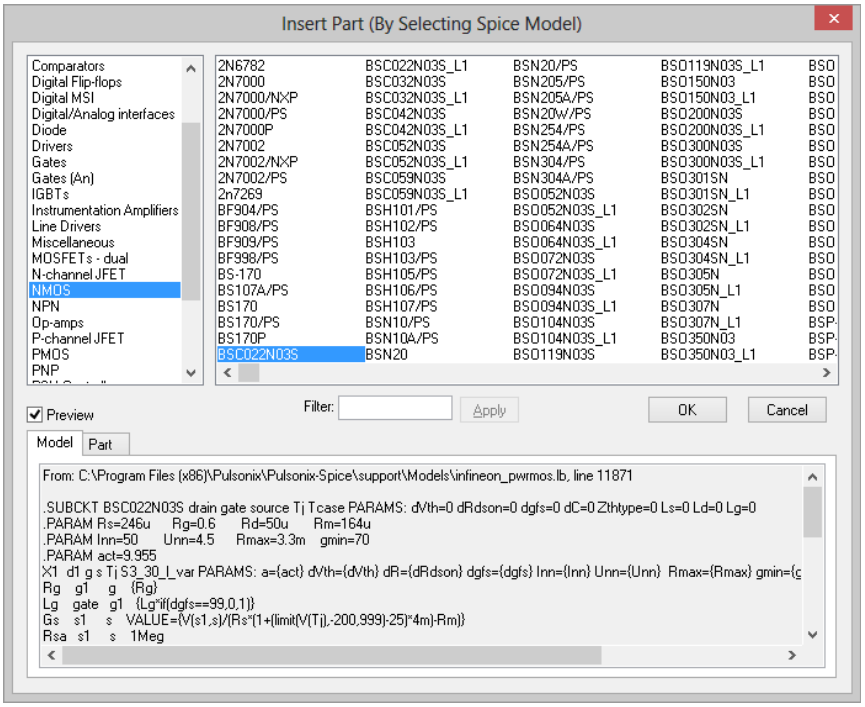
The electrical characteristics for semiconductor devices such as transistors and for more complex devices such as operational amplifiers are not built in to the simulator program but are defined in separate model library files. These are text files containing .model and .subckt controls. Some libraries have been supplied with **Pulsonix Spice** but you can obtain others from device manufacturers (usually at no cost) and other model vendors. You can also create them yourself using a text editor. Many vendor libraries may be downloaded from the Internet and installed into the Pulsonix environment.

This section explains how to install and manage model libraries, and the creation of Pulsonix Parts to refer to these Spice models.

Inserting Schematic Parts using the Model Name

Pulsonix provides a convenient method of inserting a Part when you know the SPICE model or subcircuit name. Models are arranged in categories to allow for rapid searching.

To use this tool, select the **Insert Part Using Model** option from the **Simulation** menu. The following dialog will be displayed.



```
2N6782       BSC022N03S_L1  BSN20/PS       BSO119N03S_L1  BSO
2N7000       BSC032N03S     BSN205/PS      BSO150N03      BSO
2N7000/NXP   BSC032N03S_L1  BSN205A/PS     BSO150N03_L1   BSO
2N7000/PS    BSC042N03S     BSN20W/PS      BSO200N03S     BSO
2N7000P      BSC042N03S_L1  BSN254/PS      BSO200N03S_L1  BSO
2N7002       BSC052N03S     BSN254A/PS     BSO300N03S     BSO
2N7002/NXP   BSC052N03S_L1  BSN304/PS      BSO300N03S_L1  BSO
2N7002/PS    BSC059N03S     BSN304A/PS     BSO301SN       BSO
2n7269       BSC059N03S_L1  BSO052N03S     BSO301SN_L1    BSO
BF904/PS     BSH101/PS      BSO052N03S_L1  BSO302SN       BSO
BF908/PS     BSH102/PS      BSO064N03S     BSO302SN_L1    BSO
BF909/PS     BSH103         BSO064N03S_L1  BSO304SN       BSO
BF998/PS     BSH103/PS      BSO072N03S     BSO304SN_L1    BSO
BS-170       BSH105/PS      BSO072N03S_L1  BSO305N        BSO
BS107A/PS    BSH106/PS      BSO094N03S     BSO305N_L1     BSO
BS170        BSH107/PS      BSO094N03S_L1  BSO307N        BSO
BS170/PS     BSN10/PS       BSO104N03S     BSO307N_L1     BSP
BS170P       BSN10A/PS      BSO104N03S_L1  BSO350N03      BSP
BSC022N03S   BSN20          BSO119N03S     BSO350N03_L1   BSP
```

```
From: C:\Program Files (x86)\Pulsonix\Pulsonix-Spice\support\Models\infineon_pwrmos.lb, line 11871

.SUBCKT BSC022N03S drain gate source Tj Tcase PARAMS: dVth=0 dRdson=0 dgfs=0 dC=0 Zthtype=0 Ls=0 Ld=0 Lg=0
.PARAM Rs=246u    Rg=0.6    Rd=50u    Rm=164u
.PARAM Inn=50    Unn=4.5    Rmax=3.3m   gmin=70
.PARAM act=9.955
X1  d1 g s Tj S3_30_I_var PARAMS: a={act} dVth={dVth} dR={dRdson} dgfs={dgfs} Inn={Inn} Unn={Unn}  Rmax={Rmax} gmin={g
Rg   g1    g   {Rg}
Lg   gate  g1   {Lg*if(dgfs==99,0,1)}
Gs   s1    s   VALUE={V(s1,s)/(Rs*(1+(limit(V(Tj),-200,999)-25)*4m)-Rm)}
Rsa  s1    s   1Meg
```

All models installed using the **Add/Remove Libraries** option from the **File** menu from within the Simulator's **Command Shell** or by using the Drag 'n' Drop method will be displayed. Most will be categorised. Any models implemented as a subcircuit and not found in the model database (the file ALL.CAT in Pulsonix Spice support folder) will be under category "*** Unknown ***". If you select and attempt to place an unknown model, you will be presented with the "Associate Model" dialog box explained in next paragraph. This allows you to categorise the model and - more importantly - associate a Schematic part for it.

If you can't find a model under the expected category, select the "* All Models *" category. Every model currently installed will be displayed here. (Note, the first time you access large libraries you may have to wait a second or two to see the list of models when selecting this category).

Use the filter to narrow the search and speed up locating the required model.

Use the Preview Model check box to display details from the model or subcircuit file. This needs the simulator to be running to work and may take a while to extract the data.

Note that you do not need to think about whether a model is implemented as a subcircuit or a primitive - Pulsonix Spice will sort this out automatically. E.g. some RF BJT's are implemented as subcircuits to include package parasitics but most BJT's are primitives defined with a .model control.

Model Management

Installing Model Libraries

You can install a model library very easily by placing the file or files in the same directory as the standard Pulsonix Spice libraries and then renaming them to have the .LB extension. Then in the **Simulator** select the **File** menu**, Model Library|Re-build catalog**. If your model library files need to reside in a different location then you must use the full procedure described below.

You should find many of the new devices already correctly categorised in the **Insert Part Using Model** dialog. Other devices will be in the "*** Unknown ***" category. If you select one of these, you will be required to provide some information on it so that Pulsonix Spice can choose an appropriate Schematic part. This is done using the simulators Associate Models and Parts option

Pulsonix Spice needs to know where your device model files are located. In the case of the libraries supplied the necessary mechanism will have already been set up and no further action is necessary. (But note that for upgraded installations, this may not be the case)

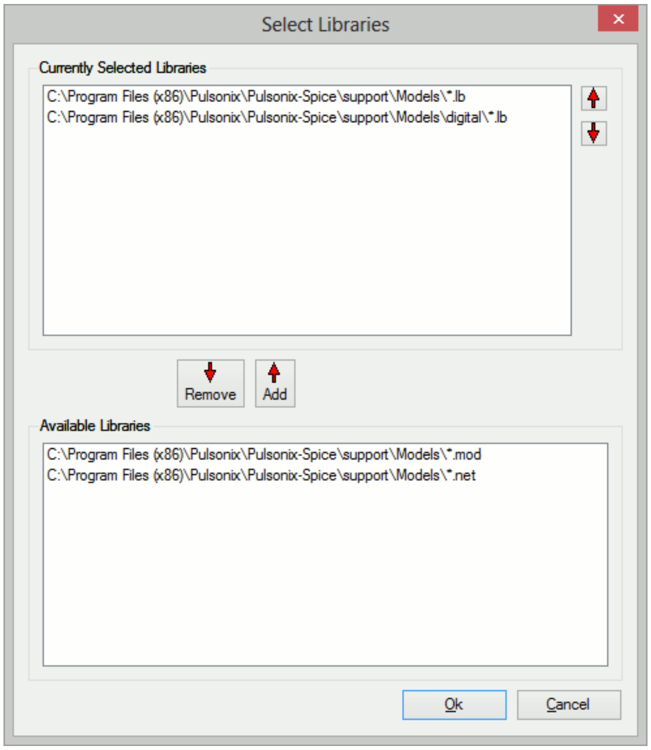
Proceed as follows:

1. Run Simulator. Form Schematic Design Editor using the **Simulation** menu, **Simulator > Simulator Command Shell** option
2. Copy the model files to a suitable directory on a fixed or remote drive. If the files are arranged in different directories, they may stay that way. Pulsonix Spice can handle any file structure.

3. Select **File|Model Library|Add/Remove Libraries...**

4. A dialog box will be displayed providing the means to select a directory. When you close this dialog box Pulsonix Spice will search the directory you specify and recuse through all sub directories below for files containing compatible models. You can specify the root of a whole disk if you wish but note that the searching operation will take several minutes if you do. Select the lowest level directory under which all models are known to reside. You must **double-click** the directory you want.

5. After the searching operation is complete another dialog will be displayed:



C:\Program Files (x86)\Pulsonix\Pulsonix-Spice\support\Models\*.lb
C:\Program Files (x86)\Pulsonix\Pulsonix-Spice\support\Models\digital\*.lb

C:\Program Files (x86)\Pulsonix\Pulsonix-Spice\support\Models\*.mod
C:\Program Files (x86)\Pulsonix\Pulsonix-Spice\support\Models\*.net

The top window shows pathnames of model libraries that are currently installed. The lower window contains additional libraries that are available. Use the Add button to transfer files in the lower window to the top window. You may use the shift and control keys to make multiple selections in each window. If you wish to remove installed libraries, select them (from top window) the press the Remove
The up and down arrow buttons allow you to change the search order of the libraries. This can be important if the same model name appears in more than one library.

When you press **OK**, Pulsonix Spice will build a catalog of models that will be used by the **Insert Part Using Model** option. This option will only display models that have been installed using the procedure above. Pulsonix Spice uses a database of over 30000 devices to categorise each model it finds in the libraries you install. So, for example, if you install the library available from Burr-Brown, when you use **Insert Part Using Model**, you will find - say - the INA102 listed under "Instrumentation Amplifiers". If you select this device an appropriate part will be used for it. Pulsonix Spice knows what an INA102 is because it is in its database. (The database is the file "ALL.CAT" which resides in the Pulsonix Spice support directory).

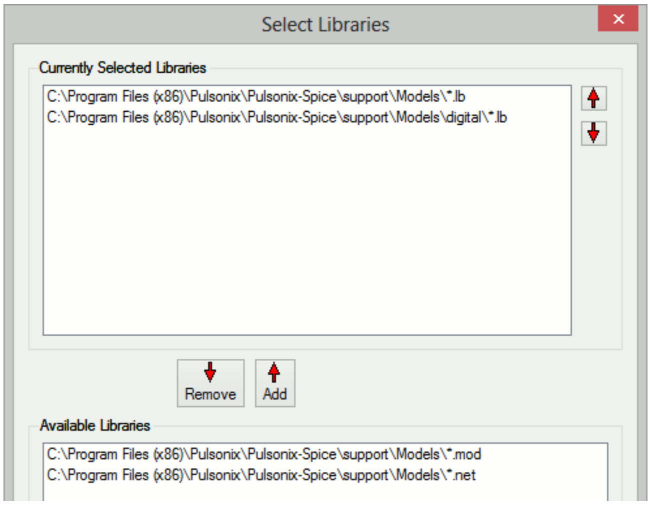
After installing models Pulsonix Spice will be able to find the electrical definition of each installed device. However the **Insert Part Using Model** dialog will not necessarily recognise all of them. If this is the case you will need to associate the unknown models with appropriate Schematic parts.

Drag and Drop Model Installation

You can install model files or folders containing files by picking them up in windows explorer and dropping them into the command shell. In most situations that is all you need to know. The following, however, explains the procedure in detail.

Note: Copy the files to a suitable directory on a fixed or remote drive. If the files are arranged in different folders, they may stay that way. Pulsonix Spice can handle any file structure.

1. Open the windows explorer and find the file, files or folder containing the device models
2. Select the items you wish to install. You can install multiple files at once – hold the control key down to select multiple items. You can also install an entire folder or multiple folders. You only thing you can't do is install files and folders at the same time.
3. Make sure that the Pulsonix Spice **Command Shell** is visible. If it is obscured, or not running you can bring it to the front by using the **Simulation** menu, **Simulator > Simulator Command Shell** option.
4. Pick up the items in windows explorer and drop them into the message window of the Command Shell.
5. If you installed individual files, you will see a message box asking you to confirm that you wish to continue, just click **OK**, the model files are now installed.
6. If you drop in folders, a search will be made in those folders for SPICE models. The **Select Libraries** dialog will then be displayed as shown below:



C:\Program Files (x86)\Pulsonix\Pulsonix-Spice\support\Models\*.lb
C:\Program Files (x86)\Pulsonix\Pulsonix-Spice\support\Models\digital\*.lb

C:\Program Files (x86)\Pulsonix\Pulsonix-Spice\support\Models\*.mod
C:\Program Files (x86)\Pulsonix\Pulsonix-Spice\support\Models\*.net

Select the items you wish to install in the lower box (**Available Libraries**) and transfer them to the upper box (**Currently Selected Libraries**) by pressing the **Add** button. You can also change the order of the items in the upper box. This affects the search order when a simulation is run.

Note

If the message:

```
Unknown file type xxx
```

is displayed when you drop a file, it means that no valid SPICE models were found in the file. It does not mean the file has the wrong extension. Pulsonix Spice will accept any extension for model files with the exception of the extensions used for graph files (sxgph).

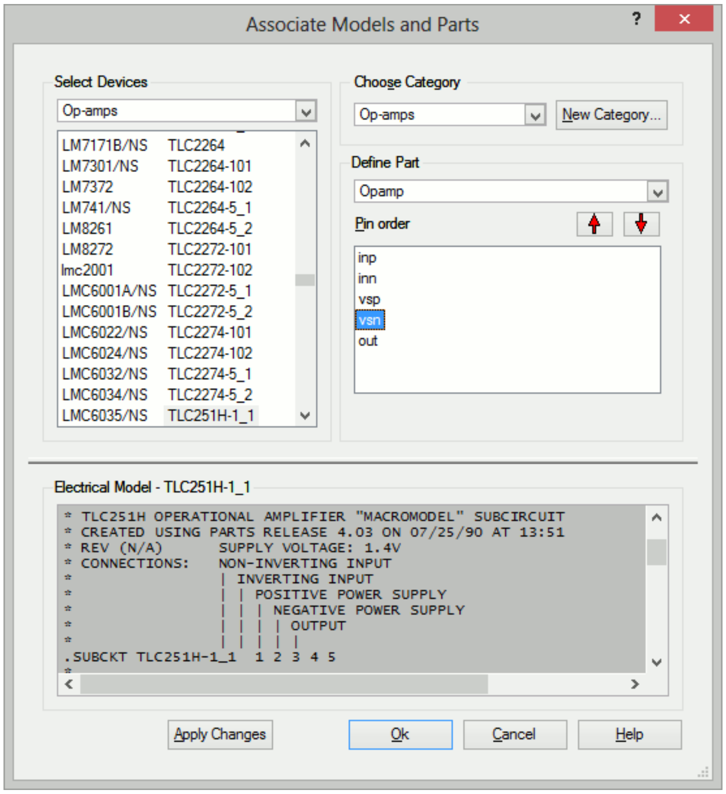
Removing Model Libraries

Select the **Remove Libraries** option from the **Model Library** menu. A dialog box similar to that shown above will be displayed but with the **Available Libraries** box empty. Select the devices you wish to remove from the **Currently Selected Libraries** box.

Associating Models with Parts

As explained above Pulsonix Spice allows you to install models using the **Add/Remove Libraries** menu. A model provides an electrical description of the device but not what Schematic part to use nor what category it should be in the **Insert Part Using Model** dialog. Pulsonix Spice is able to determine this for itself if the model is implemented using a .model control as all .models refer to a particular type of device (NPN, NMOS, Diode etc.). Models implemented as subcircuits however remain a problem as there is nothing in a .subckt definition which tells Pulsonix Spice what the device is. For example a three terminal regulator and a power MOSFET use identical syntax - Pulsonix Spice can't tell the difference. To resolve this Pulsonix Spice is shipped with a database of known model numbers providing a named Schematic part, model category and if relevant a pin mapping order. If the model is in the database, no further action is required by the user and the model will appear in the **Insert Part Using Model** dialog under the correct category and select the correct part. If the model is not in the database, you can provide the appropriate information using the "Associate Parts" dialog box.

The "Associate Parts" dialog box can be opened with the **Command Shell** menu **File|Model Library|Assoc Models and Parts**. When you do this the following dialog appears:



```
* TLC251H OPERATIONAL AMPLIFIER "MACROMODEL" SUBCIRCUIT
* CREATED USING PARTS RELEASE 4.03 ON 07/25/90 AT 13:51
* REV (N/A)       SUPPLY VOLTAGE: 1.4V
* CONNECTIONS:    NON-INVERTING INPUT
*                 | INVERTING INPUT
*                 | | POSITIVE POWER SUPPLY
*                 | | | NEGATIVE POWER SUPPLY
*                 | | | | OUTPUT
*                 | | | | |
.SUBCKT TLC251H-1_1  1 2 3 4 5
*
```

Although this dialog box looks complicated, using it is quite simple.

In the top left hand group you select the model or models that you wish to associate. The drop-down box at the top has a list of categories. Usually you would select a model or models in the ***** Unknown ***** category but you can also edit the association of known models in other categories.

Once the category has been selected, a list of models in that category will be displayed in the list box below. You should then select a model or models to associate. To select multiple items hold the control key down while selecting. Note that you will not be allowed to select multiple models that have different numbers of pins. To help you determine what type of device it is, its electrical model is displayed in the window that covers most of the lower half of the dialog box. You must now define the Category, Part and, if necessary, the Pin Order for the model. This is done using the top right hand group of controls titled "Choose Part/Category". Select an appropriate category and part.

Note that only compatible Pulsonix parts that have the same number of pins as the selected model will be shown.

If an appropriate Part is not available, cancel the dialog then create an appropriate Part using the Pulsonix Library Manager (see below for more detail). Then use the Pulsonix **Simulator Setup** option from the **Simulation** menu to tell the simulator about new parts by pressing the **Update Simulator Part List** button. After doing this, when you reopen the dialog box you will see your new part listed.

The next list box allows the pin order to be changed. Most models such as opamps and MOSFETs use a de-facto standard pin order and you probably won't need to change the default. You can check the pin order from the "Electrical Model" display at the bottom of the dialog box. Many subcircuit definitions are preceded by text which identifies each connection to the sub circuit. This must correspond exactly to the pin order of the part. If the pin order does not match you can change it using the up and down arrow buttons. Simply select a pin in the list box then move it up or down the list. Note that the change will only apply to the model(s) you are currently editing; other models associated with the same part will be unaffected.

Once you have finished selecting the category, part and pin mapping you **must** select the **Apply Changes** button. Your edits will be lost if you don't.

Embedded Association

It is possible to embed association information within the model file itself. This is useful if you wish to prepare a model to distribute to other users and wish to spare them the burden of performing the association process themselves. Models with embedded association can be installed by dropping their files in the command shell with no other action being required.

Only subcircuit devices may receive embedded association information. The information is placed in a specially formatted comment line after the .SUBCKT line but before the first device or command. The line is in the form:

*#ASSOC Category=*category* Symbol=*symbol* [Mapping=*mapping*]

category	Category for part. If it has spaces this *must* be enclosed in double quotation marks, i.e “Zenner Diode”.
symbol	Existing Pulsonix part name (it must have the correct number of pins for the subcircuit).
mapping	Mapping information. This changes the mapping between the subcircuit terminals and the part pin order. Usually, its easiest simply to arrange the subcircuit pin order to match the part pin order in which case this is not required. If however, there is some reason why rearranging the subcircuit pins is not desirable, you can instead specify the pin order using the mapping value. The mapping value is a list of part pin numbers that match to the corresponding part pin. So a mapping value of 2,3,1 says that the first subcircuit terminal connects to pin 2 of the part, the second subcircuit terminal connects to pin 3 and the third to pin 1.

Example

```
.SUBCKT IRF530 D G S
*#ASSOC Category=NMOS Symbol=nmos_sub Mapping=2,3,1
.ENDS
```

Priorities

It’s possible that association information could be provided from multiple sources in which case the possibility of conflict arises. If this is the case the following priorities apply:

1. User supplied association (e.g. using the associate parts and models dialog) takes precedence over embedded association.
2. Embedded association takes precedence over pre-defined association. Predefined association is what is stored in the ALL.CAT catalog file supplied with Pulsonix Spice.

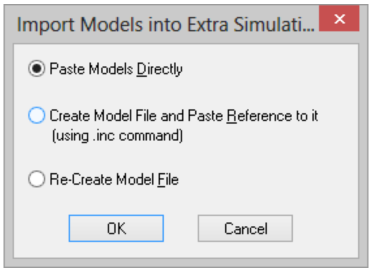
Importing Models To The Pulsonix Schematic

Pulsonix Spice provides a means to automatically import all models needed for a Schematic into that Schematic. This is done using the **Extra Simulation Data** dialog accessed from the **Schematics Simulation** menu. The models are placed in the text window.

Once the models are imported to a Schematic, it will no longer be necessary for Pulsonix Spice to locate the models in the library when a simulation is run. This has the following benefits:

1. It makes the Schematic completely self-contained. This is useful for archiving or if you wish to pass the Schematic to a third party.
2. You can edit the models locally without affecting the global library.

To import models to a Schematic, use the **Import Models** button on the **Extra Simulation Data** dialog. You will be presented with the following dialog:


Import Models into Extra Simulati...
Paste Models Directly
Create Model File and Paste Reference to it (using .inc command)
Re-Create Model File
OK
Cancel

When **OK** is pressed the models are extracted from the Simulator model libraries and saved in a file with same name as the schematics design appended with a **.mod** extension.

Use **Paste Models Directly** to paste a copy of the models from the **.mod** file directly into the Extra Simulation Data.

Use **Create Model File and Paste Reference to it** to paste a reference to the **.mod** file into the Extra Simulation Data using a .INC control.

If you previously used the **Create Model File and Paste…** option and have since changed the circuit, use **Re-Create Model File** to update the already referenced model file with any new models.

Defining Parts for Simulation

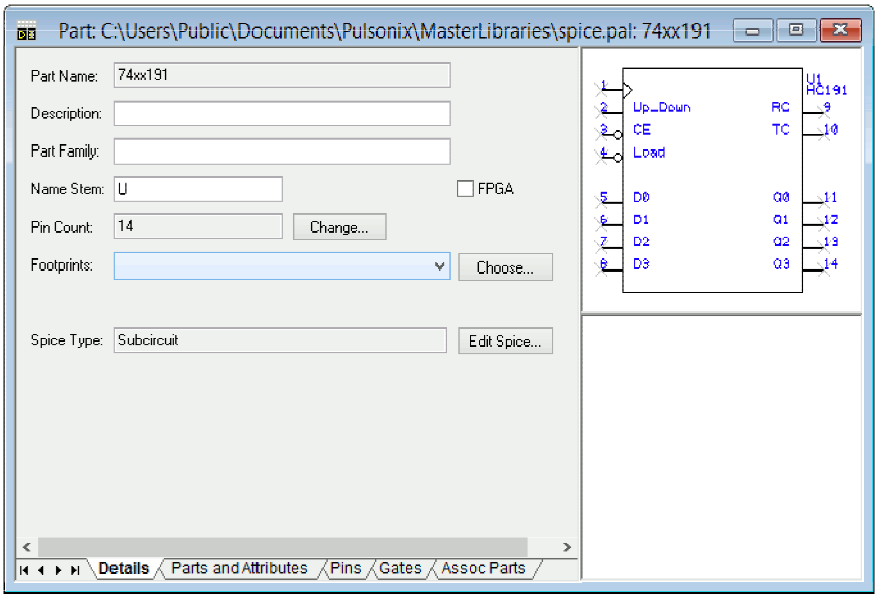
Overview

A large variety of schematic Spice parts are provided with Pulsonix which should cover many uses. However, there will be occasions when you wish to define your own new parts for use with the simulator, or to modify the spice information on one of the standard parts. This section describes how this can be done.

To use Pulsonix together with the Spice simulator the parts you use must have Part definitions which have Spice device information and a Schematic Symbol. The Pulsonix product has been created so that the same circuit created and simulated can also be passed through to the PCB design editor without further editing of Parts, or a separate process or creation of a new Schematic.

Using the Parts Editor

For a Pulsonix Part, you need to define what type of SPICE device it is. Using the Pulsonix program and the **Parts Editor** dialog from the **Library Manager**, on the Details page, you have access to the **Spice Type** field.


Part: C:\Users\Public\Documents\Pulsonix\MasterLibraries\spice.pal: 74xx191

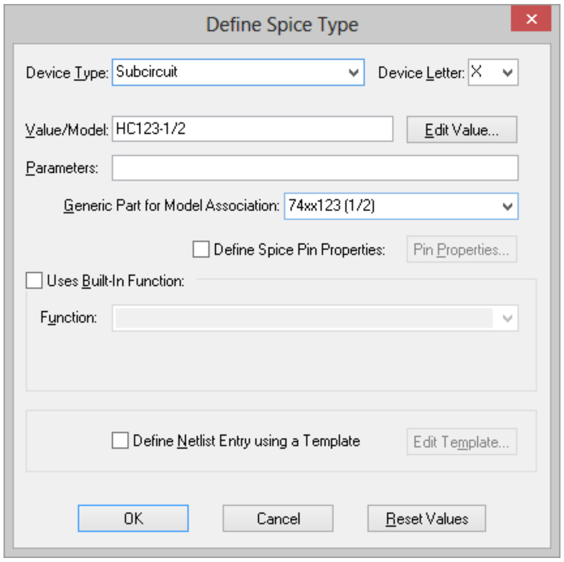
Part Name: 74xx191
Description:
Part Family:
Name Stem: U
FPGA: ☐
Pin Count: 14
Footprints:
Spice Type: Subcircuit

The dialog above shows a Schematic ‘only’ Part where no PCB Footprint has been defined yet. If you wish to pass the Schematic design into the PCB design environment then the PCB Footprint must be defined on this dialog.

Note: None of the standard Spice Parts have PCB footprints associated with them.

When you click the **Edit Spice** button you are presented with a dialog which allows you to edit the Spice information that defines the part’s simulation behaviour.

If the Part is heterogeneous (contains more than one type of symbol) then a special dialog is used to supply Spice details for each symbol type. This dialog is dealt with in the section on **Heterogeneous Parts** later.


Define Spice Type

Device Type: Subcircuit
Device Letter: X
Value/Model: HC123-1/2
Parameters:
Generic Part for Model Association: 74xx123 (1/2)
Define Spice Pin Properties: [ ]
Uses Built-In Function: [ ]
Function:
Define Netlist Entry using a Template [ ]

First select the **Device Type** from the list, or select the corresponding standard SPICE **Device Letter**. The lists contain all SPICE device types plus two special values. Use **<Unspecified>** if you don't know what SPICE device the part is, the netlister will use the first letter of the component name to specify the device type. Use **<Not Spice Device>** for parts that do not want to be included in the SPICE netlist.

The **Value/Model** field shows the default Spice value, model name or subcircuit name dependent on the device type. Use the **Edit Value** button to use specific dialogs to aid entering of this value, for example to pick from a list of model names. The value is saved into the **<Spice Value>** attribute on the Part. This allows you to add an attribute position to the corresponding symbol in order to display the value when added to a schematic design. If the device type is one of the SPICE model types or a subcircuit, other controls are shown that allow you to set up model association and pin properties. These are described in more detail later.

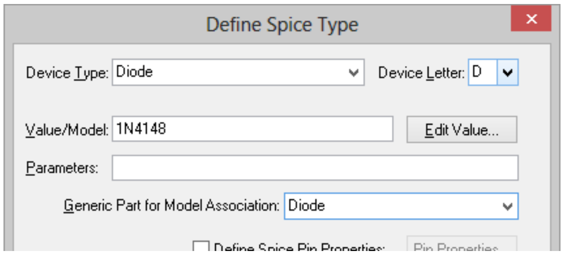
The Parameters field is for entering Spice Device parameters. These can be entered directly, if you know the correct SPICE format, or from the **Edit Value** dialogs using the **Parameters** button (if available).

By selecting the **Uses Built-In Function** box you can define the Part using one of the special built-in devices. These are covered later in more detail.

You can also select the **Define Netlist Entry using a Template** check box and use the **Edit Template** button to provide your own SPICE netlist entry for the Part definition. This is covered later in more detail.

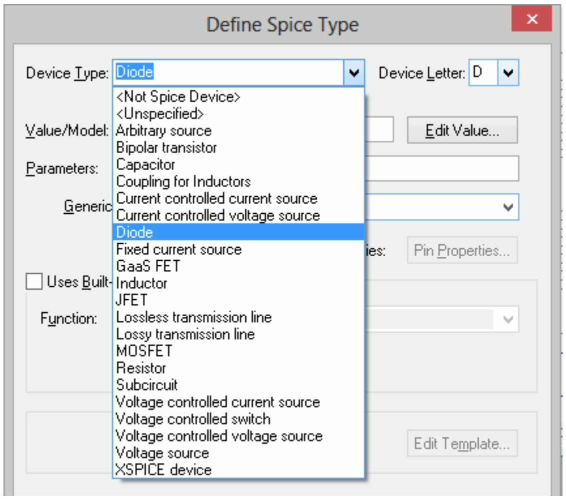
Defining Standard SPICE devices

When defining Parts which use standard SPICE Devices, use the **Edit Value** button to set up the correct values, model parameters, model name and/or sub-circuit name for the device. This is best described by using an example. The example below shows a standard diode Part:



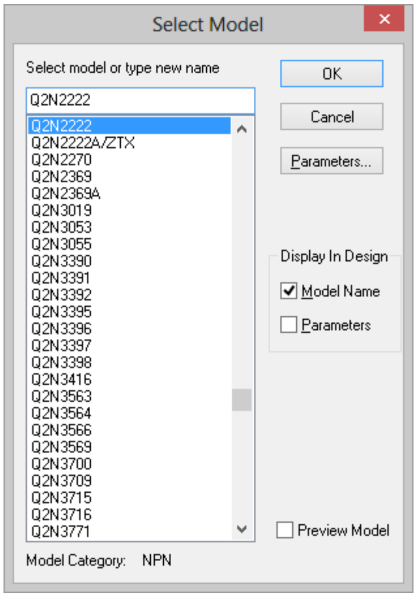
Device Type: Diode
Device Letter: D
Value/Model: 1N4148
Parameters:
Generic Part for Model Association: Diode
Define Spice Pin Properties:

By editing the **Generic Part for Model Association** using the drop down list you can specify which Part is associated with the simulator’s models, in this case **Diode** which is a generic Part.



In other cases the Part could be more specific to a particular Diode model, e.g. a Part called D1N4148 could be created for transferring to the correct PCB device for that model. This would still refer to the generic Part called Diode but would have a specific value.

In this case, using the **Edit Value** button will display the **Select Model** dialog, in this example for the Diode SPICE models available:



Select model or type new name

Q2N2222

Q2N2222
Q2N2222A/ZTX
Q2N2270
Q2N2369
Q2N2369A
Q2N3019
Q2N3053
Q2N3055
Q2N3390
Q2N3391
Q2N3392
Q2N3395
Q2N3396
Q2N3397
Q2N3398
Q2N3416
Q2N3563
Q2N3564
Q2N3566
Q2N3569
Q2N3700
Q2N3709
Q2N3715
Q2N3716
Q2N3771

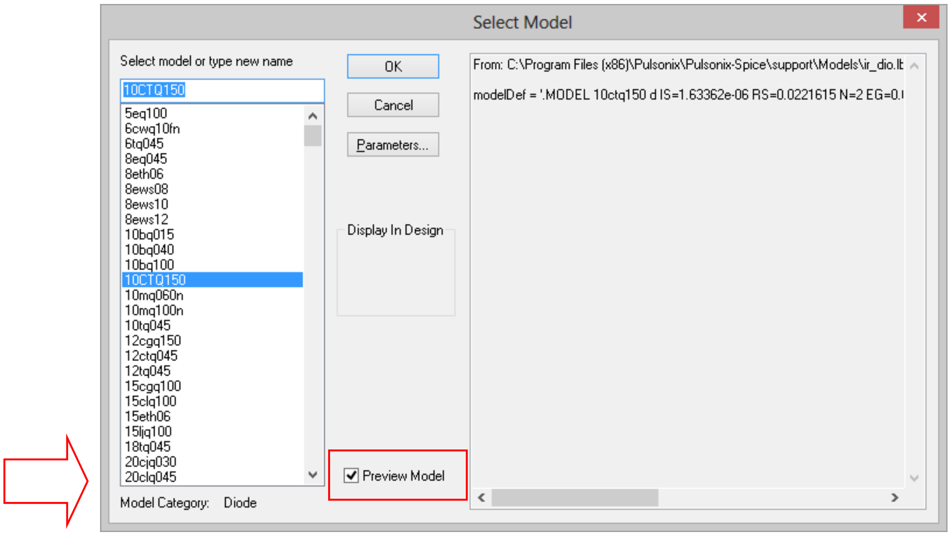
[✓] Model Name
[ ] Parameters
[ ] Preview Model

Model Category: NPN

The list will contain all models from the simulators model libraries which have been associated with the same Pulsonix generic Part (of Diode in this case).

Different **Edit** dialogs are available for the different SPICE device types. For example, a resistor will present a dialog to use to define the correct value for resistance.

The **Preview** button will start the **Command Shell** and access the Spice model. This is displayed in the **Preview Model** window:



Select model or type new name

10CTQ150

5eq100
6cwq10fn
6tq045
8eq045
8eth06
8ews08
8ews10
8ews12
10bq015
10bq040
10bq100
10CTQ150
10mq060n
10mq100n
10tq045
12cgq150
12ctq045
12tq045
15cgq100
15clq100
15eth06
15ljq100
18tq045
20cjq030
20clq045

[✓] Preview Model

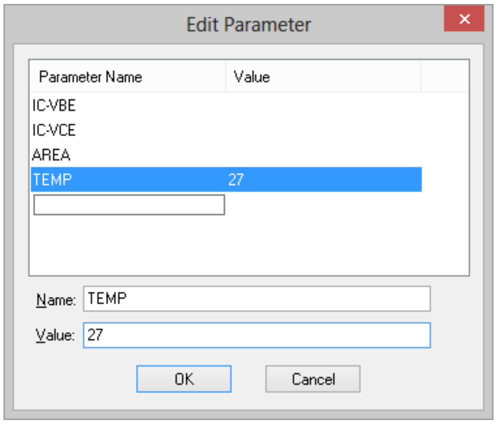
Model Category: Diode

From: C:\Program Files (x86)\Pulsonix\Pulsonix-Spice\support\Models\ir_dio.lt

modelDef = '.MODEL 10ctq150 d IS=1.63362e-06 RS=0.0221615 N=2 EG=0.

Editing Parameters

The **Parameters** button is used to change the SPICE parameters for the device. If the Parameters button is visible on the **Edit Value** dialog, press it and the following dialog will be displayed:



Parameter Name	Value
IC-VBE	
IC-VCE	
AREA	
TEMP	27

Name: TEMP

Value: 27

Parameter List

This will contain all parameters previously added to the Part and any in-built parameter names relevant to the type of device you are editing. For example, for a resistor the list will always contain **TEMP**, **TC1** and **TC2**.

Editing A Parameter

Click on the parameter name in the list. Use the **Name** and **Value** controls to change the parameter.

Adding A New Parameter

Click below the last parameter name in the list, or if there are no names click at the top of the list box. Use the **Name** and **Value** controls to add the required parameter.

Refer to the SPICE Device Reference chapter for information on which parameters each type of device can have.

Removing A Parameter

To remove a Parameter's Value from a Component, select and clear it from the list.

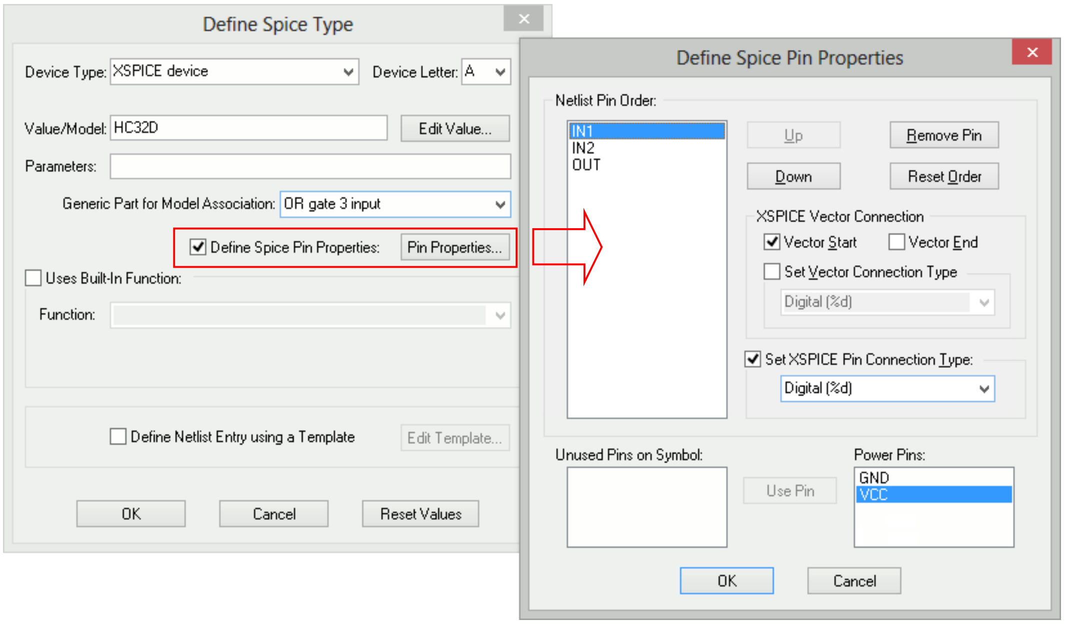
Pin Properties

From the **Define Spice Type** dialog, check the **Define Spice Pin Properties** check box and select the **Pin Properties** button to set up spice information about a parts pins. This is used for the following reasons:

1. To redefine the order of pins output to the SPICE netlist.

2. To disable a symbol pin from being output to the SPICE netlist.
3. To include power pins on the device in the SPICE netlist.
4. To define XSPICE Vector Connections, and XSPICE Connection types.

The following dialog will be displayed:



Device Type: XSPICE device
Device Letter: A
Value/Model: HC32D
Parameters:
Generic Part for Model Association: OR gate 3 input
☑ Define Spice Pin Properties:
☐ Uses Built-In Function:
Function:
☐ Define Netlist Entry using a Template

Netlist Pin Order: IN1, IN2, OUT
XSPICE Vector Connection
☑ Vector Start ☐ Vector End
☐ Set Vector Connection Type: Digital (%d)
☑ Set XSPICE Pin Connection Type: Digital (%d)
Unused Pins on Symbol:
Power Pins: GND, VCC

Netlist Pin Order

This shows which pins will be output to the SPICE netlist for this part, in the order that they will be written. The gate pins are either represented by their logic name, or schematics symbol pin number if logic names are not used.

If one of the ungated power pins is included, its name is the default net name assigned to it in the part preceded by "Power Pin:".

Select one of the pins by clicking on its name. The appropriate controls will then be enabled for acting on this selected pin.

Up and Down

Press these buttons to change the order of the pins by moving the selected pin name up or down in the list.

Remove Pin

Press this to remove a pin from the **Netlist Pin Order** list. This pin will not be included in the pin list written to the netlist for this part.

The removed pin name will be placed into the appropriate **Unused Pins On Symbol** or **Power Pins** list.

Reset Order

Press this to reset the order to match the symbol.

Unused Pins On Symbol

This list shows any pins on the symbol that will not be output to the netlist for this part. If you want to include a pin from this list, select it and press the **Use Pin** button.

Power Pins

This list shows any pins on the part not assigned to any gates, and have not been included in the **Netlist Pin Order** list. If you want to include a pin from this list, select it and press the **Use Pin** button.

For heterogeneous parts, if a gate uses the Spice type <Not Spice Device> then it's power pins will also appear in this list. This enables you to use a power gate on the part and still include its power pins on the model of the first gate on the part.

XSPICE Connections

These controls are enabled if the Spice Device Type for the part is set to "**XSPICE device**".

XSPICE Connection Types

In the device reference section of this manual, each XSPICE device has a table titled "Connection Details". Each table has the column entries "Default type" and "Allowed types". The type referred to here is the type of electrical connection e.g. voltage, current, differential or single-ended.

Some devices allow some or all of their connections to be specified with a range of types. For example, the analog to digital converter has a single ended voltage input by default. However, using a simple modification to the netlist entry, an ADC can be specified with a differential voltage input or even a differential current. Changing the type of connection involves no changes to the .MODEL control.
The following table lists all the available types. The modifier is the text used to alter a connection type at the netlist level. This is explained below.

Description	**Modifier**
Single ended voltage	%v
Single ended current	%i
Differential voltage	%vd
Differential current	%id
Digital	%d
Grounded conductance (voltage input current output)	%g
Grounded resistance (current input, voltage output)	%h
Differential conductance (voltage input current output)	%gd
Differential resistance (voltage input current output)	%hd

As well as type, all connections also have a direction. This can be in, out or inout. Voltage, current and digital connections may be in or out while the conductance and resistance connections may only be inout. Voltage inputs are always open circuit, current inputs are always short circuit, voltage outputs always have zero output impedance and current outputs always have infinite output impedance.

The conductance connections are a combined voltage input and current output connected in parallel. If the output is made to be proportional to the input , the connection would be a conductor with a constant of proportionality equal to its conductance, hence the name. Similarly, the resistance connections are a combined current input and voltage output connected in series. If the output is made to be proportional to the input, the connection would be a resistor with a constant of proportionality equal to its resistance.

Set XSPICE Pin Connection Type

If a model allows one or more of its connections to be given a different type , this can be done by checking the **Specify Type** box and choosing the correct type from the drop down list.

For example if you wish to specify a 4 bit ADC with a differential voltage input, select the pin named ANALOG_INP and set its pin type to **Differential Voltage**. The netlist entry would be something like:

A1 %vd ANALOG_INP ANALOG_INN CLOCK_IN [DATA_OUT_0 DATA_OUT_1 DATA_OUT_2 DATA_OUT_3] DATA_VALID ADC_4

XSPICE Vector Connections

Some XSPICE models feature an arbitrary number of inputs or/and outputs. For example, an AND gate can have any number of inputs. It would be inflexible to have a separate model for every configuration of AND gate so a method of grouping connections together was devised. These are known as vector connections. Vector connections are enclosed in square brackets. E.g. the netlist entry for an AND gate is:

Axxxx [in_0 in_1 .. in_n] out model_name

The pins in_0 in_1 to in_n form a single vector connection. Any number of pins may be placed inside the square brackets, in fact the same model may be used for devices with different numbers of inputs.

Check the **Vector Start** and **Vector End** boxes to define the first and last pin in the vector connection.

Set Vector Connection Type

It is possible to set the type of the entire vector connection. This type will be applied to all pins (or ports) in the vector. If you want to set this , it must be set on the pin that is marked as the start of the vector. E.g. the netlist entry for an AND gate were the vector is of type digital is :

Axxxx %d [in_0 in_1 .. in_n] out model_name

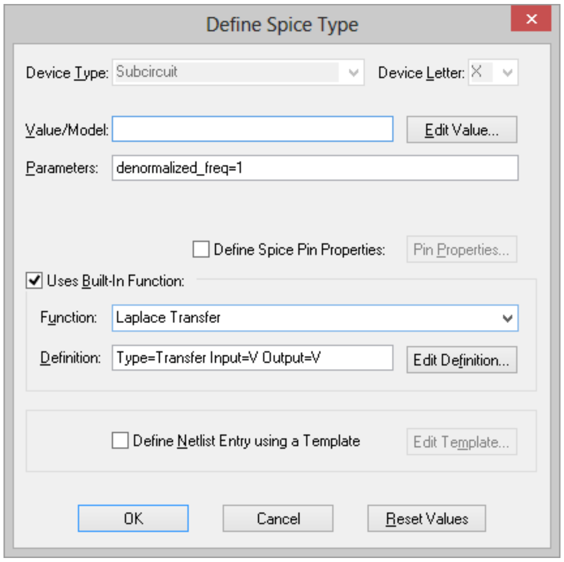
Using Built-in SPICE Functions

Pulsonix contains many built-in XSPICE devices and sub-circuits that have their own specific **Edit Value** dialogs to make it easier for you to set up the device.

Most of these also have their netlist entry defined using a Spice template. This template can be edited if, for example, you wish to use alternative built-in parameters or sub-circuit naming conventions.

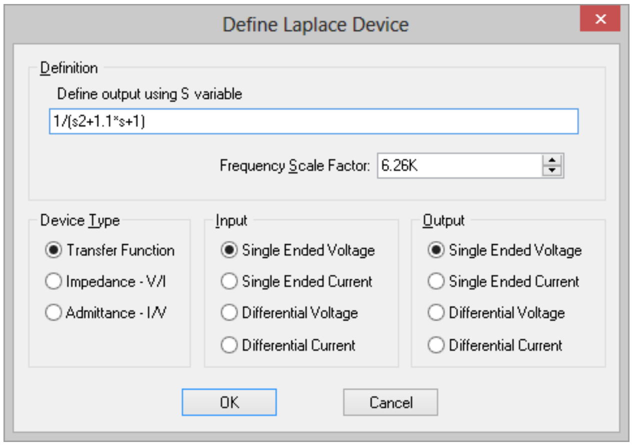
Some of these built-in devices have functional definitions that provide extra information to describe the function of the part. In this case the **Definition** field and **Edit Definition** button will appear. The definition field will contain parameters that define the function of the device, making it different from other devices using the same built-in function name.

The built-in SPICE functions are best illustrated using an example. In this example we have selected the **Laplace Transfer** function:


Define Spice Type

Device Type: Subcircuit
Device Letter: X
Value/Model:
Parameters: denormalized_freq=1
Define Spice Pin Properties: [ ]
Uses Built-In Function: [✓]
Function: Laplace Transfer
Definition: Type=Transfer Input=V Output=V
Define Netlist Entry using a Template [ ]

The Laplace Transfer function uses a **Definition** field. This is used to define the functional details for this particular part. Press the **Edit Definition** button to use a specific dialog to edit these details. Make sure the symbol used by the part matches the definition.


Define Laplace Device

Definition
Define output using S variable
1/(s2+1.1*s+1)
Frequency Scale Factor: 6.26K

Device Type: [● Transfer Function] ○ Impedance - V/I ○ Admittance - I/V
Input: [● Single Ended Voltage] ○ Single Ended Current ○ Differential Voltage ○ Differential Current
Output: [● Single Ended Voltage] ○ Single Ended Current ○ Differential Voltage ○ Differential Current

Each built-in function has its own specific **Define** dialog, each are covered in more detail in the Circuit Definition chapter.

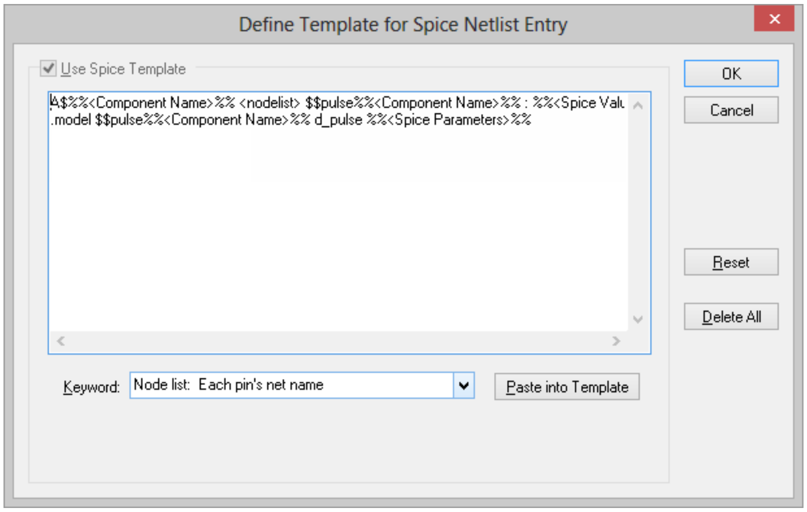
This dialog is also used when pressing the <**F7**> key when selecting one of these parts in a Schematic design. Selecting a different configuration will replace the Part in the design with the Part from the library matching the new definition.

Using SPICE Templates

Use this dialog set up your own SPICE netlist entry for a Part using a template containing plug in values that are satisfied when the netlist is generated prior to Simulation.

Normally a single line is created in the netlist for each schematic component. The line is created according to the device type, value and model name properties. If, however, a template entry has been specified this system is bypassed and the netlist entry is defined by the value of this entry.

To add your own netlist entry, check the **Define Netlist Entry using a Template** box and press the **Edit Template** button. The following dialog will be displayed:



```
A$%%<Component Name>%% <nodelist> $$pulse%%<Component Name>%% : %%<Spice Valu
.model $$pulse%%<Component Name>%% d_pulse %%<Spice Parameters>%%
```

Keyword: Node list: Each pin's net name

Use Spice Template

Check this box to enable use of the template netlist entry.

Template Entry

Type the multi-line netlist entry. Any text typed here will be output as the device entry in the SPICE netlist prior to Simulation.

Plug in Keywords

Plug in Attributes and some special keywords can be inserted into the template entry. These have their calculated values substituted into the netlist entry at the time of its generation. First click the cursor in the template entry at the place you want the keyword to be inserted, or select the text in the template entry you wish to replace with the keyword. The double percent characters mark the delimiter.

Select the required keyword from the keyword dropdown list and press the **Paste into**

Template button to insert the keyword into the template.

Selecting the **Attribute** keyword will display an edit box for you to supply the attribute name. Selecting **Net name of specified pin** keyword will display an edit box for you to specify the required pin name or schematics symbol pin number to specify which pin.

Here is a list of the keywords provided along with a description of what is plugged in when the netlist is generated:-

Attribute - The value of the specified attribute.

Component name - The component's name e.g. U4.

First pin's name - The name of the first pin on the component.

Name of pin connected to first pin - The name of the pin on the other end of the connection attached to the specified pin.

Name of specified pin - The name of the net that the specified pin is on.

Node list: Each pin's net name - The names of the nets that each pin is on, listed in schematic symbol pin order.

Part name - The name of the component's part.

Pin list for a sub-circuit call - The text "params:" followed by each of the component's pin names in schematic symbol pin order. Use at end of a SPICE sub-circuit call.

Pin list: Each pin's name - Each of the component's pin names in schematic symbol pin order.

Spice parameters - The value of the <Spice Parameters> attribute.

Spice tolerances - The value of the <Spice Tolerance> attribute.

Spice value/model - The value of the <Spice Value> attribute.

Notes: If a pin name is plugged in, the schematics symbol pin number will be used if the specified pin has no name. The schematic symbol pin order can be changed for Spice output by using the ***Pin Properties*** *option on a part's spice dialog.*

Reset

Sets the template entry back to the original text when the dialog was first displayed.

Delete All

Clears the template.

Multiple Symbol Parts

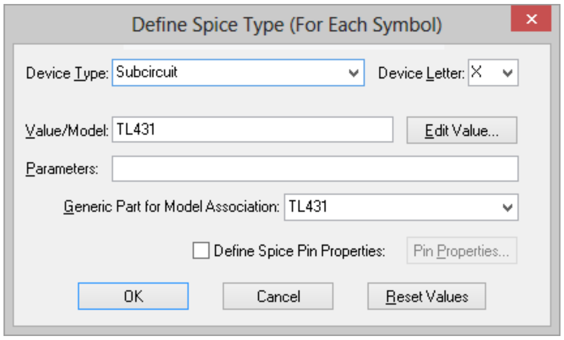
In Pulsonix, Parts are normally designed for creating PCB components, so one part may contain more than one symbol, four NAND2 gates for example. The netlister outputs each gate in the schematic as a separate device, so needs spice information for each of them. These parts fall into the two categories, explained below.

Homogeneous Parts

If the Part contains more than one gate, all using the same symbol, the Spice information defined for the Part will be used for each gate.

Heterogeneous Parts

If the Part consists of more than one type of symbol, you need to set up the Spice model for each of the symbol types. In this case the following dialog will be displayed:


Define Spice Type (For Each Symbol)

Device Type: Subcircuit

Device Letter: X

Value/Model: TL431

Parameters:

Generic Part for Model Association: TL431

☐ Define Spice Pin Properties:

Select each **Symbol** in turn and set up its Spice details. See the main section above for how to set up the controls shown on the dialog.

Heterogeneous parts can only have model or subcircuit names. You cannot use a template or special function for these parts.

Note: If using one of the symbols as a power gate, set its type to **<Not Spice Device>**. This way you will be able to use its power pins in the **Pin Properties** dialog for the other symbols if their Spice model requires them.

Simulator Device Pin Names

The following information is needed to define schematic parts for the various Spice devices supported by the simulator.

In order to be able to plot component pin currents the pin names for the schematic part must match up with those used by the simulator. So for a BJT (bipolar junction transistor) the simulator refers to the four pins as "b", "c", "e" and "s" for base, collector, emitter and substrate. The same letters must also be used for the pin names for any schematic BJT part. The simulator device pin names are listed below.

The **Spice Device** is the schematic part property which describes what type of Spice device the part refers to. SPICE uses the first letter of the component reference to identify the type of device. The Pulsonix Spice netlister prepends the model property (and a '$' symbol) to the component reference to comply with this. This makes it possible to use any component reference on the schematic.

Device	Spice Device	Pin no.	Pin names	Pin function
XSPICE device	A			See documentation for particular device.
Arbitrary Sources	B	1	p	
		2	n	
Bipolar junction transistors	Q	1	c	Collector
		2	b	Base
		3	e	Emitter
		4	s	Substrate
Capacitor	C	1	p	
		2	n	
Current Controlled Current Source (2 terminal)	F	1	p	
		2	n	
Current Controlled Current Source (4 terminal)	F	1	p	+ output
		2	n	- output
		3	any name	+ control current
		4	any name	- control current
Current Controlled Voltage Source (2 terminal)	H	1	p	
		2	n	

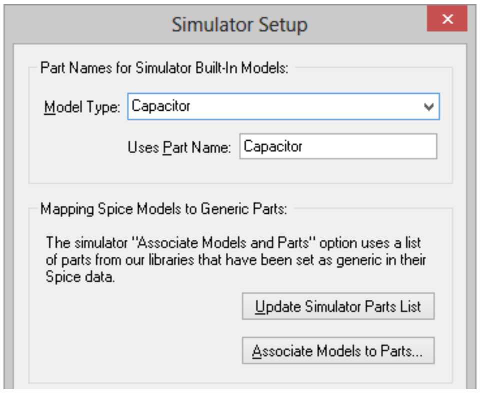
Current Controlled Voltage Source (4 terminal)	H	1	p	+ output
		2	n	- output
		3	any name	+ control current
		4	any name	- control current
Current Source	I	1	p	+
		2	n	-
Diode	D	1	p	Anode
		2	n	Cathode
GaAsFets	Z	1	d	Drain
		2	g	Gate
		3	s	Source
Inductor	L	1	p	
		2	n	
Junction FET	J	1	d	Drain
		2	g	Gate
		3	s	Source
MOSFET	M	1	d	Drain
		2	g	Gate
		3	s	Source
		4	b	Bulk (substrate)
Resistors	R	1	p	
		2	n	
Subcircuits	X	Pins can be given any name. Numbering must be in the order that pins appear in the .subckt control which defines the subcircuit. Pulsonix Spice uses a special extension of the netlist format to tell the simulator what the pin names are.		

Transmission Lines	T (lossless) O (lossy)	1	p1	Port 1 Terminal 1
		2	n1	Port 1 Terminal 2
		3	p2	Port 2 Terminal 1
		4	n2	Port 2 Terminal 2
Voltage Controlled Current Source	G	1	p	+ output
		2	n	- output
		3	cp	Non-inverting control input
		4	cn	Inverting control input
Voltage Controlled Switch	S	1	p	Switch terminal 1

Model Association to New Parts

After adding new parts to the Pulsonix libraries that correspond to models or subcircuits contained in the Simulator's model libraries, you may wish to use the Simulator's **Associate Models and Parts** dialog to map these new parts to model and subcircuit names. This will then enable you to use the **Insert Part Using Model** option when editing a schematic design. Note, it is the generic part name defined in the parts spice information that is used in part to model association.

Before you can set up this association, you need to inform the Simulator about the new parts. To do this, from the Pulsonix Schematic, select the **Simulation** menu and use the **Simulator> Simulator Setup** option. The following dialog is displayed:


Simulator Setup

Part Names for Simulator Built-In Models:

Model Type: Capacitor

Uses Part Name: Capacitor

Mapping Spice Models to Generic Parts:

The simulator "Associate Models and Parts" option uses a list of parts from our libraries that have been set as generic in their Spice data.

Part Names for Simulator Built-in Models

The Pulsonix-Spice simulator contains SPICE models that use a .Model definition to define them. The simulator uses a built-in table to find the required part according to what type of model it is. Use this table here to provide your Pulsonix part names for these standard model types.

Update Simulator Parts List

Use this button to inform the Simulator about new Pulsonix Parts.

Associate Models to Parts

This is a quick way of getting to the Simulator’s Associate dialog.

Sundry Topics

.LIB Control

The .lib control allows the local specification of model library for a particular circuit

Library Diagnostics

When enabled, library diagnostics display messages showing the progress of the location of device models. To enable/disable select **File|Options|General** then **Model Library**.

Local Models

You can also enter a model or subcircuit directly in the netlist in which case the library will not be searched for that device. Local models always take precedence. Refer to the previous chapter *Getting Started* section on *Adding Extra Netlists* to find out how to add extra netlist lines to a schematic.

Library indexing mechanism

This is a technique used to speed the search for models and subcircuits. It is completely transparent and requires no action from you. Pulsonix Spice creates an index file for each library specification (i.e. cache, LibFile option or .lib entries) it encounters. This index files contain details of the file locations of models and subcircuit definitions referenced by the library specification. These index files can then be used for later simulation runs to speed the search for models and subcircuits. Index files are automatically rebuilt if any of the library files referenced are modified. (Modifications are detected by comparing file dates). All index files are named SIMXIDX.*n* where *n* is a number from 0 to 999 and are stored in your Application Data directory.

In earlier releases these index files were read and checked every time a simulation was run. For very large libraries, this could take several seconds as it checked the file dates of all model files to see if they had been modified. With the current release, these indexes are loaded into memory just once when Pulsonix Spice starts. Further, the loading of the indexes is performed in the background so that in most cases, by the time a simulation is run, they are already loaded and devices can be located almost instantaneously.

There is however a drawback to this approach. Now that the indexes are loaded only once, no checks are made to detect if any models have changed. This is not a problem if a modification is made to a model already in the index as, if it is actually used in a design, Pulsonix Spice will check its date (and if necessary re-index it) when it reads the model definition in. The problem arises when new models are added to a library. Pulsonix Spice won't know that they exist and won't find them until the indexes are reloaded. When this happens it will be necessary to manually instruct Pulsonix Spice to reload its indexes. This can be done with the **File** menu option **Model Library|Re-build Catalog**

Catalog Files

The information in this topic maybe of interest to users wishing to understand the inner workings of the model category system. The data for model and suggested Pulsonix parts stored in the catalog files. Some of these files live in the Application folder on your computer. If you don’t know where this is, use the utility program …Pulsonix-Spice\Support\Help\FindAppDataDir.exe to show the folder path.

ALL.CAT	Resides in Pulsonix Spice support directory. Stores catalog data for over 30000 devices and is supplied with Pulsonix Spice. Pulsonix Spice never modifies this file.
USER_V2.CAT	Resides in the application data directory. Stores catalog data supplied by the user. Data in this file overrides data in ALL.CAT. The Associate Models dialog box writes to this file.
OUT.CAT	Resides in the application data directory. This is generated by the re-build category function from information in ALL.CAT and USER_V2.CAT and installed models. It will also be automatically created if it does not already exist. You can also force it to be rebuilt at any time by selecting menu File|Model Library|Re-build Catalog

File Format

Catalog files are text files. Each line provides data about a single device in semi-colon delimited fields. The fields are as follows:

Field 1	Device name as it appears in the Insert Part Using Model dialog.
Field 2	Pulsonix Component name
Field 3	Model property - X for subcircuits, as appropriate for other devices. (If this field is empty in ALL.CAT and USER_V2.CAT it is determined automatically from electrical model when the catalog is built)
Field 4	Category
Field 5	Sub-category (currently not used)
Field 6	Pin mapping order
Field 7	Path name (duplicate device names only)

When you re-build the model catalog the model information will be written to a file called "OUT.CAT". This is located under the application_data directory. This is in the same format as "ALL.CAT" in the root folder. "ALL.CAT" is never modified. The process of building "OUT.CAT" may take a few seconds if the model library is large. If the models haven't yet been indexed then this will be done first. The indexing operation can take several minutes for a large library.

We recommend that you do not alter the ALL.CAT file. This file will be overwritten when you upgrade to a future version and any edits you make will be lost. OUT.CAT is generated by Pulsonix Spice and should never be manually edited.

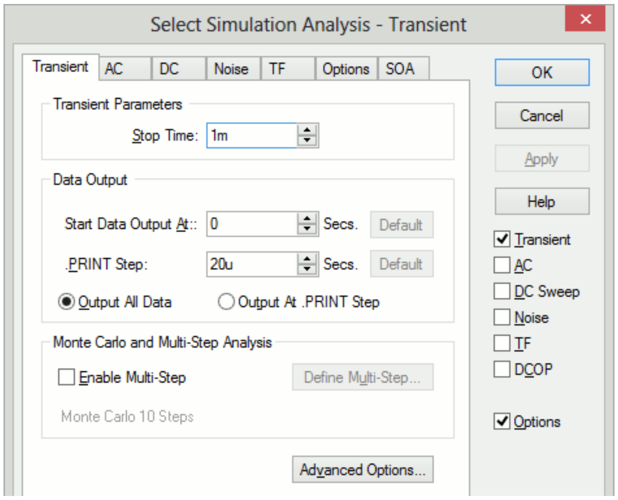
Your edits in the **Associate Models** dialog are written to USER_V2.CAT in the application data folder. If you do edit USER_V2.CAT to change the model categories, you must rebuild OUT.CAT afterwards. To do this select the menu File|Model Library|Re-build Catalog. This may take a few seconds.

Chapter 6. Analysis Modes

Overview

In this chapter we describe the various analysis modes available and how to set them up from the schematic editor. There is more information on analysis modes including full details of the netlist commands to invoke them, in the Simulator chapter later.

Most of the analyses can be setup using the **Simulation Parameters** dialog Box which is opened with the schematic menu **Simulator> Simulation Parameters**…



Transient Parameters

Stop Time: 1m

Data Output

Start Data Output At:: 0 Secs.

.PRINT Step: 20u Secs.

[✓ Output All Data] Output At .PRINT Step

Monte Carlo and Multi-Step Analysis

☐ Enable Multi-Step

Monte Carlo 10 Steps

☑ Transient
☐ AC
☐ DC Sweep
☐ Noise
☐ TF
☐ DCOP

☑ Options

You can also enter the ‘raw’ netlist commands in the schematic using the **Extra Simulation Data** option. The contents of data is output as well as the modes and options setup in the **Simulation Parameters** dialog so you can freely switch between the two methods. The **Simulation Parameters** dialog box does not support sensitivity and pole-zero analysis so these methods must be set up using the Extra Simulation Data.

Running Simulations

Overview

Once an analysis has been set up using the procedures described in this chapter, a simulation would normally be run in *synchronous* mode perhaps by selecting the **Simulation>Simulation Design** menu. In *synchronous* mode, you cannot use any part of the program while the simulation is running.

There are also other methods of running a simulation. You can run a simulation for a netlist directly and you can also run in *asynchronous* mode. These are explained in the following sections.

Starting, Pausing and Aborting Analyses

Starting an Analysis

To start a simulation in normal (synchronous) mode, use the **Simulation>Simulation Design** menu or press the F9 key. A dialog box will show the status of the simulation.

Pausing an Analysis

You can pause the simulation by selecting the **Pause** button on the simulator status dialog box. To restart select the **Resume** button (the **Pause** button changes name when simulation pauses) or the **Simulator>Resume** menu item.

When a simulation is paused, you can carry on using the program as if the simulation had completed. This includes plotting results of the simulation completed so far. If you decide you do not wish to continue the run, there is no need to explicitly abort it. You can just start a new run in the normal way. If you do this you will be asked if you would like to resume the pending run. If you answer 'No', the pending run will be automatically aborted and the new run started.

Aborting an Analysis

There is actually never a need to explicitly abort an analysis. If you decide you do not wish to continue a run, just pause it as described above. Pause is the same as abort except that you have the option to change your mind and restart.

Nevertheless there is an abort facility. Simply select the **Simulator>Abort** menu. When you abort a run, you will not be able to restart it.

There is just one benefit of aborting a run instead of pausing it. When an analysis is aborted, the simulator frees up the memory it needed for the run. Note that this does not happen after a run completes normally. If you need to free up simulator memory after a normal run completes, type Reset at the command line.

Running Analyses in Asynchronous Mode

In *asynchronous* mode, the simulation runs in the background and you are free to carry on using the Pulsonix environment for entering schematics or viewing results from previous analyses. Because, the simulation is running in the background, it is necessary for the simulation process to be detached from the front end environment and for this reason it is not possible to use .GRAPH or fixed probes to plot simulation results during the course of the run. Also you must manually load the simulation data when the run is complete.

Starting an Asynchronous Run

1. Select **Schematic** menu **Simulation>Run Asynchronous...** Note a simulation status box appears similar to the box used for synchronous runs but with an additional **Activity** box at the bottom. Any messages generated by the simulator will be displayed here.
2. When the simulation is complete, you must load the data manually. The name of the file to load will be displayed in the command shell when the simulation starts. Select menu **File>Data>Load Temporary Data…** to load data file. You will be

able to random probe the schematic used to run the analysis in the normal manner once this file is loaded.

Pausing and Aborting Asynchronous Runs

To pause, press the **Pause** button. Note that you can load the data generated so far after pausing the run as described above.

To abort a run, press the **Close** button.

Running an Analysis on a Netlist

You can run an analysis on a netlist created by hand or perhaps with a third party schematic entry program.

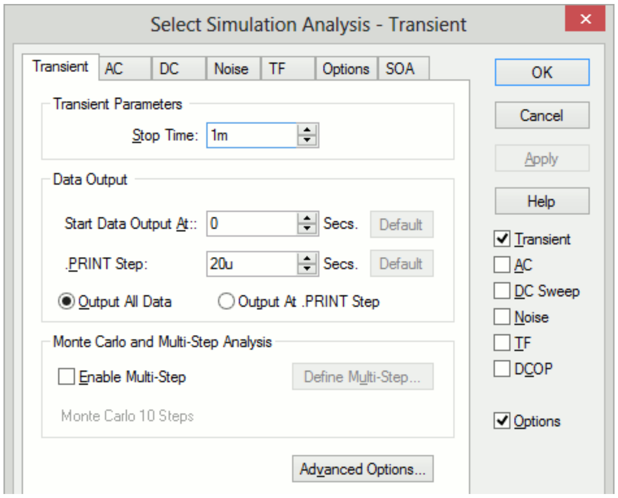
To run a netlist in synchronous mode, select the command shell menu **Simulator>Run Netlist Asynchronous…** then locate the netlist file. See "Running Analyses in Asynchronous Mode" above for further information about running asynchronous analyses.

Transient Analysis

In this mode the simulator computes the behaviour of the circuit over a time interval specified by the stop time. Usually, the stop time is the only parameter that needs specifying but there are a number of others available.

Setting up a Transient Analysis

1. Select menu **Simulation>Simulation Parameters**
2. Select **Transient** check box on the right.
3. Select the **Transient** tab. Enter parameters as described in the following sections.



Transient Parameters
Stop Time: 1m

Data Output
Start Data Output At:: 0 Secs.
.PRINT Step: 20u Secs.
(●) Output All Data ( ) Output At .PRINT Step

Monte Carlo and Multi-Step Analysis
☐ Enable Multi-Step
Monte Carlo 10 Steps

☑ Transient
☐ AC
☐ DC Sweep
☐ Noise
☐ TF
☐ DCOP
☑ Options

Transient Parameters

Enter the stop time as required. Note that the simulation can be paused before the stop time is reached allowing the results obtained so far to be examined. It is also possible to restart the simulation after the stop time has been reached and continue for as long as is needed. For these reasons, it is not so important to get the stop time absolutely right. You should be aware, however, that the default values for a number of simulator parameters are chosen according to the stop time. (the minimum time step for example). You should avoid therefore entering inappropriate values for stop time.

Data Output Options

Sometimes it is desirable to restrict the amount of data being generated by the simulator which in some situations can be very large. You can specify that data output does not begin until after some specified time and you can also specify a time interval for the data.

Output all data/Output at .PRINT step
The simulator generates data at a variable time step according to circuit activity. If **Output all data** is checked, all this data is output. If **Output at .PRINT step** is checked, the data is output at a fixed time step regardless of the activity in the circuit. The actual interval is set by the >PRINT step. This is explained below.

If the **Output at .PRINT step option** is checked, the simulator is forced to perform an additional step at the required interval for the data output. The fixed time step interval data is not generated by interpolation as is the case with generic SPICE and other products derived from it.

Start data output @
No simulation data will be output until this time is reached.

.PRINT step
.PRINT is a simulator command that can be specified in the netlist to enable the output of tabulated data in the list file. See “The Simulator” section for details of .PRINT.

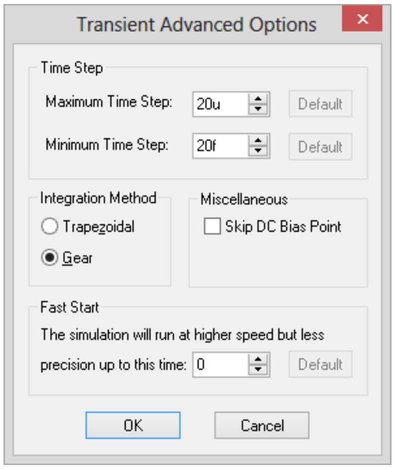
The value specified here controls the interval used for the tabulated output provided by .PRINT but the same value is also used to determine the data output interval if **Output at .PRINT step** is specified. (See above).

Monte Carlo and Multi-step Analysis

See section **Multi-step Analyses** later in this chapter

Advanced Options

The **Advanced Options** button on the **Transient Analysis** dialog opens the dialog below:


Transient Advanced Options

Time Step
Maximum Time Step: 20u
Minimum Time Step: 20f

Integration Method
Trapezoidal [✓ Gear]

Miscellaneous
Skip DC Bias Point ☐

Fast Start
The simulation will run at higher speed but less precision up to this time: 0

Time Step

The simulator always chooses the time step it uses but it will not rise above the maximum time step specified here.

If the simulator needs to use a time step smaller than the minimum specified, the simulation will abort. Reduce this value if the simulation aborts with the message “Time step too small”. This might happen for long runs on circuits that have very small time constraints.

Integration method

Set this to **Gear** if you see an unexplained triangular ringing in the simulation results. Always use **Trapezoidal** for resonant circuits. A full discussion in integration methods is given in the Convergence section of The Simulator chapter.

Skip DC bias point

If checked, the simulation will start with all nodes at zero volts. Note that unless all voltage and current sources are specified to have zero output at time zero, the simulation may fail to coverage if this option is specified.

Fast start

The accuracy of the simulation will be relaxed for the period specified. This will speed up the run at the expense of precision.

This is a means of accelerating the process of finding a steady state in circuits such as oscillators and switching power supplies. It is often of little interest how the steady state is reached so precision can be relaxed while finding it.

Note that the reduced precision can also reduce the accuracy at which a steady state is found and often a settling time is required after the fast start period.

Restarting a Transient Run

After a transient analysis has run to completion, this is it has reached its stop time, it is still possible to restart the analysis to carry on from where it previously stopped. To restart a transient run:

1. Select the menu **Simulation>Restart Transient…** . The restart dialog will be displayed by the simulator.
2. In the **New Stop Time** box enter the time at which you wish the restarted analysis to stop. Press **Ok** to start run.

See Also

".TRAN" in The Simulator Control Reference in the "The Simulator" chapter.

Operating Point

To specify a DC operating point analysis check DCOP. Note that an operating point is performed automatically for all analysis modes and this is only useful if it is the only analysis specified.

Operating point analysis does not have any additional parameters so there is no tab sheet for it.

See Also

".OP" in the Simulator Control Reference in the "The Simulator" chapter

Sweep Modes

Each of the analysis modes DC, AC, AC Noise and Transfer Function are swept. That is they repeat a single analysis point while varying some circuit parameter. There are 6 different sweep modes that can be applied to these analyses. These modes are also used to define multi step analyses which are explained later in this chapter. The 6 modes are:

- Device
- Temperature
- Parameter
- Model parameter
- Frequency (not applicable to DC)
- Monte Carlo

As well as 6 different modes there are 3 different sweep methods.

- Linear
- Decade
- List

Dialog support for the List method is only available for the definition of Multi-step analyses. The simulator also offers an Octal sweep method but this is not supported by the Simulation Parameters Dialog.

Each of the sweep modes is explained in more detail below.

Device Sweep

In this mode the principal value of a single device is swept. The analysis definition must specify the component reference for the device. The following types of device may be used.

Device	**Value swept**
Capacitor	Capacitance
Diode	Area
Voltage controlled voltage source	Gain
Current controlled current source	Gain
Voltage controlled current source	Transconductance
Current controlled voltage source	Transresistance
Current source	Current
JFET	Area
Inductor	Inductance
Bipolar Transistor	Area
Resistor	Resistance
Lossless Transmission Line	Impedance
Voltage source	Voltage
GaAs FET	Area

Temperature

Global circuit temperature is swept

Model Parameter

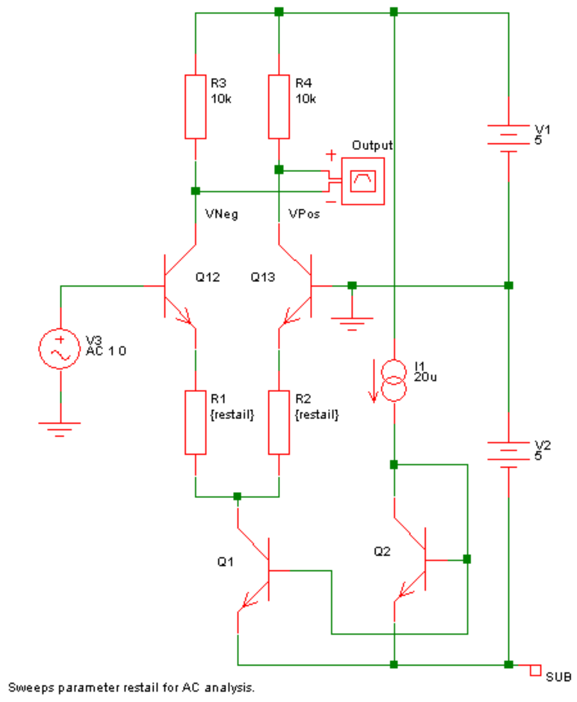
Similar to device sweep except applied to a named model parameter. Both the model name and the parameter name must be specified.

Special Note

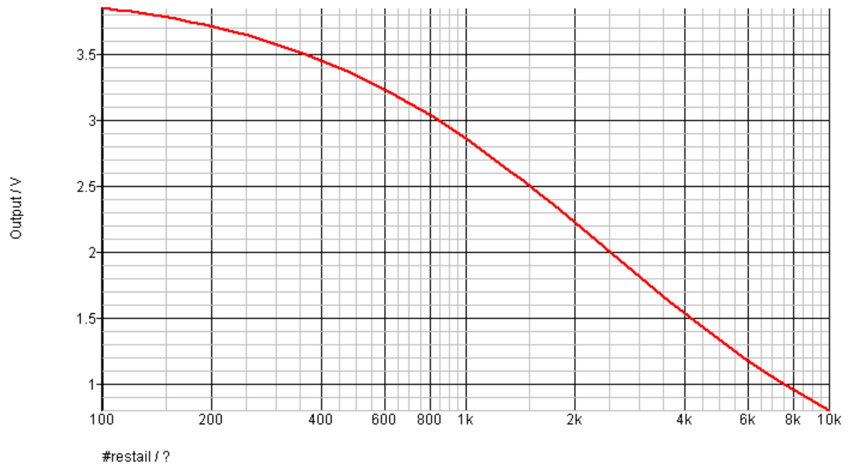
It is recommended that any model parameter being swept is also specified in the .MODEL parameter list. In most cases it isn’t actually necessary but there are a few instances – such as for terminal resistance parameters – where it is necessary.

Parameter

A user named variable that can be referenced in any number of expressions used to define model or device parameters. Here is an example. (See Examples\Spice\Sweep\AC_Param)



This is a simple long tailed pair. The above circuit resistors R1 and R2 have been given the values {restail}. *restail* is a parameter that is swept in an AC sweep to plot the gain of the amplifier vs tail resistance at 100kHz. Here is the result of the run:



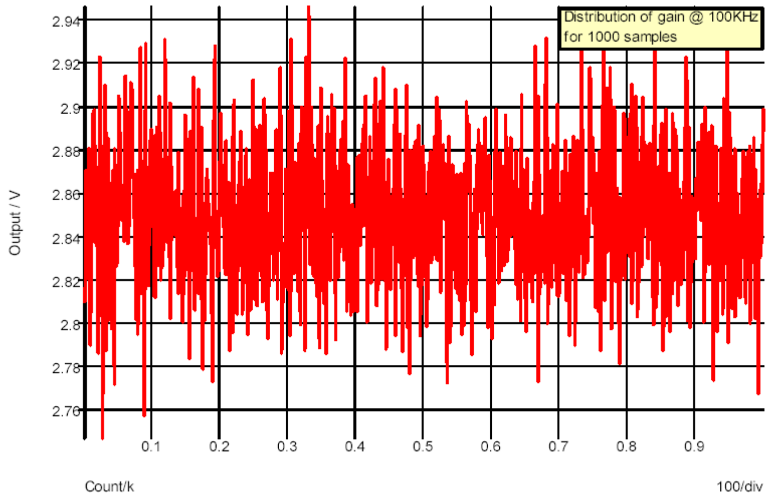
Note that this analysis mode is not available in standard SPICE or the majority of its derivatives. Most offer parameter sweeping, but only for DC analysis.

Frequency

Sweeps frequency for the small signal analysis modes namely AC, AC Noise and Transfer Function. In standard SPICE it is the only sweep mode available for AC and Noise while Transfer Function cannot be swept at all.

Monte Carlo

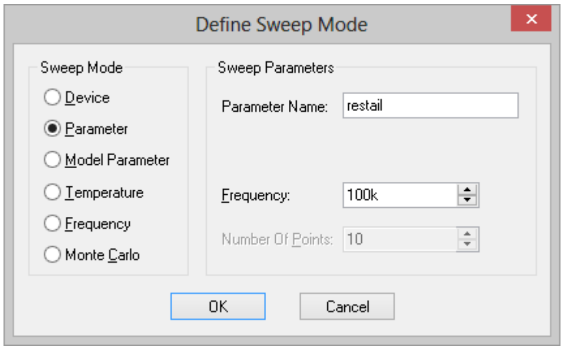
Repeats analysis point for a specified number of times with device tolerances enabled. The following graph show the result for the same circuit as shown above but with restail=1k and with a 1000 point Monte Carlo AC sweep. This run took 0.6 seconds with a 1.5G P4:



The graph shows the variation in gain for 1000 samples. Using the histogram feature a statistical distribution of the above can easily be plotted.

Setting up a Swept Analysis

From the **Simulation** menu, **Simulation Parameters**, on the AC, DC, Noise or Transfer Function analysis dialogs, select the **Define Mode…** button in the **Sweep Parameters** box. This will bring up the following dialog



Sweep Mode: [● Parameter]

Sweep Parameters

Parameter Name: restail

Frequency: 100k

Number Of Points: 10

Select the desired mode on the left then enter the necessary parameters on the right. The parameters required vary according to the mode as follows:

Mode	**Parameters**
Device	Device component reference (e.g.V1) Frequency (AC, Noise and TF only)
Parameter	Parameter name Frequency (AC, Noise and TF only)
Model Parameter	Model name Model parameter name Frequency (AC, Noise and TF only)
Temperature	Frequency (AC, Noise and TF only)
Frequency (not available for DC)	None
Monte Carlo	Number of points Frequency (AC, Noise and TF only)

DC Sweep

Operates in any of the sweep modes described earlier except Frequency. Repeats a DC operating point calculation for the range of circuit parameters defined by the sweep mode.

Setting up a DC sweep

1. Select menu **Simulation|Simulation Parameters**…
2. Select **DC sweep** check box on the right
3. Select **DC** tab at the top. Enter parameters as described in the following sections.

Sweep Parameters

Start value, Stop value
Defines sweep range stop and start values

Points per decade, Number of points
Defines sweep range. The number of points of the sweep is defined per decade

for a decade sweep. For a linear sweep you must enter the total number of points.

Device/Parameter/Model Name
The device name for a device sweep, parameter name for a parameter sweep or the model name for a model parameter sweep may be entered here. It may also be entered in the sweep mode dialog opened by pressing **Define Mode…**

Define Mode…
Sets up desired sweep mode. See “Setting up a Swept Analysis” a few pages back.

Monte Carlo and Multi-step Analysis

See Multi-step Analyses section later in this chapter

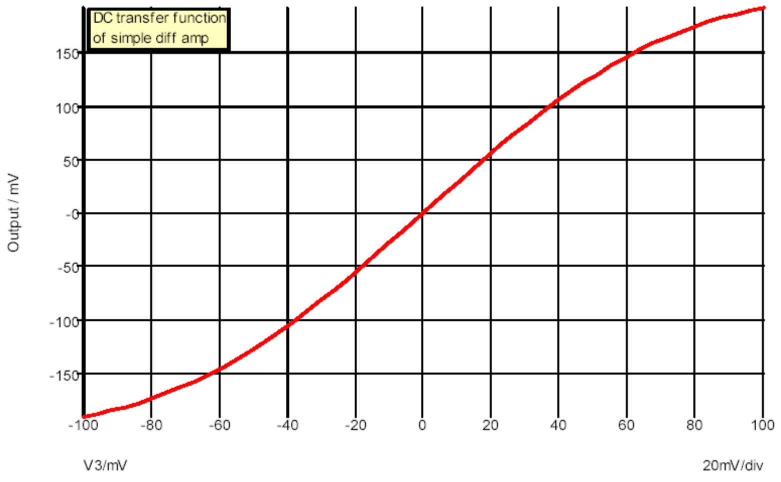
See Also

.DC in the Simulation Control reference section of the “Simulator” chapter

Example

The following is the result of a DC sweep of V3 in the example circuit shown previously with restail set to 1K. Analysis parameters were as follows:

Sweep mode: Device, V3
Sweep range: -0.1 to 0.1, linear sweep with 50 points



AC Sweep

An AC analysis calculates the small signal response of a circuit to any number of user defined inputs. The small signal response is computed by treating the circuit as linear about its DC operating point.

Like DC, AC Noise and Transfer Function analyses, AC analysis is a swept mode and can operate in any of the 6 modes documented in “Sweep Modes” earlier in this chapter. With some of these modes – e.g. sweeping a resistor value – it will be necessary for the

DC operating point to be recalculated at each point while with others – such as frequency sweep – it is only necessary to calculate it at the start of the run.

For AC Analysis to be meaningful there must be at least one voltage or current source on the circuit with an AC specification.

Setting up an AC Sweep

1. Select menu **Simulation|Simulation Parameters…**
2. Select **AC** check box on the right
3. Select **AC** tab at the top. Enter parameters as described in the following sections.

Sweep Parameters

Start value, Stop value
Defines sweep range stop and start values

Points per decade, Number of points
Defines sweep range. The number of points of the sweep is defined per decade for a decade sweep. For a linear sweep you must enter the total number of points.

Define Mode…
Sets up desired sweep mode. See "Setting up a Swept Analysis"

Monte Carlo and Multi-step Analysis

See section on Multi-step Analysis later in this chapter

Data output

Check the **Save all currents** check box to enable the output of all current data including semiconductor devices. If this box is not checked the current into devices such as transistors and diodes will not be saved. In AC analysis the CPU time required to output data can be very significant relative to the solution time, so you should be aware that checking this box may slow down the simulation significantly.

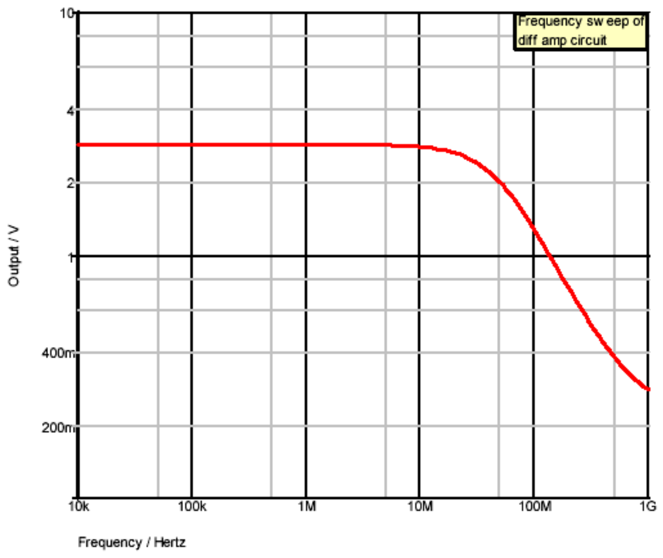
Note that this check box only affects AC analyses.

See Also

".AC" in the Simulation Control Reference section of the "Simulator" section.

Example

Both the examples shown in "Sweep Modes" previously in this chapter are AC analyses. The following is a frequency sweep which is the traditional AC analysis mode. This was performed on the circuit shown with restail = 1k.



Frequency sw eep of diff amp circuit
10
4
2
1
400m
200m
Output / V
10k
100k
1M
10M
100M
1G
Frequency / Hertz

Noise Analysis

Like AC analysis, AC Noise analysis is a small signal mode. The circuit is treated as linear about its DC operating point and the contribution of all noisy devices to a designated output is computed. The total noise at that output is also calculated and optionally the noise referred back to an input source may also be computed.

Like DC, AC and Transfer Function, it is a swept mode and can be operated in any of the 6 modes described in "Sweep Modes" earlier. With some of these modes – e.g. sweeping a resistor value – it will be necessary for the DC operating point to be recalculated at each point while with others – such as frequency sweep – it is only necessary to calculate it at the start of the run.

Note that it is not necessary to apply an AC specification to any source – including the optional input referred source – as it is with standard SPICE and many (if not all) of its derivatives.

Setting up an AC Noise analysis

1. Select menu **Simulation|Simulation Parameters.**
2. Select **Noise** check box on the right
3. Select Noise tab at the top. Enter parameters as described in the following sections.

Sweep Parameters

Start value, Stop value	Defines sweep range stop and start value
Points per decade	Defines sweep range. The number of points Number of points of the sweep is defined per decade for a decade sweep. For a linear sweep you must enter the total number of points.
Define Mode…	Sets up desired sweep mode. See "Setting up a Swept Analysis"

Noise Parameters

Output node	This is compulsory. It is the name of the circuit node as it appears in the netlist. Usually the schematic chooses default node names but we recommend that when running a noise analysis that you assign a user defined signal name to your designated output node
Reference node	Optional. Output noise is referred to this node. This is assumed to be ground if it is omitted.
Source name	Optional. Voltage or current source to which input source is referred. Enter the component reference of either a voltage or current source.

Monte Carlo and Multi-step Analysis

See Multi-step Analysis section later in this chapter.

See Also

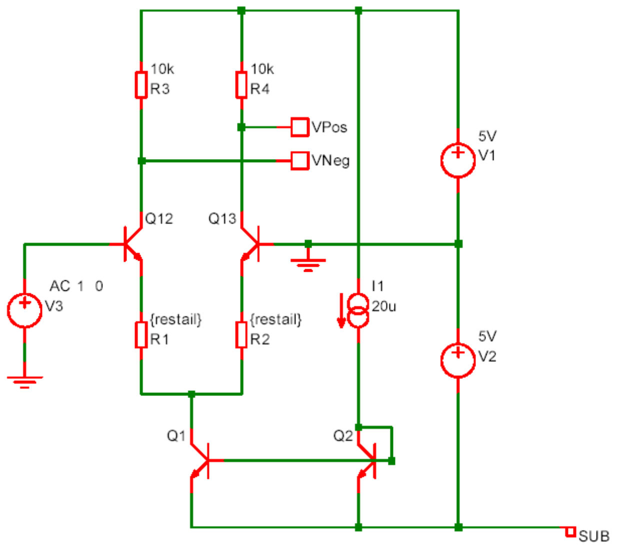
.NOISE in the Simulation Control Reference section of the "Simulator"chapter.

Plotting Results of Noise Analysis

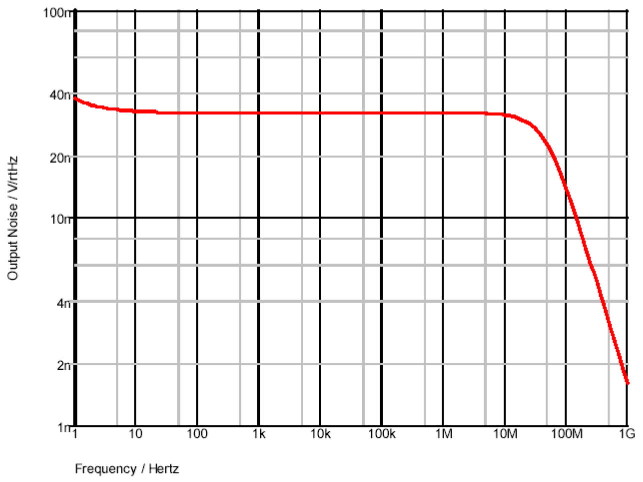
Refer to “Plotting Noise Analysis Results” in the Graphs & Probes chapter.

Example 1

Frequency Sweep



The result of a noise analysis on the above circuit using a frequency sweep

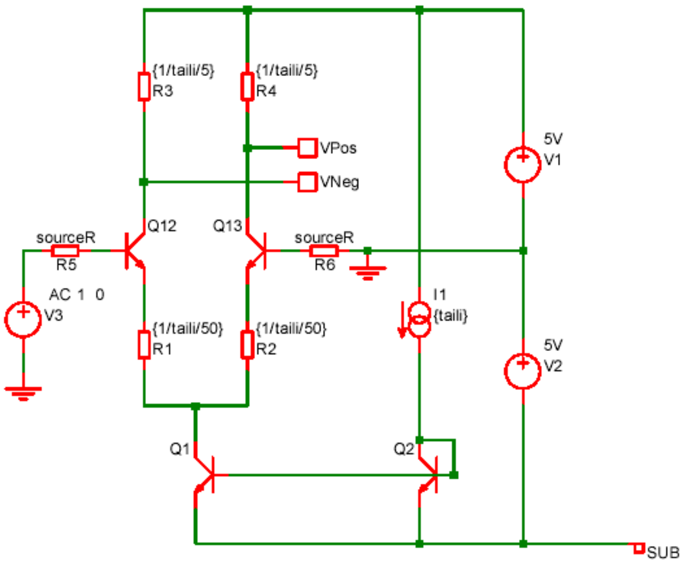


Example 2

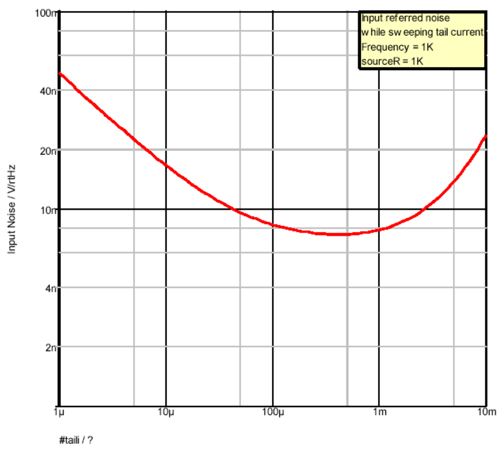
Noise with a Parameter Sweep

In the following circuit we wish to find the optimum value of tail current for a source impedance if 1KΩ To do this we sweep the parameter *taili* which is used to set the current as well as the values for R1, R2, R3 and R4. As can be seen from the graph about 300µA would seem to be best. The noise analysis was setup with the following parameters:

Sweep parameter *taili* from 1µ to 10m, 25 points per decade
Output node: VPos
Reference node: VNeg
Input source: V3
sourceR = 1000 (set with .PARAM control)



The result:



Transfer Function

Transfer function analysis is similar to AC analysis in that it performs a swept small signal analysis. However, whereas AC analysis calculates the response at any circuit node from a (usually) single input source, transfer function analysis calculates the individual responses from each source in the circuit to a single specified output node. This allows, for example, the series mode gain, common mode gain and power supply rejection of an amplifier to be measured in one analysis. The same measurements could be performed using AC analysis but several of them would need to be run. Transfer function mode also calculates output impedance or admittance and, if an input source is specified, input impedance.

Setting up a Transfer Function Analysis

1. Select menu **Simulation|Simulation Parameters**…
2. Select **Transfer Func** check box on the right
3. Select **TF** tab at the top. Enter parameters as described in the following sections.

Sweep Parameters

Start value, Stop value
Defines sweep range stop and start values

Points per decade, Number of points
Defines sweep range. The number of points of the sweep is defined per decade for a decade sweep. For a linear sweep you must enter the total number of points.

Define Mode…
Sets up desired sweep mode. See “Setting up a Swept Analysis” earlier in this chapter.

Transfer Function Parameters

Voltage/Current
Specify whether the output is a node voltage or device current.

Output node/Output source
This is compulsory. If voltage mode is selected it is the name of the circuit node to which the gain of all circuit sources will be calculated. It is the node name as it appears in the netlist. Usually the schematic chooses default net names but we recommend that when running a transfer function analysis that you assign a user defined name to your designated output node.

If current mode is selected it is the name of a voltage source through which the output current is measured. The simulation will calculate the gain for every circuit source to this current.

Reference node
Optional and only available in voltage mode. Output voltage is referred to this node. This is assumed to be ground if it is omitted.

Source name
Optional. Input impedance to this source will be calculated if specified.

Monte Carlo and Multi-step Analysis

See Multi-step Analyses section later in this chapter.

See Also

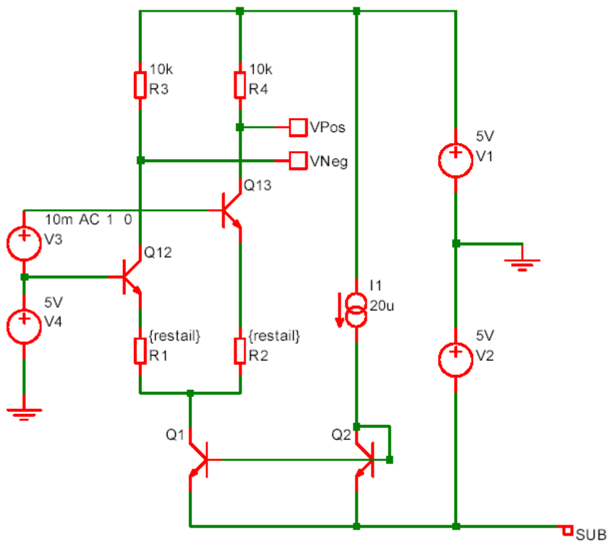
“.TF” in the Simulator Control Reference in the “Simulation” chapter

Plotting Transfer Function Analysis Results

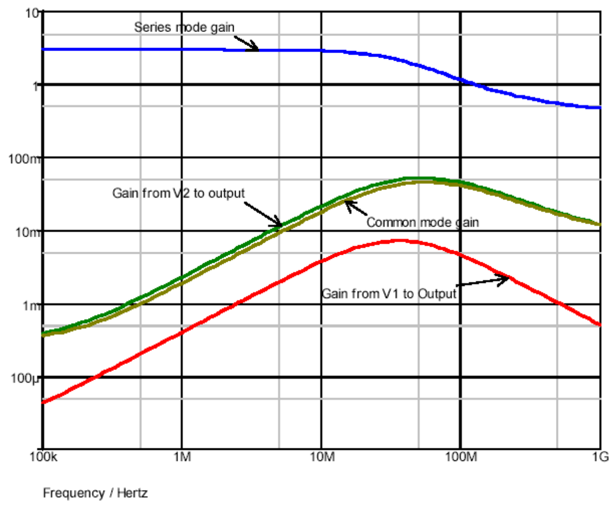
See “Plotting Transfer Function Analysis Results” in the Graphs and Probes chapter

Example

Perform transfer function frequency sweep on the following circuit.



The results:



All of the above waveforms were created with a single analysis.

Pole-zero

Pole zero analysis is a small signal analysis mode that – as the name implies – locates the poles and zeros of a circuit. Note that circuits containing transmission lines or any of the Philips compact models are not supported.

IMPORTANT Pole-zero analysis is an unsupported mode. This means that we cannot provide assistance in its use nor will we be able to resolve any problems found with it.

Setting up a Pole-zero Analysis

Add a control of the following form using the **Extra Simulation Data** option on the **Schematics Simulation** menu.

.PZ *N1 N2 N3 N4* CUR|VOL POL|ZER|PZ

Where N1, N2 are the input nodes and N3, N4 are the output nodes. CUR means the transfer function is of the type (output voltage)/(input current) while VOL means it is (Output voltage)/(input voltage). Usually the last parameter would be PZ which instructs the simulator to find both poles and zeros. The alternatives instruct it to find one or the other. This may be used if the simulator aborts because it didn't converge on poles or on zeros, at least it can be instructed to find the other.

Viewing Results

Select the command shell menu Simulator|List Pole-zero results. The poles and zeros will be listed in complex form.

Example

An example circuit that is already setup is provided in Examples\Spice\Pole-zero\simple_amp.sxsch. Also provided is another circuit containing a Laplace block defined from the results of the pole-zero analysis. This circuit can be found at Examples\Spice\Pole-zero\verify_pz.sxsch. The example demonstrate a method of verifying the results and also an application for pole-zero analysis. The application is a method of modelling a complex circuit as a small signal block. First run a pole-zero analysis to locate the poles and zeros then build the Laplace transform from them. The Laplace transform can then be entered into the Laplace block. Note that pole-zero analysis does not provide the gain of the circuit. This will need to be evaluated separately, perhaps using transfer function analysis.

Sensitivity

This control instructs the simulator to perform a DC sensitivity analysis. In this analysis mode, a DC operating point is first calculated then the linearised sensitivity of the specified circuit voltage or current to every model and device parameter is evaluated. The results are output to the netlist .OUT file by default but can be changed with SENSFILE option) and they are also placed in a new data group. The latter allows the data to be viewed in the message window (type Display) at the command line and can also be accessed from scripts for further analysis.

IMPORTANT Sensitivity analysis is an unsupported mode. This means that we cannot provide assistance in its use nor will we be able to resolve any problems found with it.

Setting up a Sensitivity Analysis

Add a control of the following form using the **Extra Simulation Data** option on the **Schematic Simulation** menu.

.SENS V(*nodename* [,*refnodename*])| I(*sourcename*)

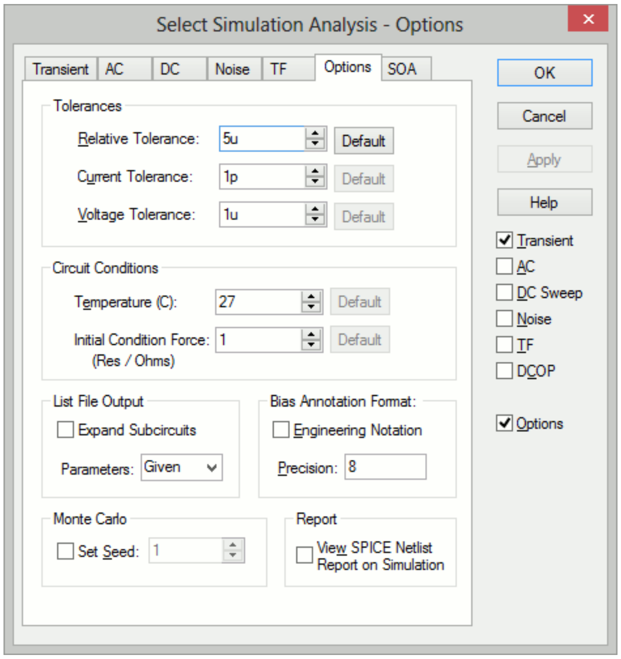
nodename	Output node to which sensitivities are calculated
refnodename	Reference node. Ground if omitted
sourcename	Voltage source to measure output current to which sensitivities are calculated

Simulator Options

The simulator features a large number of option settings although, fortunately, the vast majority can be left at their default values for nearly all applications. A few option settings can be set via the **Simulation Parameters** dialog box and these are described in the following sections. The remainder can be controlled using the simulator's .OPTIONS control details of which may be found in the Simulator Control Reference section in the "Simulator" chapter.

Setting Simulator Options

1. Select menu **Simulation Parameters**….
2. Select **Options** tab. The following will be displayed:
3. Check the **Options** box on the right hand side of the dialog for these options to be included in the netlist.



Select Simulation Analysis - Options

Tolerances
- Relative Tolerance: 5u [Default]
- Current Tolerance: 1p [Default]
- Voltage Tolerance: 1u [Default]

Circuit Conditions
- Temperature (C): 27 [Default]
- Initial Condition Force: (Res / Ohms) 1 [Default]

List File Output
- ☐ Expand Subcircuits
- Parameters: Given

Bias Annotation Format:
- ☐ Engineering Notation
- Precision: 8

Monte Carlo
- ☐ Set Seed: 1

Report
- ☐ View SPICE Netlist Report on Simulation

- ☑ Transient
- ☐ AC
- ☐ DC Sweep
- ☐ Noise
- ☐ TF
- ☐ DCOP
- ☑ Options

Tolerances

Relative Tolerance	Controls the overall accuracy of the simulation. The default value is 0.001 and this is adequate for most applications. If you are simulating oscillator circuits it is recommended to reduce this to 0.0001 or lower. Increasing this value will speed up the simulation but often degrades accuracy to an unacceptable level.
Current Tolerance	Sets the minimum tolerance for current. It may be beneficial to increase this for circuits with large currents.
Voltage Tolerance	Sets the minimum tolerance for voltage. It may be beneficial to increase this for circuits with large voltages.

Circuit Conditions

Temperature	Circuit temperature in °C.
Initial Condition Force Resistance	Initial conditions apply a voltage to a selected node with a force resistance that defaults to 1Ω This option allows that force resistance to be changed.

List File Output

Expand subcircuits	If checked, the listing of expanded subcircuits will be output to the list file. This is sometimes useful for diagnosing problems.
Parameters	Controls the level of model and device parameter output to the list file. Options are:
None	No Output
Brief	Only values defined by an expression are output
Given	The default. Values that are explicitly defined are output
Full	All parameter values are output including defaults

Monte Carlo

Seed
Seed for pseudo random number generator is used to generate random numbers for tolerances. See Multi-step Analyses section below.

Bias Annotation Format

Use to change the format of the voltages and currents displayed in the circuit when using Bias annotation marker components in the schematic.

Report

Check the View Spice netlist report to display a schematic report on generating the netlist.

Multi-step Analyses

Multi-step analyses are not available with all versions of the product.

The analysis modes, Transient, AC, DC, Noise and Transfer Function can be setup to automatically repeat while varying some circuit parameter. Multi-step analyses are defined using the same 6 sweep modes used for the individual swept analyses. The 6 modes are briefly described below. Note that Monte Carlo analysis is the subject of a whole chapter see “Monte Carlo Analysis is described in more detail in the “Simulator! chapter.

- Device. Steps the principal value of a device. E.G. the resistance of a resistor, voltage of a voltage source etc. The component reference of the device must be specified.
- Model parameter. Steps the value of a single model parameter. The name of the model and the parameter name must be specified.
- Temperature. Steps global circuit temperature.
- Parameter. Steps a parameter that may be referenced in an expression.
- Frequency. Steps global frequency for AC, Noise and Transfer Function analysis.

- Monte Carlo. Repeats run a specified number of times with tolerances enabled.

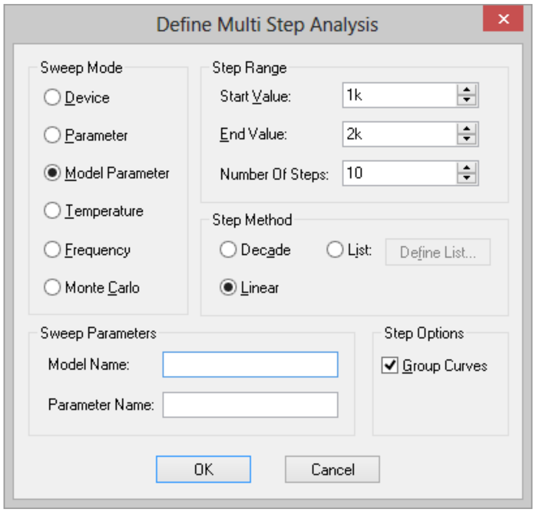
As well as 6 different modes there are 3 different sweep methods which an be applied to all modes except Monte Carlo. These are:

- Linear
- Decade
- List

The simulator also offers an Octal sweep method but this is not supported by the Simulation Parameters Dialog.

Setting up a Multi-step Analysis

From the **Simulation** menu and **Simulation Parameters**, define **Transient**, **AC**, **DC**, **Noise** or **Transfer Function (TF)** as required, then check **Enable Multi-step** and press the **Define Multi-step** button. For transient/DC analysis you will see the following dialog box. Other analysis modes will be the same except that the frequency radio button will be enabled.



Sweep Mode: Device / Parameter / [✓ Model Parameter] / Temperature / Frequency / Monte Carlo

Start Value: 1k
End Value: 2k
Number Of Steps: 10

Step Method: Decade / List / [✓ Linear]

Model Name:
Parameter Name:

Group Curves [✓]

Enter parameter as described below. Only the boxes for which entries are required will be enabled. In the above example, only the **Number of steps** box is enabled as this is all that is required for Monte Carlo mode.

Sweep Mode

Choice of 6 modes as described above.

Step Parameters

Define range of values. If **Decade** is selected you must specify the number of steps per decade while if **Linear** is specified, the total number of steps must be entered. If **List** is selected, you must define a sequence of values by pressing **Define List**… .

Group Curves	Curve traces plotted from the results of multi-step analyses will be grouped together with a single legend and all in the same colour. For Monte Carlo analysis, this is compulsory; for other analyses it is off by default.

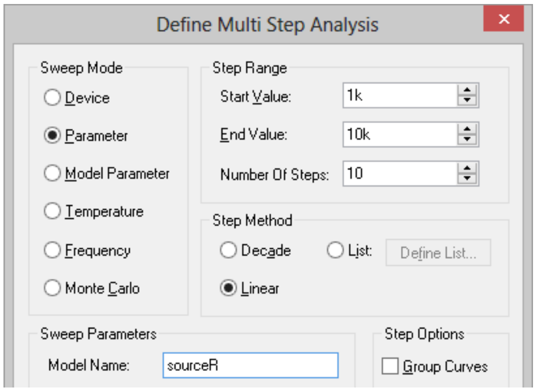
Sweep Parameters

The parameters required vary according to the mode as follows:

Mode	**Parameters**
Device	Device name (e.g. V1)
Parameter	Parameter name
Model Parameter	Model name Model Parameter name
Temperature	None
Frequency (not DC or transient)	None
Monte Carlo	None

Example 1

Refer to circuit in the Noise Analysis section, Example 2 a few pages back. In the previous example we swept the tail current to find the optimum value to minimise noise for a 1K source resistance. Here we extend the example further so that the run is repeated for a range of source resistances. The source resistance is varied by performing a parameter step on *sourceR*. Here is what the dialog settings are for the multi-step run:


Define Multi Step Analysis

Sweep Mode: ( ) Device [✓ Parameter] ( ) Model Parameter ( ) Temperature ( ) Frequency ( ) Monte Carlo

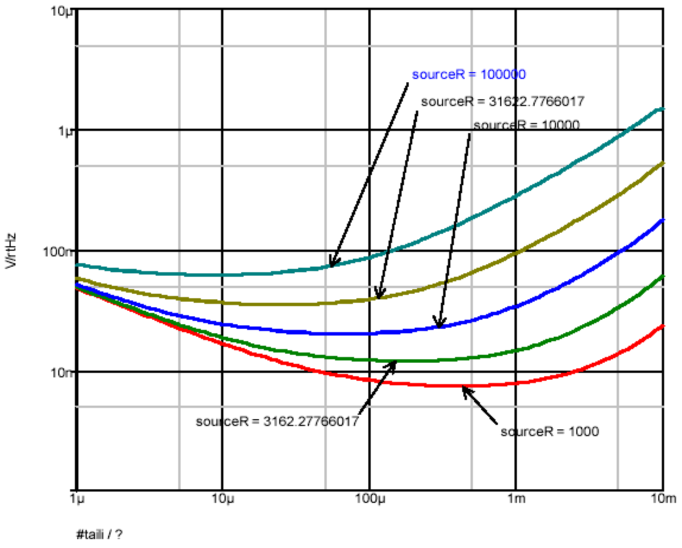
Step Range
Start Value: 1k
End Value: 10k
Number Of Steps: 10

Step Method: ( ) Decade ( ) List [✓ Linear]

Sweep Parameters
Model Name: sourceR

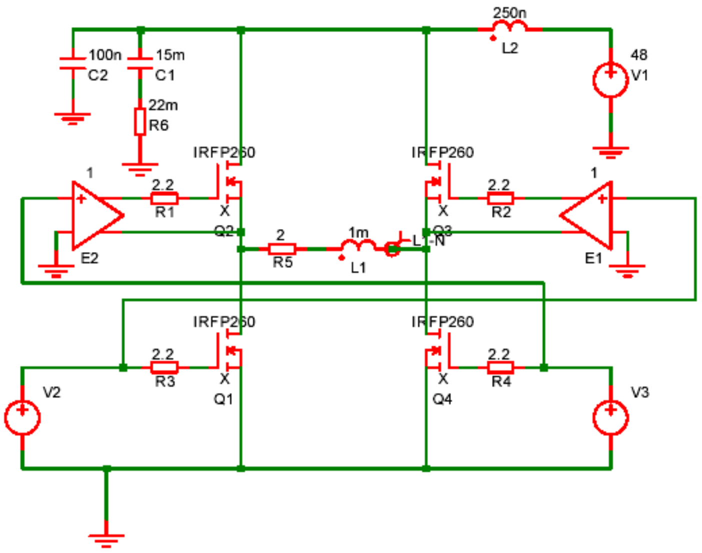
Step Options
☐ Group Curves

This does a decade sweep varying *sourceR* from 1K to 100k with 2 steps per decade. This is the result we get:



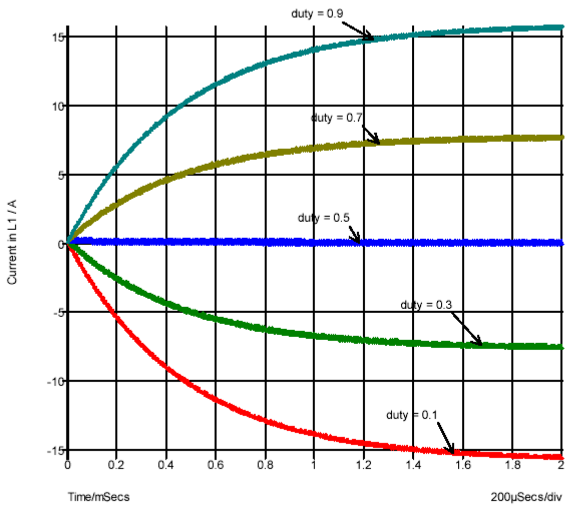
Example 2

The following circuit is a simple model of a full bridge switching amplifier used to deliver a controlled current into an inductance.



Sources V2 and V3 have been defined to be dependent on a parameter named *duty* which specifies the duty cycle of the switching waveform. See Examples\Spice\Bridge\Bridge.sxsch.

This was setup to perform a multi step analysis with the parameter *duty* stepped from 0.1 to 0.9. This is the result:



Safe Operating Area (SOA) Testing

SOA Overview

Safe Operating Area (SOA) testing is a feature that can be used with DC or Transient analyses. With SOA testing, you can set maximum and minimum limits for any simulation quantity and the simulator will display when those limits are violated. It is intended to check that semiconductor devices are operating within the manufacturer's design limits.

To use SOA testing, you must do two things:

1. Define the SOA limits for the models or devices you are using.

2. Enable and configure SOA testing

Item 1. above is covered in detail in the *Pulsonix Spice Device Reference Manual* - see section .SETSOA and also the LIMIT parameter described in the section titled .MODEL. up simple limit tests using some simple schematic symbols is described below.

Defining Simple Limit Tests

Parts

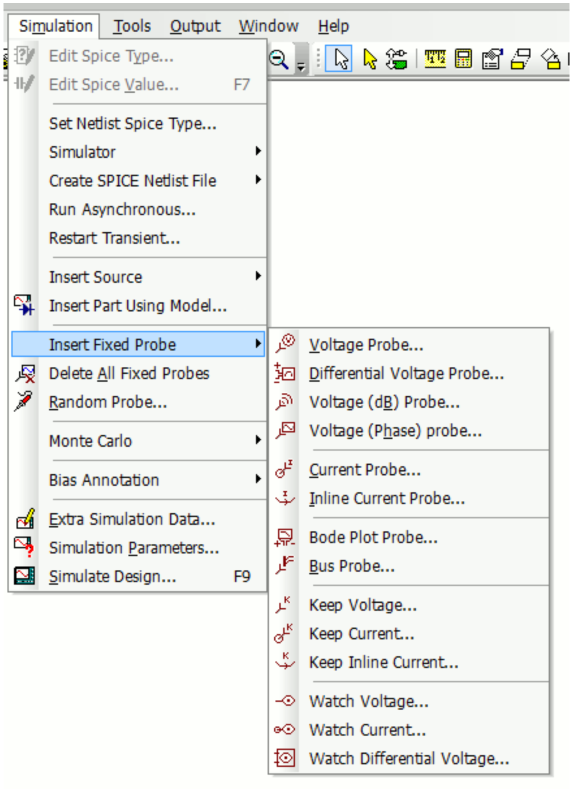
Parts with built-in functions are available to support SOA testing: **Watch - Current**, **Watch - Voltage** and **Watch - Differential Voltage**. These are used to create Watch parts for simple SOA limit testing.

1. Over and under voltage on a single node

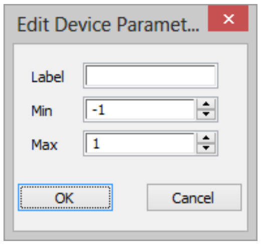
2. Over and under current on a single device pin

3. Over and under differential voltage on a node pair

These three Parts have been added to the Spice Parts library. These are also available from the **Insert Fixed Probes** menu and on the **Miscellaneous** Parts popup toolbar.



When editing Parts created with the SOA functionality, when using <**F7**> key to edit the values, the generic **Edit Device Parameters** dialog is displayed for you to supply the **Min** and **Max** limits and a **Label** to mark any SOA violations.


Edit Device Paramet...

Label

Min -1

Max 1

The **minimum limit**. Use a large negative number (e.g. -1e100) if you don't wish to specify a minimum limit.

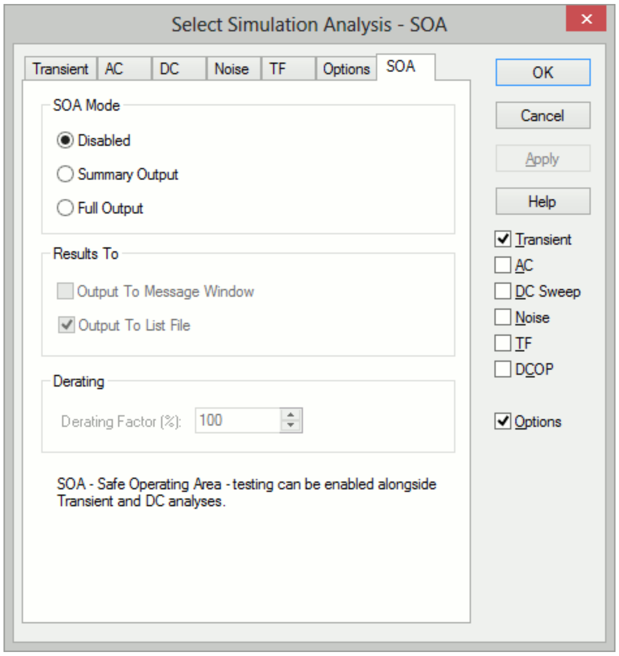
The **maximum limit**. Use a large positive number (e.g. 1e100) if you don't wish to specify a maximum limit.

A **label**. The default value is %REF% that will resolve to the device's component reference. You can enter any literal value instead.

▶ To set up SOA testing

1. Select **Simulator** menu and **Simulation Parameters**.

2. Select the **SOA** tab.



SOA Mode
[● Disabled]
○ Summary Output
○ Full Output

Results To
☐ Output To Message Window
☑ Output To List File

Derating
Derating Factor (%): 100

SOA - Safe Operating Area - testing can be enabled alongside Transient and DC analyses.

☑ Transient
☐ AC
☐ DC Sweep
☐ Noise
☐ TF
☐ DCOP
☑ Options

3. Under **SOA** mode choose either **Summary Output** or **Full Output**. In summary output mode, only the first violation for each SOA device will be reported. In full output mode, all violations are reported.

4. In **Results to:** choose where you would like the results reported. Note that writing results to the message window is a time consuming operation and it is recommended that you should not select this if you are expecting a large number of violations.

Running Simulation

Run the simulation in the normal way. If there are any violations, the results will be reported in the location or locations specified in the Results to: section.

Advanced SOA Limit Testing

The simulator control .SETSOA allows much more sophisticated definitions for SOA limits. In particular, you can define limits for all devices belonging to a specified model. Suppose that you are using a BJT model that has a Vcb limit of 15V. While you could place a differential voltage watch device across each instance of this model, this would be time consuming and error prone. Instead, you can define a single .SETSOA control that refers to the model name of the device. The simulator will then automatically set up the limit test for every instance of that model.

You would usually enter a .SETSOA control in the schematic editor’s <**F11>** window. For details of the .SETSOA syntax, please refer to the *Pulsonix Spice Device Reference Manual* section titled .SETSOA.

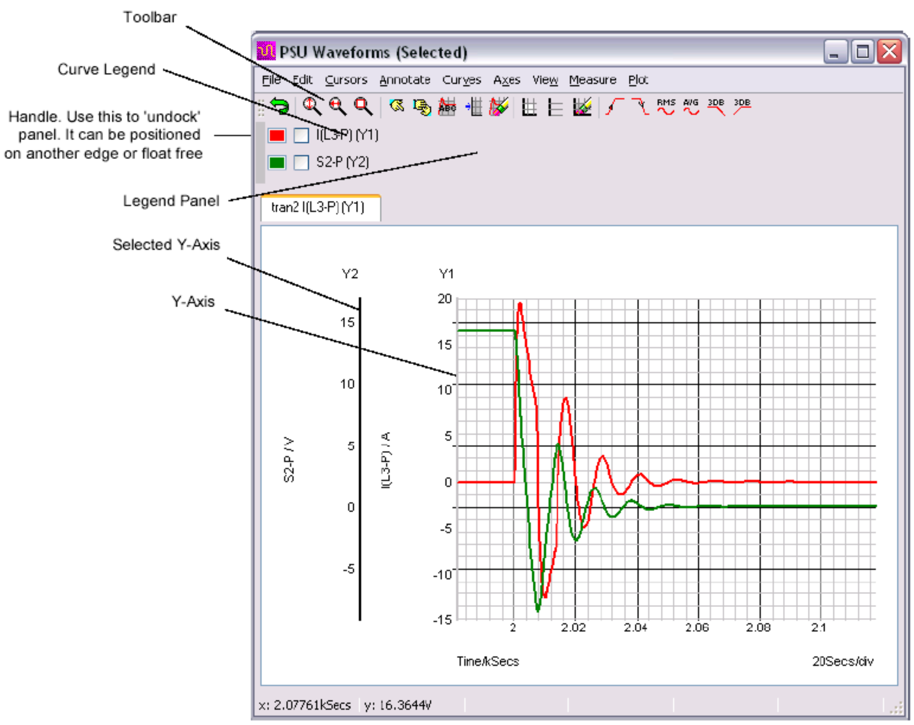
Chapter 7. Graphs and Probes

Overview

The basics of how to create graphs of your circuit's signals were explained in Chapter 3. This chapter provides a full reference on all aspects of probing and creating graphs.

Elements of the Graph Window

Main Window

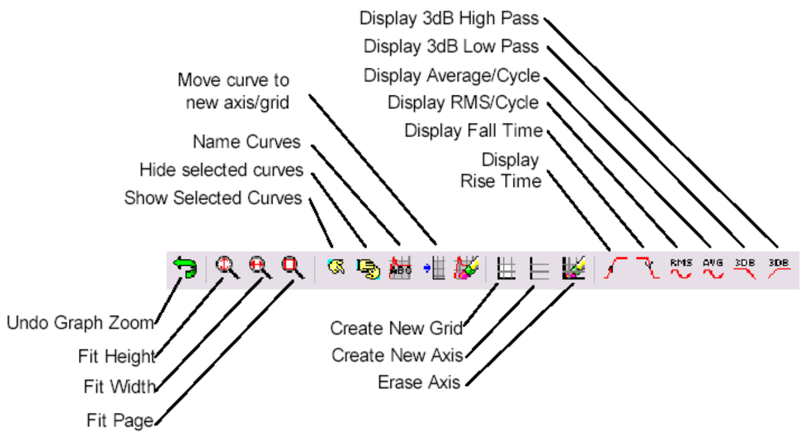

Toolbar
Curve Legend
Handle. Use this to 'undock' panel. It can be positioned on another edge or float free
Legend Panel
Selected Y-Axis
Y-Axis
PSU Waveforms (Selected)
File Edit Cursors Annotate Curves Axes View Measure Plot
I(L3-P) (Y1)
S2-P (Y2)
tran2 I(L3-P) (Y1)
Y2
Y1
S2-P / V
I(L3-P) / A
Time/kSecs
20Secs/div
x: 2.07761kSecs y: 16.3644V

Note: the legend panel can be resized. With most systems a ‘resize’ handle is clearly visible on the lower edge but with the standard Windows XP theme - e.g. the above picture - this is not the case.

Windows and Tabbed Sheets

Normally new graphs are created within the same window as a "Tabbed Sheet". A row of tabs will appear at the top of the graph window allowing you select which graph you wish to view. You can also create a new graph window using **New Graph Window** from the Schematic's **Random Probe** shortcut menu. This will create an empty window to which you may add new graphs.

Graph Toolbar



The above shows the function of each of the buttons on the **graph toolbar**. These are referred to in the following sections.

Plotting Curves

Probes: Fixed vs. Random Probes

Much of this section and some of the next have already been covered previously. It is repeated here for convenience.

Pulsonix Spice provides two approaches to creating plots of simulated results from a Schematic.

The first approach is to fix voltage or current probes to the Schematic before or during a run. Pulsonix Spice will then generate graphs of the selected voltages and/or currents automatically. Normally the graphs for fixed probes are opened and updated while the simulator is running. The probes have a wide range of options which allow you to specify - for example - how the graphs are organised and when and how often they are updated. These probes are known as "Fixed Probes".

The second approach is to randomly probe the circuit after the run is complete. (You can also do this during a run by pausing first). With this approach, the graph will be created as you point the probe but will not be updated on a new run. These probes are known as "Random Probes".

You do not need to make any decisions on how you wish to probe your circuit before starting the run. You can enter a circuit without any fixed probes, run it, then randomly probe afterwards. Alternatively, you can place - say - a single fixed probe on an obvious point of interest, then randomly probe to investigate the detailed behaviour of your circuit.

There are currently 8 types of fixed probe to suit a range of applications. The random probing method allows you to plot anything you like including device power, FFT's, arbitrary expressions of simulation results and X-Y plots such as Nyquist diagrams. It *is* possible to set up fixed probes to plot arbitrary expressions of signals but this requires manually entering the underlying simulator command, the .GRAPH control. There is no direct Schematic support for this.

Fixed Probes

There are eight types of fixed probe:

1. Current. Plots the current in a device pin.
2. Voltage. Plots the voltage on a net.
3. Differential voltage. Plots the voltage difference between two points.
4. dB Probe. Plots the voltage in dBs.
5. Phase Probe. Plots the voltage as phase.
6. Inline Current. Two terminal device that plots the current flowing through it.
7. Bode Plot. Plots db and phase of vout/vin. Connect to input and output of a circuit to plot its gain and phase.
8. Bus Plot. Plots bus signals in “Logic Analyser” style.

They are simply Schematic parts with special spice properties. When you place a fixed probe on the Schematic, the voltage or current at the point where you place the probe will be plotted each time you run the simulation. The probes have a wide range of options which can be set by selecting the probe then pressing **F7**. These options are covered in detail later in this chapter.

You can place these fixed probe parts on a Schematic using any of three methods as follows:

1. Use the **Insert Fixed Probe** option from the **Simulation** menu.
2. From the **Parts** toolbar use the **Probes** popup toolbar.
3. Use the **Part Browser** from the **View** menu, open **Probes** category.

Fixed Current Probes

Current probes must be placed directly over a Component pin. They will have no function if they are not.

Fixed Voltage Probes

Voltage probes must be placed directly over a connection. They will have no function if they are not.

Fixed Differential Voltage Probes

Attach it’s pins to the two nets you wish to plot the voltage difference between.

Inline Current

Insert into the connection on the net where you wish to plot.

Bode Plot

Connect to input and output nets of the circuit.

Bus Plot

Place over the bus segment of interest.

Keep Probe

Place on connection to limit the amount of data output by the simulator.

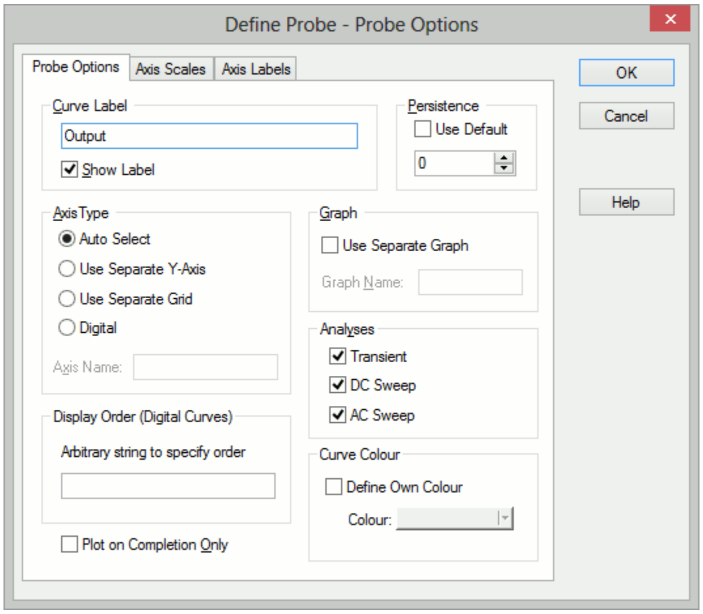
Watch Probe

Used on a net for SOA Testing.

Fixed Probe Options

Fixed probes have a wide range of options allowing you to customise how you want the graph plotted. For many applications the default settings are satisfactory. In this section, the full details of available probe options are described.

To change a probe's options, select it then press <**F7**> or use shortcut menu **Edit Spice Value/Model...**. The following dialog will be displayed:



Curve Label: Output
[✓] Show Label

Persistence
☐ Use Default
0

Axis Type
[✓ Auto Select]
Use Separate Y-Axis
Use Separate Grid
Digital
Axis Name:

Graph
☐ Use Separate Graph
Graph Name:

Analyses
☑ Transient
☑ DC Sweep
☑ AC Sweep

Display Order (Digital Curves)
Arbitrary string to specify order

Curve Colour
☐ Define Own Colour
Colour:

☐ Plot on Completion Only

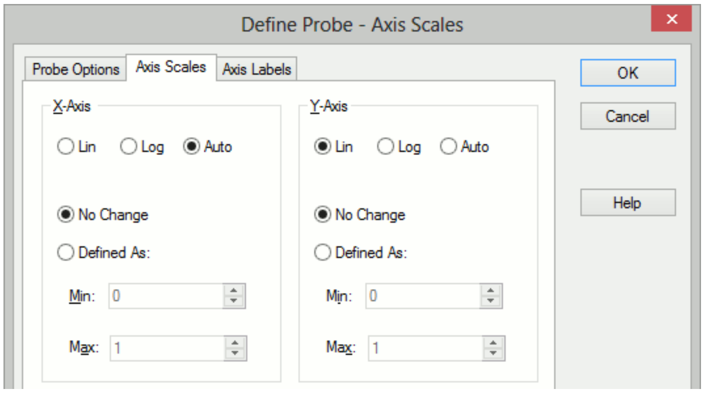
The elements of each tabbed sheet are explained below.

Probe Options Sheet

Curve Label	Text that will be displayed by the probe on the Schematic and will also be used to label resulting curves. Use **Show Spice Value** to toggle display of this in the schematic.
Persistence	If non-zero, curves created from the curve will have a limited lifetime. The persistence value is the number of curves from a single probe that will be displayed at once, the oldest being automatically deleted. If set to zero, they will never be deleted.
Axis Type	Specifies the type of y-axis to use for the curve.
Auto Select	Will use main y-axis unless its unit are incompatible. E.g. plotting a current but the graph already has a voltage. In that case, a new y-axis will be created alongside the main one. If the signal is digital, a digital axis (see below) will be used for this probe.
Use Separate Y-axis	Will always use its own separate y-axis. If you specify this you can optionally supply an axis name. The value of the axis name is arbitrary and is used to identify the axis so that multiple fixed probes can specify the same one. This name is not used as a label for display purposes but simply as a means of identification. Axes can be labelled "Axis Labels" sheet. See below.
Use Separate Grid	Similar to above but uses a new grid that is stacked on top of main grid.
Digital	Use a digital axis. Digital axes are placed at the top of the window and are stacked. Each one may only take a single curve. As their name suggests, they are intended for digital traces but can be used for analog signals if required.
Graph	Check "Use separate graph" if you wish a new graph sheet to be used for the probe. You may also supply a graph name. This works in the same way as axis name (see above). It is not a label but a means of identification. Any other probes using the same graph name will have their curves directed to the same graph sheet.
Colour	If Use default is checked, the colour will be chosen automatically in a manner that tries to minimise duplicate colours on the same graph. Alternatively uncheck this box then press Edit... to select a colour of your choice. In this case the trace will always have the same colour.

Analyses	Specifies for which analyses the probe is enabled. Note, other analysis modes such as noise and sensitivity are not included because these don't support Schematic cross probing of current or voltage.
Plot on completion only	If checked the curve will not be created until the analysis is finished. Otherwise they will be updated at an interval specified in the Simulator's Options dialog File|Options|General...

Axis Scales Sheet


Define Probe - Axis Scales

Probe Options | Axis Scales | Axis Labels

X-Axis
Lin Log [✓ Auto]
[✓ No Change]
Defined As:
Min: 0
Max: 1

Y-Axis
[✓ Lin] Log Auto
[✓ No Change]
Defined As:
Min: 0
Max: 1

X-Axis/Y-Axis

Lin/Log/Auto	Specify whether you want X-Axis to be linear or logarithmic. If "Auto" is selected, the axis (X or Y) will be set to log if the x values are logarithmically spaced. For the Y-axis it is also necessary that the curve values are positive for a log axis to be selected.
No Change	Keep axis scales how they are. Only relevant if adding to an existing graph.
Defined	Set axis to scales defined in Min and Max boxes

Axis Labels Sheet

This sheet has four edit boxes allowing you to specify, x and y axis labels as well as their units. If any box is left blank, a default value will be used or will remain unchanged if the axis already has a defined label.

Adding Fixed Probes After a Run has Started

When you add a fixed probe after a run has started, the graph of the probed point opens soon after resuming the simulation. This doesn't apply to differential voltage probes. To do this:

1. Pause simulation.
2. Place a probe on the circuit in the normal way.

3. Resume simulation

Changing Update Period and Start Delay

The update period of all fixed probes can be changed from the Options dialog box. Select Command Shell menu **File|Options|General...** and click on "Graph/Probe/Data analysis". In the "Probe update times/seconds" box there are two values that can be edited. "Period" is the update period and "Start" is the delay after the simulation begins before the curves are first created.

Random Probes

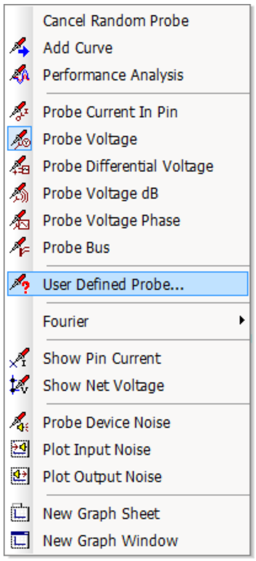
General Behaviour

A wide range of functions are available from within the Schematic's **Random Probe** mode. With a few exceptions detailed below, all random probe functions have the following behaviour.

1. If there are no graph windows open, one will be created.
2. If a graph window is open and the currently displayed sheet has a compatible x-axis to what you are probing, the new curve will be added to that sheet. E.g. if the currently displayed graph is from a transient analysis and has an x-axis of "Time", and you are also probing the results of a transient analysis, then the new curve will be added to the displayed graph. If, however the displayed curve was from an AC analysis, its x-axis would be frequency which is incompatible. In this case a new graph sheet will be created for the new curve.

Probe Functions

Once the simulation has been run, you may select **Random Probe** from the **Simulation** menu. You can probe, voltage, current, differential voltage, device power, dB, phase, Nyquist diagrams and much more. Once the **Random Probe** mode has been entered, you will see the probe cursor and you can right click and select a probe type from the menu:

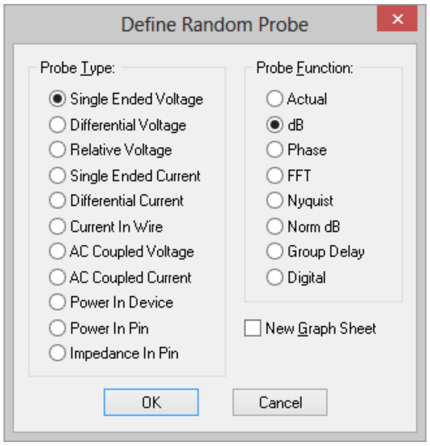


Standard Probes

The first five options in the second section of the menu are the standard probe types that match the fixed probes mentioned earlier. Pick nets when using the **Voltage** probes and device pins when using the **Current** probes. The **Differential Voltage** probe asks for you to select two nets before plotting the simulation curve.

Advanced Probe Functions

Select **User Defined Probe** option to access many more probing functions selectable from a dialog as follows:


Define Random Probe

Probe Type:
- ● Single Ended Voltage
- ○ Differential Voltage
- ○ Relative Voltage
- ○ Single Ended Current
- ○ Differential Current
- ○ Current In Wire
- ○ AC Coupled Voltage
- ○ AC Coupled Current
- ○ Power In Device
- ○ Power In Pin
- ○ Impedance In Pin

Probe Function:
- ○ Actual
- ● dB
- ○ Phase
- ○ FFT
- ○ Nyquist
- ○ Norm dB
- ○ Group Delay
- ○ Digital

☐ New Graph Sheet

Choose the required probe type and function, and whether you require to plot on a new graph sheet. Press **OK** and you will be prompted on the **Status Bar** to select the relevant item to create the plot.

Some of these functions are detailed in the next section “Notes on Probe Functions”

Showing results of DCOP analysis

The options **Show Pin Current** and **Show Net Current** display the results of a DC operating point analysis in the form of the voltage value on picked nets and current value on picked pins. The value is displayed in the Simulator’s **Command Shell** window.

New Graph Sheet / Window

Use **New Graph Sheet** and **New Graph Window** options whilst probing to choose where the next curve will be plotted.

Normally, if there are no graph windows open, one will be created. If a graph window is open and the currently displayed sheet has a compatible x-axis to what you are probing, the new curve will be added to that sheet. E.g. if the currently displayed graph is from a transient analysis and has an x-axis of "Time", and you are also probing the results of a transient analysis , then the new curve will be added to the displayed graph. If, however the displayed curve was from an AC analysis, its x-axis would be frequency which is incompatible. In this case a new graph sheet will be created for the new curve.

Use the **New Graph Sheet** command to force a fresh graph sheet in the Simulator's current graph window, or use the **New Graph Window** command to force a completely new graph window.

Plotting Noise Results

These three probes are only relevant after performing a Noise Analysis. Selecting **Plot Input Noise** and **Plot Output Noise** directly plot the results of the noise analysis without any further interaction.

Using **Probe Device Noise** prompts you to select a device for which you wish to plot the noise generated.

An error will be displayed if a "quiet" device is probed. A "quiet" device is one that does not have noise generating elements e.g. capacitors and inductors are quiet. Also subcircuits do not give a noise result.

Add Curve

More advanced plotting can be achieved with the shortcut option **Add Curve.** This opens the simulator’s **Define Curve** dialog box allowing you to enter any expression and which also provides a range of options on how you wish the graph to be plotted.

This is covered in detail in the “Plotting an Arbitrary Expression” section on the next few pages.

The **Define Curve** dialog will be shown as soon as you click on a net or device pin., and the name of the selected item will appear in the **Expression** field in the dialog. Subsequent nets and pins picked will also have their names sent to the Expression field. Alternatively you can choose the net and pin names to use from the dialog **Available Vectors** list.

Performance Analysis

Use this probe type as a method of cross probing to the simulator’s **Define Performance Analysis** dialog. This dialog allows you to enter an expression using a goal function to perform a computation on multiple curves, generated by a **Multi-Step Analysis**, to product a single value to plot. For example to plot a histogram for a **Monte Carlo** analysis.

This is covered in more detail in the **Performance Analysis and Histograms** section later in this chapter.

The **Define Performance Analysis** dialog will be shown as soon as you click on a net or device pin, and the item's name will appear in the dialogs expression field. Subsequent nets and pins picked will also have their names sent to the **Expression** field. Alternatively you can choose the net and pin names to use from the dialog **Available Vectors** list.

Probe Bus

The **Bus** probe will display a dialog to create a plot representing all the signals on the picked bus. It is possible to probe a bus as long as it is a closed bus and all the signals on the bus are digital. See **Probing Busses** section later in this chapter for more details.

Fourier

A Fourier spectrum of a signal can be obtained in a number of ways. Use the **Fourier** option on the shortcut menu to present a sub menu containing three options. You have a choice of using the default settings for the calculation of the Fourier spectrum or you can customise the settings for each plot. These options are covered in more detail later in this chapter.

Notes on Probe Functions

Impedance

Available from the **User Defined Probe** dialog. This only works in AC analysis. This works by calculating V/I at the device pin selected. Because currents are only available in voltage sources, inductors, resistors and capacitors with AC analysis, you must select a pin of one of these devices to find circuit impedance.

Device Power

Available from the **User Defined Probe** dialog. This works by calculating the sum of VI products at each pin of the device. Power is not stored during the simulation. However, once you have plotted the power in a device once, the result is stored with the vector name:

device_name#pwr

E.g. if you plot the power in a resistor R3, its power vector will be called R3#pwr. You can use this as part of an expression in any future plot.

Note that, because Pulsonix Spice is able to find the current in a sub-circuit device or hierarchical block, it can also calculate such a device's power. Be aware, however, that as this power is calculated from the VI product of the device's pins, the calculation may be inaccurate if the sub-circuit uses global nodes.

Plotting Transfer Function Analysis Results

No cross-probing is available with transfer function analysis. Instead, you must use the general purpose **Define Curve** dialog box. With this approach you must select a vector name from a list. Proceed as follows:

1. Select menu **Simulation|Random Probe|Add Curve**
2. Select a value from the **Available Vectors** drop down box.

Transfer Function Vector Names

The vector names for transfer function will be of the form:

source_name#Vgain
source_name#Transconductance
source_name#Transresistance
source_name#Igain

where *source_name* is the name of a voltage or current source.

The vectors Zout, Yout or Zin may also be available. These represent output impedance, output admittance and input impedance respectively.

Fourier Analysis

Fourier Analysis

A Fourier spectrum of a signal can be obtained in a number of ways. You have a choice of using the default settings for the calculation of the Fourier spectrum or you can customise the settings for each plot. The following menus use the default settings:

Schematics Random Probe:	**Fourier|Probe Voltage** (Quick Fourier)
Graph menu:	**Measure | Plot Fourier of Curve**
Graph menu:	**Measure | Plot Fourier of Curve** (Cursor span)

The following prompt you to customise the settings:

Schematics Random Probe:	**Fourier|Probe Voltage** (Custom Fourier)
Schematics Random Probe:	**Fourier|Add Fourier Plot**
Command shell menu:	**Graphs and Data | Fourier...**

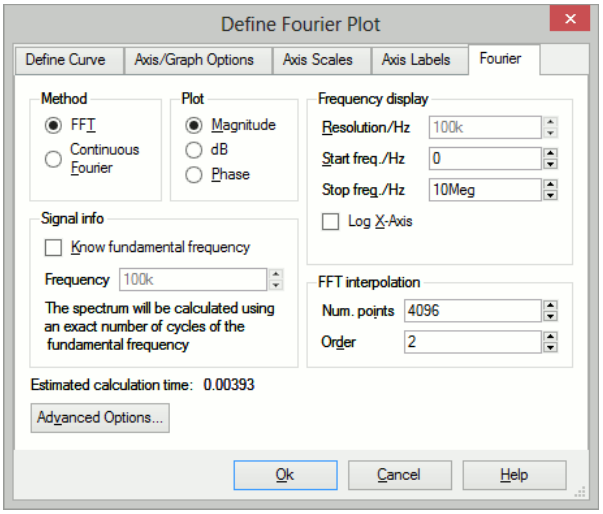
Default Settings

The default Fourier spectrum settings are:

Setting	**Default value**
Method	Interpolated FFT
Number of points	Next integral power of two larger than number of points in signal
Interpolation order	2
Span	All data except Measure|Plot Fourier of Curve (Cursor span) which uses cursor span

Custom Settings

With **Random Probe Voltage (Custom Fourier)** you will see the dialog below.



Method: [✓ FFT] Continuous Fourier

Plot: [✓ Magnitude] dB Phase

Frequency display

Resolution/Hz 100k

Start freq./Hz 0

Stop freq./Hz 10Meg

Log X-Axis

Signal info

Know fundamental frequency

Frequency 100k

The spectrum will be calculated using an exact number of cycles of the fundamental frequency

FFT interpolation

Num. points 4096

Order 2

Estimated calculation time: 0.00393

With the **Random Probe Add Fourier Plot** or with the **Command Shell** menu **Graphs and Data | Fourier** a dialog box similar to that shown in *Plotting an Arbitrary Expression* will be displayed but will include a **Fourier** tab. Click on the this tab to display the Fourier analysis options as shown below.

Method

Pulsonix Spice offers two alternative methods to calculate the Fourier spectrum: *FFT* and *Continuous Fourier*.

The simple rule is: use FFT unless the signal being examined has very large high frequency components as would be the case for narrow sharp pulses. When using Continuous Fourier, keep an eye on the Estimated calculation time shown at the bottom right of the dialog.

A description of the two techniques and their pros and cons follows.

FFT

Fast Fourier Transform. This is an efficient algorithm for calculating a discrete Fourier transform or DFT. DFTs generally operate on evenly spaced sampled data. Unfortunately the data generated by the simulator is not evenly spaced so it is therefore necessary to interpolate the data before presenting it to an FFT algorithm. The interpolation process is in effect the sampling process and the Nyquist sampling theorem applies. This states that the signal can be perfectly reproduced from the sampled data if the sampling rate is greater than twice the maximum frequency component in the signal. In practice this condition can never be met perfectly and any signal components whose frequency is greater than half the sampling rate will be *aliased* to a different frequency.

So if the number of interpolated points is too small there will be errors in the result due to high frequency components being aliased to lower frequencies. This is the Achilles heel of FFTs applied to simulated data.

The *Continuous Fourier* technique, described next, does not suffer from this problem. It suffers from other problems the main one being that it is considerably slower than the FFT.

Continuous Fourier

This calculates the Fourier spectrum by numerically integrating the Fourier integral. With this method, each frequency component is calculated individually whereas with the FFT the whole spectrum is calculated in one - quite efficient - operation. Continuous Fourier does not require the data to be interpolated and does not suffer from aliasing.

The problem with continuous Fourier is that compared to the FFT it is a slow algorithm and in many cases an FFT with a very large number of interpolated points can be calculated more quickly and give just as accurate a result.

However in cases where a signal has a very large high frequency content - such as narrow pulses - this method is superior and it is recommended that it is used in preference to the FFT in such situations.

The continuous Fourier technique has the additional advantage that it can be applied with greater confidence as the aliasing errors will not be present. It does have its own source of error due to the fact that simulated data itself is not truly continuous but represented by unevenly spaced points with no information about what lies between the points. This error can be minimised by ensuring that close simulation tolerances are used. See the “Convergence and Accuracy” section in the “Simulator” chapter for details.

Because each frequency component is calculated individually, the calculation time is affected by the values entered in Frequency Display. See below

Plot (Phase or Magnitude)

The default is to plot the magnitude of the Fourier spectrum. Select Phase if you require a plot of phase or dB if you need the magnitude in dBs,

Frequency Display

Resolution/Hz	Available only for the continuous Fourier method. This is the frequency interval at which the spectral components are evaluated. It cannot be less than 1/T where T is the time interval over which the spectrum is calculated.
Start Freq./Hz	Start frequency of the display.
Stop Freq./Hz	Stop frequency of the display.
Log X-Axis	Check this to specify a logarithmic x-axis. This will force a minimum value for the start frequency equal to 1/T where T is the time interval being analysed.

Signal Info

If the signal being analysed is repetitive and the frequency of that signal is known *exactly* then a much better result can be obtained if it is specified here. Check the Know fundamental frequency box then enter the frequency. The Fourier spectrum will be calculated using an integral number of complete cycles of the fundamental frequency. This substantially reduces *spectral leakage*. Spectral leakage occurs because both the

Fourier algorithms work on an assumption that the signal being analysed is a repetition of the analysed time interval from t=-8 to t=+8. If the analysed time interval does not contain a whole number of cycles of the fundamental frequency this will be a poor approximation and the spectrum will be in error. In practice this problem is minimized by using a *window* function applied to the signal prior to the Fourier calculation, but using a whole number of cycles reduces the problem further.

Note that the fundamental frequency is not necessarily the lowest frequency in the circuit but the largest frequency for which all frequencies in the circuit are integral harmonics. For example if you had two sine wave generators of 1kHz and 1.1KHz, the fundamental is 100Hz, not 1kHz; 1kHz is the tenth harmonic, 1.1KHz is the eleventh.

You should *not* specify a fundamental frequency for circuits that have self-oscillating elements.

FFT Interpolation

As explained above, the FFT method must interpolate the signal prior to the FFT computation. Specify here the number of points and the order. The number of points entry may be forced to a minimum if a high stop frequency is specified in the **Frequency Display** section.

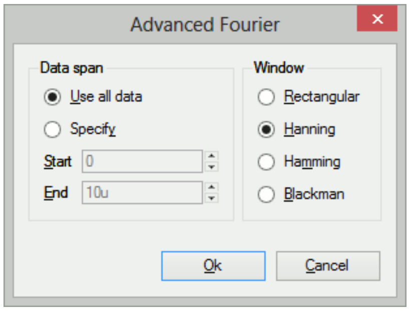
The number of interpolation points required depends on the highest significant frequency component in the signal being analysed. If you have an idea what this is, a useful trick to set the number of points to a suitable value, is to increase the stop frequency value in the Frequency Display section up to that frequency. This will automatically set the number of interpolation points to the required value to handle that frequency. If you don't actually want to display frequencies up to that level, you can bring the stop frequency back down again. The number of interpolation points will stay at the value reached.

If in doubt, plot the FFT twice using a different number of points. If the two results are significantly different in the frequency band of interest, then you should increase the number of points further.

Usually an interpolation order of 2 is a suitable value but you should reduce this to 1 if analysing signals with abrupt edges. If analysing a smooth signal such as a sinusoid, useful improvements can be gained by increasing the order to 3.

Advanced Options

Pressing the **Advanced Options**... button will open this dialog box:



Advanced Fourier

Data span
[✓ Use all data] Specify
Start 0
End 10u

Window
Rectangular [✓ Hanning] Hamming Blackman

Data Span

Usually the entire simulated time span is used for the Fourier analysis. To specify a smaller time interval click Specify and enter the start and end times.

Note that if you specify a fundamental frequency, the time may be modified so that a whole number of cycles is used. This will occur whether or not you explicitly specify an interval.

Window

A window function is applied to the time domain signal to minimise spectral leakage (See above).

The choice of window is a compromise. The trade-off is between the bandwidth of the main spectral component or lobe and the amplitude of the side-lobes. The rectangular window - which is in effect no window - has the narrowest main lobe but substantial side-lobes. The Blackman window has the widest main lobe and the smallest side lobes. Hanning and Hamming are something in between and have similar main lobe widths but the side lobes differ in the way they fall away further from the main lobe. Hamming starts smaller but doesn't decay whereas Hanning while starting off larger than Hamming, decays as the frequency moves away from the central lobe.

Despite the great deal of research that has been completed on window functions, for many applications the difference between Hanning, Hamming and Blackman is not important and usually Hanning is a good compromise.

There are situations where a rectangular window can give significantly superior results. This requires that the fundamental frequency is specified and also that the simulated signal is consistent over a large number of cycles. The rectangular window, however, usually gives considerably *poorer* results and must be used with caution.

Probing Busses

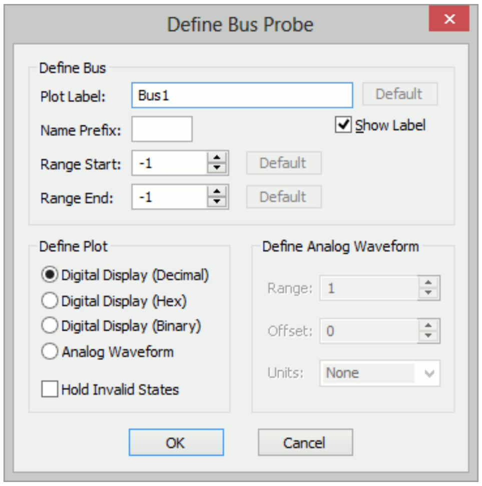
It is possible to probe a bus as long as all the signals on the bus are digital. In this case a plot representing all the signals on the bus will be created. Usually this will be a numeric display of the digital bus data, but it is also possible to display the data as an analog waveform.

To probe bus

1. Select **Random Probe|Probe Bus**
2. Click on desired bus
3. Enter the desired bus parameters as described in “Bus Probe Options” below.

Bus Probe Options

The following describes the options available for random and fixed bus probes. These options are set using the dialog box shown below.


Define Bus Probe

Define Bus
Plot Label: Bus1 Default
Name Prefix: [✓ Show Label]
Range Start: -1 Default
Range End: -1 Default

Define Plot
[● Digital Display (Decimal)]
Digital Display (Hex)
Digital Display (Binary)
Analog Waveform
Hold Invalid States

Define Analog Waveform
Range: 1
Offset: 0
Units: None

OK Cancel

Define Bus

Plot Label	This is how the curve will be labelled in the plot. It defaults to the bus name.
Name Prefix, Start, End	Defines which wires in the bus are used to create the displayed data. The default is to use all nets defined in the bus, indicated by range value -1. For a fixed bus probe, a blank name prefix indicates to probe the bus the probe symbol is over.

Plot Type

Decimal/Hexadecimal/Binary	Each of these specifies a numeric display (see below) showing the bus values in the number base selected.
Analog waveform	Specifies that the bus data should be plotted as an analog waveform.
Hold invalid states	If checked, then and invalid digital states found in the data will be replaced with the most recent valid state. If not checked, invalid states will be shown as an 'X' in numeric displays. This option is automatically selected for analog waveform mode.

Define Analog Waveform

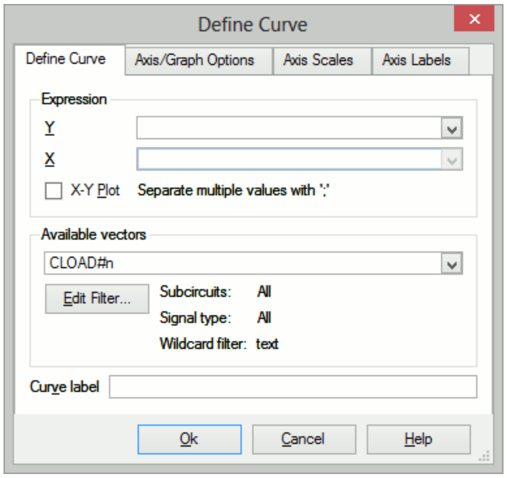
Only enabled if Analog waveform is specified in the Plot Type box. Specifies the scaling values and units for analog waveforms:

Range	Peak-peak value used for display
Offset	Offset for analog display. A value of zero will result in an analog display centred about the x-axis.
Units	Select an appropriate unit from the drop down box.

Plotting an Arbitrary Expression

Pulsonix Spice has a facility to plot an arbitrary expression of node voltages or device currents. This is accessed from within the Schematic design, **Simulation** menu, **Random Probe> Add Curve**, or from the **Command Shell** menu and **Graphs and Data** menu, **Add Curve...**

Selecting one of these option menu brings up the following dialog:



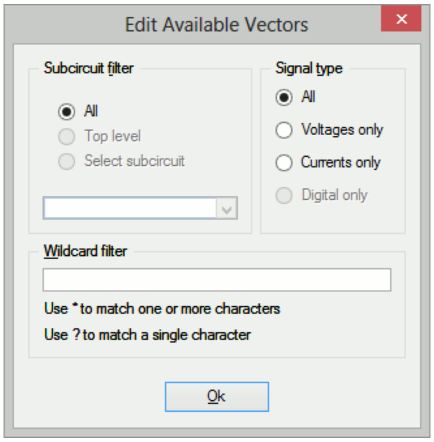
Define Curve

Expression	
Y	Enter arithmetic expression. This can use operators + - * / and ^ as well as the functions listed in the Simulator online help pages. Available voltages and currents may be selected from the "Available Vectors" box.
X	Expression for X data. Only required for X-Y plot and you must check X-Y Plot box. Expression entered in the same way as for Y data.
Available Vectors	Lists values available for plotting. (You need to tell Pulsonix Spice to save subcircuit currents and voltages using .KEEP). Press "Edit Filter" to alter selection that is displayed. See below. The names displayed are the names of the vectors created by the simulator. The names of node voltages are the same as the names of the nodes themselves. The names for device currents are composed of device name followed by a '#' followed by the pin name. Note that some devices output internal node voltages which could get confused with pin currents. E.g. q1#base is the internal base voltage of q1 not the base current. The base current would be q1#b. For the vector names output by a noise analysis refer to section on *The Simulator*, starting with .noise – Noise Analysis.
Edit Filter	See below

Curve Label	Enter text string to label curve

Edit Filter

Pressing the **Edit Filter** button on the **Define Curve** dialog displays:



Subcircuit filter: [✓ All] Top level / Select subcircuit

Signal type: [✓ All] Voltages only / Currents only / Digital only

Use * to match one or more characters

Use ? to match a single character

This allows you to select what is displayed in the available vectors dialog. This is useful when simulating large circuits and the number of vectors is very large.

Subcircuit Filter

All	Vectors at all levels are displayed
Top level	Only vectors for the top-level are displayed
Select subcircuit	All subcircuit references will be displayed in the list box. Select one of these. Only vectors local to that subcircuit will be displayed in Available Vector list.

Signal Type

All	List all signal types
Voltages Only	Only voltages will be listed
Currents only	Only currents will be listed
Digital Only	Only digital vectors will be listed

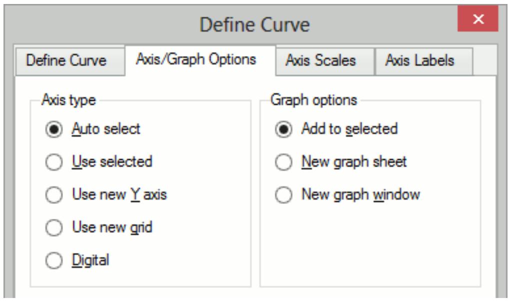
Wildcard filter

Enter a character string containing '*' and/or '?' to filter vector names. '*' matches one or more occurrences of any character and '?' matches any single character. Some examples:

*	matches anything
X1.*	matches any signal name that starts with the three letters: X1
X?.*	matches any name that starts with an X and with a '.' for the third letter
*.Q10#C	matches any name ending with .Q10#C i.e. the current into any transistor called Q10
*.U1.Vout	matches any name ending with .U1 C11 i.e. any node called Vout in a subcircuit with reference U1

Axis/Graph Options

The **Axis/Graph Options** page you to control where the curve for the probed signal will be placed.



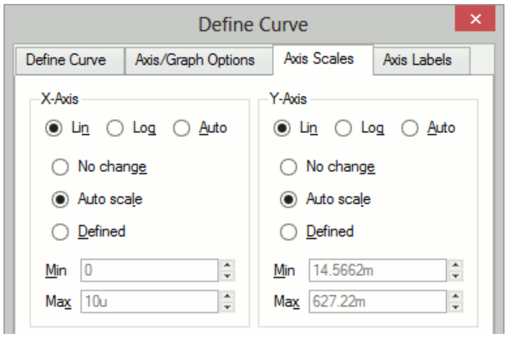
Axis type: [✓ Auto select] Use selected / Use new Y axis / Use new grid / Digital

Graph options: [✓ Add to selected] New graph sheet / New graph window

Axis Type	Select an appropriate axis type. Note that you can move a curve to a new axis or grid after it has been plotted.
Auto select	Select an appropriate axis automatically. See section on the "AutoAxis" feature for more info.
Use Selected	Use currently selected y-axis
Use New Y-axis	Create a new y-axis alongside main one
Use New Grid	Create a new grid stacked on top of main axis
Digital Axis	Create a new digital axis. Digital axes are placed at the top of the window and are stacked. Each one may only take a single curve. As their name suggests, they are intended for digital traces but can be used for analog signals if required.
Graph Options	
Add To Selected	Add curve to currently selected and displayed graph sheet
New Graph Sheet	Create a new graph sheet within current graph window
New Graph Window	Create a new graph window.

Axis Scales

The **Axis Scales** page allows you to specify limits for x and y axes.



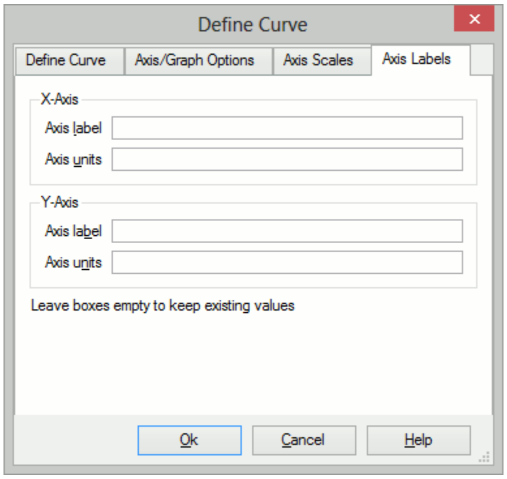
X-Axis: [✓ Lin] Log Auto; No change [✓ Auto scale] Defined; Min 0; Max 10u

Y-Axis: [✓ Lin] Log Auto; No change [✓ Auto scale] Defined; Min 14.5662m; Max 627.22m

X-Axis/Y-Axis	
Lin/Log/Auto	Specify whether you want X-Axis to be linear or logarithmic. If "Auto" is selected, the axis (X or Y) will be set to log if the <u>x values</u> are logarithmically spaced. For the Y-axis it is also necessary that the curve values are positive for a log axis to be selected.
No Change	Keep axes how they are. Only relevant if adding to an existing graph.
Auto scale	Set limits to fit curves
Defined	Set to limits defined in Min and Max boxes

Axis Labels

This sheet has four edit boxes allowing you to specify, x and y axis labels as well as their units. If any box is left blank, a default value will be used or will remain unchanged if the axis already has a defined label.



Leave boxes empty to keep existing values

Plot Journals and Updating Curves

Overview

You can repeat previous plotting operations in one of two ways.

The “Update Curves” feature rebuilds the current graph sheet using the latest available data. This allows you to randomly probe a schematic and then update the curves with new results for a new simulation run.

The “Plot Journal” feature allows you to save the plots in the current graph sheet for later reconstruction. This does not save the data, it saves the vector names and expressions used to create the graph’s curves. In fact this is done by building a Pulsonix Spice script to plot the curves.

Update Curves

Make sure that no curves are selected then select graph menu Plot|Update Curves. The curves currently on the graph sheet will be redrawn using the current simulation data. Although this would usually be the latest simulation run, you can also use this feature to restore the curves back to those from an earlier run. Use the Graphs and Data|Change Data Group… menu to select earlier data.

Options

By default all curves are redrawn, that is the older ones are deleted. You can change this behaviour so that older curves are kept. Select menu **Plot|Update Curves Settings**… then uncheck the **Delete old curves** box.

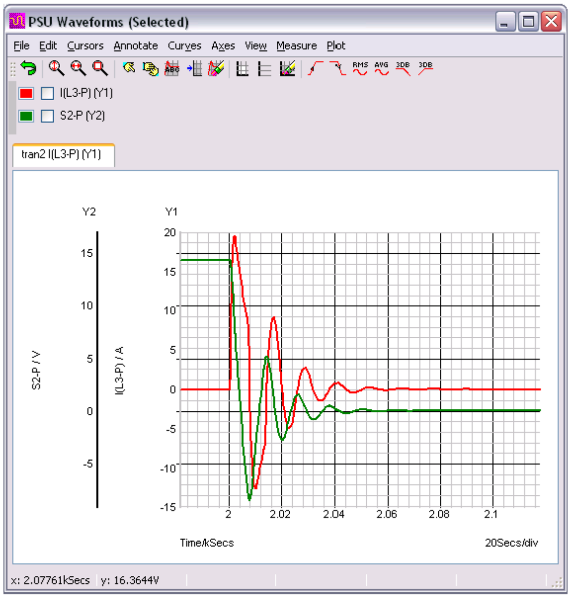
Plot Journals

First create a plot journal using the menu **Plot|Create Plot Journal…** then choose a file name. The file created has a .sxscr extension – it's the same extension used by scripts because the file created *is* a script.

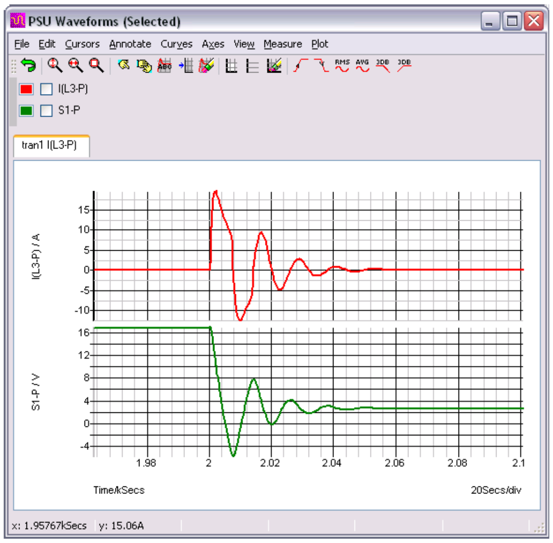
To run the plot journal, you will of course first need to run a simulation or load previous data so that the journal has some data to work with. The plot journal itself does not store any data. With the simulation data you wish to work with in place, select either graph menu **Plot|Run Plot Journal**… or command shell menu **Graph|Run Plot Journal**… This simply runs a script located in the current directory. Note that the plot journal always creates a new graph sheet.

Graph Layout - Multiple Y-Axis Graphs

Graphs may have additional Y axes to accommodate plotting results with incompatible scales. This occurs particularly for plotting dB and phase against each other and also for voltage and current. The additional Y axes may either be superimposed or stacked. In the user interface and the remainder of this documentation these are referred to respectively as **Axes** and **Grid**. These are illustrated below.



Current and Voltage plotted on separate "Axes"



Current and Voltage plotted on separate "Grids"

"AutoAxis" feature

When you plot a new curve on an existing graph, Pulsonix Spice will select - or if necessary create - a compatible axis for that curve. The decision is made on the basis of the curve's Units i.e. voltage, current etc. The rules it follows are:

- If the currently selected axis or grid (shown by black axis line) has the same units as curve to be plotted or if it has undefined units (designated by a '?' on label), that axis will be used.
- If any other axis or grid has compatible units (i.e. same as curve or undefined) that axis will be used.
- If no axes exist with compatible units, a new axis (not grid) will be created to accommodate the curve.

The above works for random probing. For plots created with Fixed Probes this is the default behaviour, but it can be changed. See Fixed Probes for more detail.

Manually Creating Axis and Grids

Two toolbar buttons "Create new grid" and "Create new axis" allow manual creation of new axes and grids. These will be initially empty. Subsequent random probe operations will use the new axis or grid unconditionally as long as it remains selected (see below).

Selecting Axis

Some operations are performed on the selected axis or grid. The selected axis or grid will be displayed with its vertical axis line a deep black while the remaining axes and grids will be light grey. Newly created axes and grids are always selected. To select an axis, click the left mouse button immediately to the left of the vertical axis line.

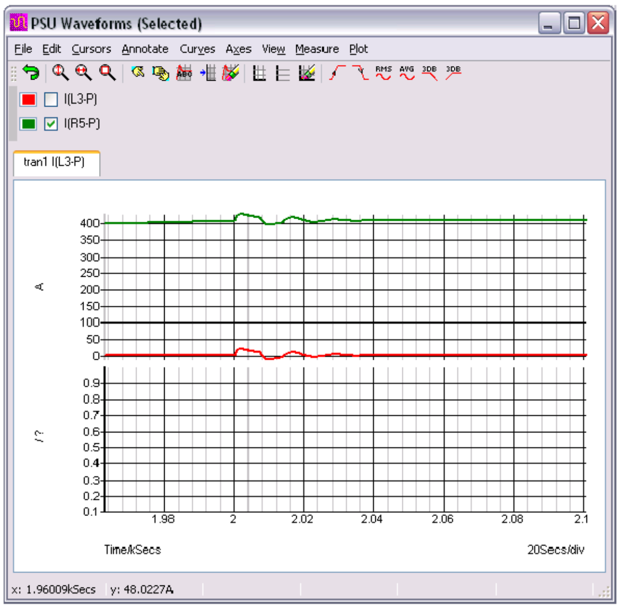
Moving Curves to Different Axis or Grid

You can freely move curves around from one axis or grid to another. Proceed as follows:

- Select the curve or curves you wish to move by checking its checkbox next to the coloured legend which designates the curve.
- Select the axis you wish to move it to. (See above)
- Press the "Move selected curves to new axis" button. The curves will be re-drawn on the new axis. Any axes that become empty as a result of this operation will be deleted unless it is the "Main" axis. See section below on "Deleting axes"

Deleting Axis

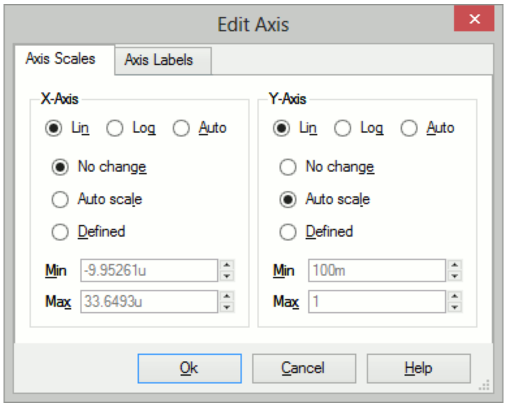
To delete an axis, select it then press **Erase axis** button. Note that you cannot erase an axis or grid that has curves attached to it nor can you erase the **Main** axis. The main axis is the first axis that is created on a graph. For example with the following graph:



If you attempt to delete the selected axis (the lower one), nothing will happen. Instead you should move the two curves in the top axis to the lower one. See above section on how to move curves.

Editing Axis

You can edit axis scales, label and units by selecting the graph popup menu **Edit Axis...** This brings up the following dialog box:


Edit Axis

Axis Scales | Axis Labels

X-Axis: [✓ Lin] Log Auto; [✓ No change] Auto scale Defined; Min -9.95261u; Max 33.6493u

Y-Axis: [✓ Lin] Log Auto; No change [✓ Auto scale] Defined; Min 100m; Max 1

The function of the Axis scales sheet and axis labels sheet is similar to the sheets of the same name in the define curve dialog box.

Reordering Grids and Digital Axes

You can change the vertical order of the analog grids and digital axes. To change the analog grid order:

1. Select **Axes|Reorder Grids**...
2. You will be presented with a list of currently displayed grids identified by their yaxis title. Use the up and down arrow buttons to arrange them in the order required then press **Ok**

Note that the main axis (the one at the bottom) cannot be moved.

To change the digital axis order:

1. Select menu **Axes|Reorder Digital Axes**...
2. Rearrange entries in list as described above for analog grids.

Plotting the Results from a Previous Simulation

- Select the menu item Graphs and Data|Change Data Group...
- Select the name of the previous run (or group) that you require. The current group will be highlighted. (Note that the AC analysis mode generates two groups. One for the ac results and the other for the dc operating point results. Transient analysis will do the same if the start time is non-zero).
- Plot the result you require in the normal way. A word of warning: If the netlist (or schematic) has undergone any modifications other than component value changes since the old simulation was completed, some of the net names may be different and the result plotted may not be of what you were expecting.

Note: By default, only the three most recent groups are kept. This can be changed using the Group Persistence option using the Set command. A particular group can be kept permanently using the Graphs and Data|Keep Current Data Group menu item.

Although only three groups are held at a time, the data is actually stored on a disc file which will not necessarily have been deleted. If you wish to access an old run, use File|Load Data... and retrieve the data from the TEMPDATA directory created under the Pulsonix Spice install directory. The file will have the same name as the group appended with ".SXDAT". Whether or not the data file is still available depends on a preference setting. In the case of Monte Carlo Analyses, it will be named mcn.SXDAT.

Combining Results from Different Runs

There are occasions when you wish to - say - plot the difference between a node voltage for different runs. To do this in Pulsonix Spice you need to type a command at the command line. This will be of the form "Plot *vector1-vector2*" for the example of plotting the difference. *vector1* and *vector2* are the names of the signals. However, as the two signals come from different runs we need a method of identifying the run. This is done by prefixing the name with the *group name* followed by a colon. The group name is an analysis type name (tran, ac, op, dc, noise, tf or sens) followed by a number. The signal name can be obtained from the Schematic. For voltages, this is simply the name of the net and for currents it is the device name followed by a '#' followed by the pin name. E.g. q3#c is the current into the collector of Q3. The group name is displayed in the simulator progress box when the simulation is running. You can also find the current group by selecting **Simulator|Change Data Group...** and noting which group is highlighted in the dialog box.

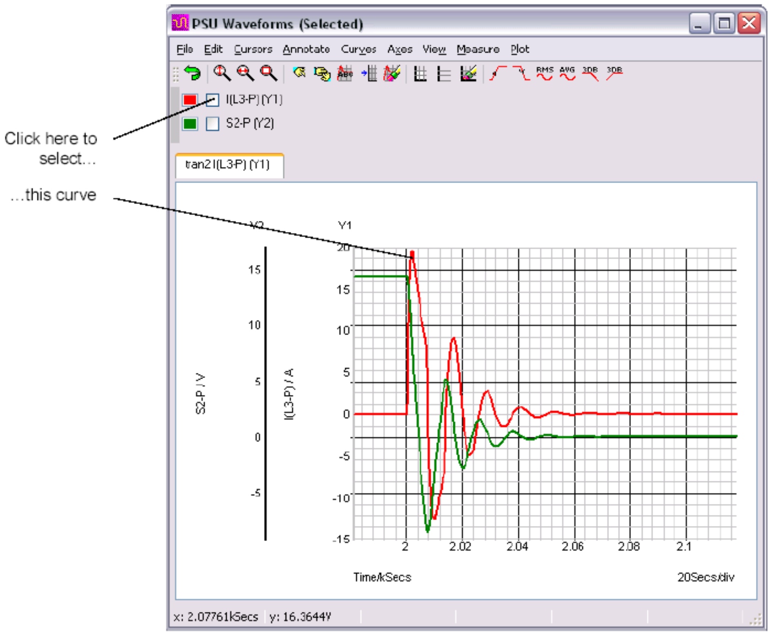
Here is an example. In tutorial1, the signal marked with the "Amplifier Output" probe is actually called "Q3_E". The latest run (group) is called "tran4". We want to plot the output subtracted from the output for the previous run. The previous run will be "tran3". So we type at the command line:

```
Plot tran4:q3_e-tran3:q3_e
```

This will create a new graph sheet. If you wanted to add the curve to an existing graph, use the Curve command instead of Plot.

Curve Operations

Selecting Curves



Deleting Curves

To delete a curve (or curves), select it (or them) then press "Erase selected curves" button. Any axes or grids other than "Main" axis left empty by this operation will also be deleted.

Hiding and Showing Curves

A curve may be hidden without it actually being deleted. This is sometimes useful when there are many curves on a graph but the detail of one you wish to see is hidden by others. In this instance you can temporarily remove the curves from the graph. To hide a curve (or curves) select it (or them) then press "Hide selected curves" button. To show it (or them) again, press "Show selected curves" button.

Re-titling Curves

You can change the title of a curve by selecting it then pressing **Name curve** button. This will change the name of the curve as displayed in the legend panel. (Above main graph area and below toolbar).

Highlighting Curves

You can highlight one or more curves so that they stand out from the others. This is useful if there are many overlapping curves displayed.

To Highlight Curves

Select the curves you wish to highlight then press <H> or menu **Curves | Highlight Selected Curves**.

To Unhighlight Curves

Select the curves you wish to unhighlight then press <U> or menu **Curves | Unhighlight Selected Curves**.

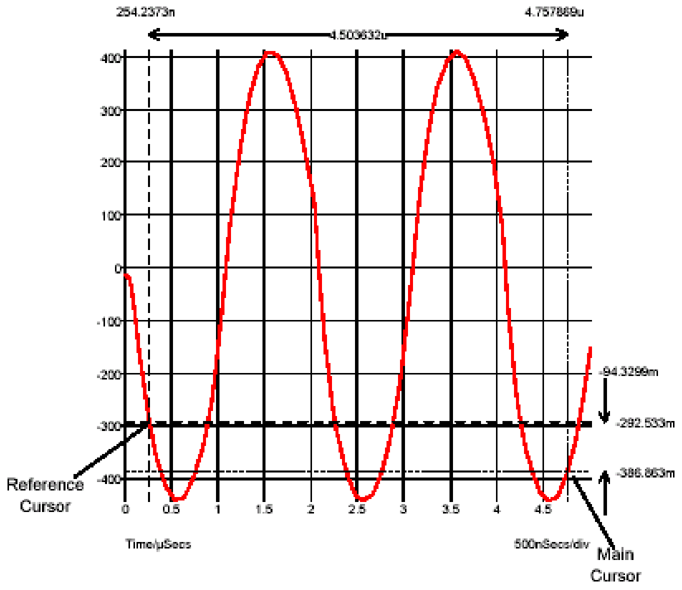
To Unhighlight All Curves

Select menu **Curves | Unhighlight All Curves**.

Graph Cursors

Overview

Graph cursors can be used to make measurements from waveforms. In their default configuration they consist of two dimensioned crosshairs as shown below:



The cursors can be moved horizontally or vertically while tracking an attached curve or they can be picked up and dragged onto another curve. Initially there are just two cursors, but there is the facility to add additional cursors without any maximum limit.

Cursor Operations

Displaying

To switch on/off the cursor display select the graph menu Cursors | Toggle On/Off.

Moving

Cursors can be moved by any of three methods:

1. Left to right. In this mode the x-position of the cursor is varied while the cursor tracks the curve to which it is attached. To use this method, place the mouse on the vertical crosshair but away from the intersection with the horizontal crosshair. You should see the mouse cursor shape change to a left-right arrow. Press left mouse key and drag.
2. Up-down. Similar to 1. above but instead the y-position is varied. To use this method, place the mouse on the horizontal crosshair but away from the intersection with the vertical crosshair. You should see the mouse cursor shape change to an up-down arrow. Press left mouse key and drag.
3. Drag and drop. In this mode the cursor is picked up and moved without tracking any curve. It can be dropped to any location and will then snap to the nearest curve. To use this method, place the mouse cursor at the intersection of the crosshairs. You will see the cursor shape change to a four-pointed arrow. Press left key and drag to new location.

You can also move the reference cursor in left-right mode using the right mouse button.

Moving Cursors along a Curve

You can move a cursor to a peak or trough using the hot-key defined in the following table:

Key	**Function**
F5	Move main cursor to next peak
Shift-F5	Move main cursor to previous peak
F6	Move main cursor to next trough
Shift-F6	Move main cursor to previous trough
F7	Move reference cursor to next peak
Shift-F7	Move reference cursor to previous peak
F8	Move reference cursor to next trough
Shift-F8	Move reference cursor to previous trough

These operations can also be accessed from the graph menu Cursors | Move.

Hiding Cursors

You can temporarily hide all or some of the displayed cursors. Menu Cursors | Hide/ Show | All has a toggle action and will hide all cursors if all cursors are currently displayed and vice-versa. If some cursors are visible and some are hidden, you will be presented with an option to hide all cursors or show all cursors. Menu Cursors | Hide/Show | Select allows you to selectively hide or show some cursors.

Freezing Cursors

You can freeze the cursors so that they can't be moved accidentally. Select menu **Cursors | Freeze/Unfreeze**.

Aligning Cursors

Select menu **Cursors | Align** to align the two cursors so that they have the same y position.

Changing Cursor Style

The crosshair style is the default but there is an alternative style where each cursor is a small cross. Select menu **Cursors | Set style to change**. This is the same style used with Pulsonix Spice version 1.

Additional Cursors

Pulsonix-Spice has the ability to display any number of cursors, not just the standard two.

To Add an Additional Cursor

1. Select menu **Add Additional Cursor...**
2. Enter a suitable label for the cursor. This is displayed at the bottom of the graph and to avoid clutter, we recommend that you use a short label such as a single letter.
3. Select to which other cursor you wish the new cursor to be referenced for both horizontal and vertical dimensions. Select '**none**' if you do not wish it to be referenced to any currently displayed cursor. Note that you may reference further additional cursors to this one if desired.
4. Press Ok. The new cursor will be initially displayed at the start of the x-axis and attached to the first curve on the sheet. You may subsequently move it as desired.

To Remove Additional Cursors

1. Select **Cursors** | **Remove Additional Cursors...**
2. Select the cursor or cursors to be removed. These are identified by their labels.

Cursor Readout

There are a number of options as to how the cursors' absolute and relative positions are displayed. Initially all values are displayed as dimensions on the graph. This can be altered in a number of ways:

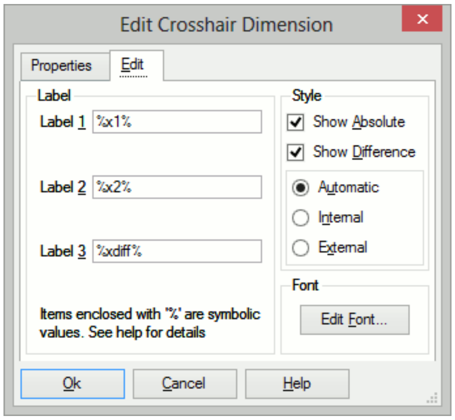
You can opt to have just the absolute or just relative readings displayed

The actual format of the graph readout can be customised. E.g. extra text can be added, perhaps something like 'Delay = xxxnS' where xxx is the relative reading.

The values can optionally be displayed in the status bar with or without the graph readings.

Editing Style or/and Format of Cursor Dimension

To edit the **Cursor Dimension**, **double-click** on one of the displayed values of the cursor dimension or select it, then select **Edit Selected Object** from the **Annotate** menu. The following dialog will open:



Edit values as described below:

Label

The labels are the three values displayed on the dimension. Label 1 is the value displayed above the reference cursor, label 2 is the value displayed above the main cursor and label 3 is the value displayed as the difference. %x1%, %x2% and %xdiff% are symbolic values that will be substituted with the absolute position of the reference cursor, the absolute position of the main cursor and the difference between them respectively. You can add additional text to these. For example, if you changed label 1 to 'Pulse Start = %x1%' the value displayed for the position of the reference cursor would be prefixed with 'Pulse Start = '.

You can use expressions relating constants and symbolic values enclosed by '%'. Expressions must be enclosed in braces: '{' and '}'. For example, the expression {1/%xdiff%} will cause the difference value to be displayed as a reciprocal. This is useful if you wanted to display a frequency instead of a period. You can use any arithmetic operator along with many of the functions described in the script reference manual in these expressions.

Style

Show Absolute	Clear check box to disable display of the absolute positions of the cursors.
Show Difference	Clear check box to disable display of relative positions.
Automatic/Internal/External	Style of dimension. Internal means that the arrows will always be displayed between the cursors.

External means they will always be displayed outside the cursors. In automatic mode the style will change according to the spacing and position.

Note, if you clear both absolute and difference, you will only be able to restore the display of the dimension by switching cursors off then on again.

Edit Font

Select the font used for readout text.

Properties Tab

The **Properties** tab lists all available properties of the **Crosshair Dimension** object.

Status Bar Readout

You can optionally have the cursor read out in the status bar instead of or as well as the on-graph dimension display. Select menu **Cursors | Display Options...** and select option as required. This will change the current display.

You can opt to have this preference used as the default. Select Command Shell menu **File | Options | General...** then Graph/Probe/Data Analysis tab. Select appropriate option in Cursor readout section.

Show Curve Info

The menu Cursors | Show Curve Info will display in the Command Shell information about the curve which currently has the main cursor attached. The following information is listed:

Curve name

Source group	Source group The name of the simulation group that was current when the curve was created
Curve id	Only required when accessing curves using script commands.
Run number	If there are multiple curves generated by a Monte Carlo run, this is a number that identifies the run number that created the curve. This number can be used to plot the curve alone and also to identify the seed value used for that Monte Carlo step.

Cursor Functions

There are four functions which return the current positions of the cursors and these can be used in script expressions . These are:

XDatum()

YDatum()

XCursor()

YCursor()

Curve Measurements

Overview

A number of measurements can be applied to selected curves. The results of these measurements are displayed below the curve legend and are also printed. Some of these measurements can be selected from the tool bar and more can be called directly from the "Legend Panel's" pop-up menu (Right click in Legend Panel). The remainder are listed in a tree structured list that is opened with the legend panel menu **More Functions...** or by pressing F3.

Note that the legend panel may be resized by dragging its bottom edge with the mouse.

In general to perform a measurement, select the curve or curves then select measurement from tool bar or legend panel popup menu. If there is only one curve displayed, it is not necessary to select it.

Available Measurements

The following table list all the measurements available and how they may be accessed:

Transient measurements:	Access from	Description
RMS/Full	menu - RMS menu - More Functions...	RMS for full span
RMS/Per Cycle	Toolbar menu - RMS/cycle menu - More Functions...	RMS for each cycle
RMS/Cursor span	menu - More Functions...	RMS over span defined by cursor positions
RMS - AC coupled/Full	menu - More Functions...	RMS of AC component only for full span
RMS - AC coupled/Per Cycle	menu - More Functions...	RMS of AC component only for each cycle
RMS - AC coupled/Cursor span	menu - More Functions...	RMS of AC component only over span defined by cursor positions
Mean/Full	menu - Mean menu - More Functions...	Mean for full span
Mean/Per Cycle	Toolbar menu - Mean/cycle menu - More Functions...	Mean for each cycle
Mean/Cursor span	menu - More Functions...	Mean over span defined by cursor positions
Peak/Full	menu - Peak menu - More Functions...	Magnitude and location of peak for full span
Peak/Cursor span	menu - More Functions...	Magnitude and location of peak over span defined by cursor positions
Trough/Full	menu - Trough menu - More Functions...	Magnitude and location of trough for full span
Trough/Cursor span	menu - More Functions...	Magnitude and location of trough over span defined by cursor positions
Peak-Peak/Full	menu - More Functions...	Peak-peak for full span
Peak-Peak/Cursor span	menu - More Functions...	Peak-peak over span defined by cursor positions
Rise Time/First edge/10-90%	Toolbar menu - Rise Time menu - More Functions...	Rise time of first rising edge in waveform, measured at 10% and 90% points
Rise Time/First edge/20-80%	menu - More Functions...	Rise time of first rising edge in waveform, measured at 20% and 80% points

Rise Time/First edge/Custom	menu - More Functions...	Rise time of first rising edge in waveform, measured at user defined points
Rise Time/Cursors define edge/10-90%	menu - More Functions...	Rise time of edge defined by cursor positions, measured at 10% and 90% points
Rise Time/Cursors define edge/20-80%	menu - More Functions...	Rise time of edge defined by cursor positions measured at 20% and 80% points
Rise Time/Cursors define edge/Custom	menu - More Functions...	Rise time of edge defined by cursor positions, measured at user defined points
Fall Time/First edge/10-90%	Toolbar menu - Fall Time menu - More Functions...	Fall time of first falling edge in waveform, measured at 10% and 90% points
Fall Time/First edge/20-80%	menu - More Functions...	Fall time of first falling edge in waveform, measured at 20% and 80% points
Fall Time/First edge/Custom	menu - More Functions...	Fall time of first falling edge in waveform, measured at user defined points
Fall Time/Cursors define edge/10-90%	menu - More Functions...	Fall time of edge defined by cursor positions, measured at 10% and 90% points
Fall Time/Cursors define edge/20-80%	menu - More Functions...	Fall time of edge defined by cursor positions measured at 20% and 80% points
Fall Time/Cursors define edge/Custom	menu - More Functions...	Fall time of edge defined by cursor positions, measured at user defined points
Duty Cycle/Full	menu - More Functions...	Duty cycle measured over full span
Duty Cycle/Cursor span	menu - More Functions...	Duty cycle measured over span defined by cursor positions
Overshoot/Absolute/Auto	menu - More Functions...	Maximum absolute value of overshoot
Overshoot/Relative/Auto	menu - More Functions...	Maximum value of overshoot relative to pulse amplitude
Overshoot/Absolute/Cursors define edge	menu - More Functions...	Absolute value of overshoot measured at pulse defined by cursor positions
Overshoot/Relative/Cursors define edge	menu - More Functions...	Value of overshoot relative to pulse amplitude measured at pulse defined by cursor positions

Settling time/1%	menu - More Functions...	Settling time to 1%. Pulse defined by cursor positions
Settling time/0.1%	menu - More Functions...	Settling time to 0.1%. Pulse defined by cursor positions
Settling time/0.01%	menu - More Functions...	Settling time to 0.01%. Pulse defined by cursor positions
Settling time/Custom	menu - More Functions...	Settling time to user defined specification. Pulse defined by cursor positions
Frequency/Full	menu - Frequency menu - More Functions...	Frequency measured over full span.
Frequency/Cursor span	menu - More Functions...	Frequency measured over span defined by cursor positions
Distortion/Full	menu - More Functions...	Distortion measured over full span
Distortion/Cursor span	menu - More Functions...	Distortion measured over span defined by cursor positions
AC measurements		
-3dB Lowpass	Toolbar menu - More Functions...	-3dB point assuming a low pass roll off
-3dB Highpass	Toolbar menu - More Functions...	-3dB point assuming a high pass roll off
+3dB Lowpass	menu - More Functions...	+3dB point assuming a response with low frequency lift
+3dB Highpass	menu - More Functions...	+3dB point assuming a response with high frequency lift
Other Measurements		
Temperature coefficient	menu - More Functions...	Average temperature coefficient for temperature sweep only
Total Noise	menu - More Functions...	Integrated noise for noise analysis only

Notes on Algorithms Used for Curve Measurements

Some of the measurements algorithms make some assumptions about the wave shape being analysed. These work well in most cases but are not "fool-proof". The following notes describe how the algorithms work and what their limitations are.

All the measurement algorithms are implemented by internal scripts. The full source of these is supplied on the installation CD. These can be adapted to create your own measurement functions. More information can be found in the script reference manual.

"/Cycle" and Frequency measurements

These measurements assume that the curve being analysed is repetitive and of a fixed frequency. The results may not be very meaningful if the waveform is of varying frequency or is of a "burst" nature. The "/cycle" measurements calculate over as many whole cycles as possible.

Each of these measurements use an algorithm to determine the location of x-axis crossings of the waveform. The algorithm is quite sophisticated and works very reliably. The bulk of this algorithm is concerned with finding an optimum base line to use for x-axis crossings.

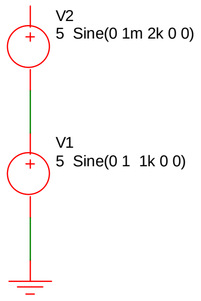
The per cycle measurements are useful when the simulated span does not cover a whole number of cycles. Measurements such as RMS on a repetitive waveform only have a useful meaning if calculated over a whole number of cycles. If the simulated span does cover a whole number of cycles, then the "Full" version of the measurement will yield an accurate result.

Rise and fall time, and overshoot measurements

These measurements have to determine the waveforms pulse peaks. A histogram method is used to do this. Flat areas of a waveform produce peaks on a histogram. The method is very reliable and is tolerant of a large number of typical pulse artefacts such as ringing and overshoot. For some wave-shapes, the pulse peaks are not well enough defined to give a reliable answer. In these cases the measurement will fail and an error will be reported.

Distortion

This calculates residue after the fundamental has been removed using an FFT based method. This algorithm needs a reasonable number of cycles to obtain an accurate result. The frequency of the fundamental is displayed in the message window. Note that most frequency components between 0Hz and just before the second harmonic are excluded. The precision of the method can be tested by performing the measurement on a test circuit such as:



The signal on the pos side of V2 has 0.1% distortion. Use V1 as your main test source (assuming you are testing an amplifier) then after the simulation is complete, check that the distortion measurement of V2 is 0.1%. If it is inaccurate, you will need either to increase the number of measurement cycles or reduce the maximum time step or both.

You can adjust the amplitude of V2 appropriately if the required resolution is greater or less than 0.1%.

Note, that in general, accuracies of better than around 1% will require tightening of the simulation tolerance parameters. See section "Convergence and Accuracy" for more information on this subject.

Frequency response calculations

These must find the passband for their calculations. Like rise and fall a histogram approach is used to find its approximate range and magnitude. Further processing is performed to find its exact magnitude.

Note that the algorithms allow a certain amount of ripple in the passband which will work in most cases but will fail if this in excess of about 3dB.

Note that the frequency response measurements are general purpose and are required to account for a wide variety of responses including those with both high and low pass elements as well as responses with band pass ripple. This requirement compromises accuracy in simpler cases. So, for example, to calculate the -3dB point of a low pass response that extends to DC, the 0dB point is taken to be a point midway between the start frequency and the frequency at which roll-off starts. A better location would be the start frequency but this would be inaccurate if there was a high pass roll off at low frequencies. Taking the middle point is a compromise which produces good - but not necessarily perfect - results in a wide range of cases. To increase accuracy in the case described above, start the analysis at a lower frequency, this will lower the frequency at which the 0dB reference is taken.

Plots from curves

Two plots can be made directly from selected curves. These are described below

FFT of selected curve

With a single curve selected, from legend popup menu select
Plot FFT of Selected Curve. A new graph sheet will be opened with the FFT of the curve displayed. To plot an FFT of the curve over the span defined by the cursor locations select **Plot FFT of Selected Curve (Cursor span)**.

Smoothed curves

With a single curve selected, from legend popup menu select **More Functions...** then under the **Plot** branch select **LP Filter** and choose a time constant. Press **OK** and a new curve will be displayed showing a smoothed version of the original curve.

This system uses a first order digital IIR filter to perform the filtering action.

Graph Zooming and Scrolling

Zooming with the Mouse

To zoom in on a portion of a graph, place the cursor at the top left of the area you wish to view, press and hold the left mouse key then move cursor to bottom right of area and release left key. The axes limits will be modified appropriately.

To view whole graph again select the graph popup **Zoom|Full** or toolbar button.

Zooming with the Keyboard

F12 to zoom out

shift-F12 to zoom in

HOME returns graph to full view. (Same as graph popup **Zoom|Full**)

Recovering an earlier zoom

Press the toolbar "**Undo Zo**om" button to recover earlier zoom or scroll positions.

Scrolling with the keyboard

up, down, left and right cursor keys will scroll the active graph.

Zooming selected axes.

When using the mouse to zoom graphs, all curves on the same grid are normally zoomed together. To zoom only the curve on the selected grid, press and hold the shift key while selecting the zoom area with the mouse.

Notes: *If you add a new curve to a graph which has been zoomed, the axes limits will not change to accommodate that curve; if the new curve does not lie within the zoomed area you will not see it. Selecting graph popup **Zoom|Full** or pressing HOME key restores the graph to auto-scaling and the limits will always adjust so that all curves are visible even ones subsequently added.*

Zoom to Fit Y-axis

To zoom in the y-axis only to fit the displayed x-axis, press the **Fit Height** toolbar button:

Annotating a Graph

A number of objects are available to annotate graphs for documentation purposes. These are:

- Curve Marker. A single arrow, line and item of text to identify a curve or feature of a curve
- Legend Box. Box of text that lists all the names of curves currently displayed.
- Text Box. Box containing text message.
- Free Text. Similar to text box but without border and background.
- Caption. As free text but designed for single line heading.

Curve Markers

Placing

To place a curve marker, select menu Annotate | Add Curve Marker. A single curve marker should appear in the right hand margin of the graph.

Moving

To move it, place the mouse cursor at the arrow head - you should see the cursor shape change to a four pointed arrow - then left click and drag to your desired location. When you release the marker it will snap to the nearest curve.

Moving Label

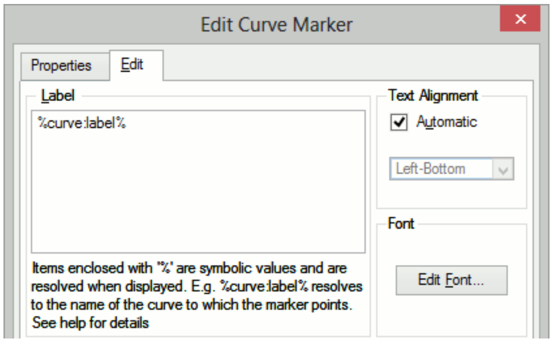
To move the text label alone, place the mouse cursor to lie within the text, then left click and drag. You will notice the alignment of the text with respect to the arrowed line change as you move the text around the arrow. You can fix a particular alignment if preferred by changing the marker's properties. See below.

Deleting

First select the marker by a single left click in the text. The text should change colour to blue. Now press delete key or menu **Annotate | Delete Selected Object**.

Editing Properties

To edit the **Curve Marker**, **double-click** the marker's label or select then select **Edit Selected Object** from the **Annotate menu**. The following dialog will open:



%curve:label%

[✓] Automatic

Left-Bottom

Items enclosed with '%' are symbolic values and are resolved when displayed. E.g. %curve:label% resolves to the name of the curve to which the marker points. See help for details

Label

Text of the marker's label. %curve:label% automatically resolves to the curve's label. If the curve name is edited with menu **Curves | Rename** curve this value will reflect the change. You can of course enter any text in this box.

You can also use expressions in the same manner as for cursor dimensions. See “Label” previously.

Text Alignment

This is how the label is aligned to the arrowed line. If set to automatic the alignment will be chosen to be the most appropriate for the relative position of the label and the arrowhead. Uncheck automatic and select from the list to fix at a particular alignment.

Font

Press **Edit Font**... to change font for text.

Other Properties

Snap To Curve	You can switch off the action that causes curve markers to always snap to a curve. Select Properties tab then double click on **Snap To Curve** item. Select **Off**. You will now be able to move the curve marker to any location.

Legend Box

Placing

Select menu **Annotate | Add Legend Box.** A box listing all the curve names will appear at the top left of the graph.

Moving

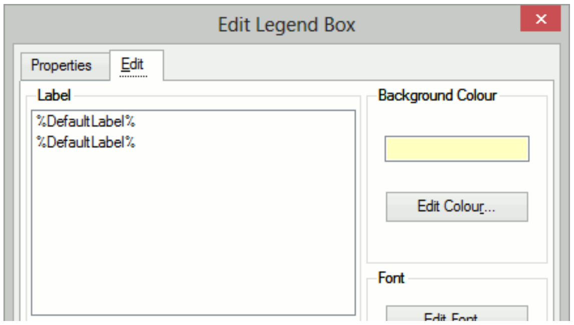
Place cursor inside the box and drag to new location.

Resizing

You can alter the maximum height of the box by placing the mouse cursor on it's bottom edge and dragging. The text in the box will automatically reposition to comply with the new maximum height.

Editing Properties

To edit a **Legend**, **double-click** on the box or select the item, then from the **Annotate** menu Annotate select the **Edit Selected Object**. The following dialog will be opened:



%DefaultLabel%
%DefaultLabel%

Label

Lists each label in the box. These are usually %DefaultLabel% which resolves to the name of the referenced curve. To edit, double click on the desired item. You can also enter the symbols %X1% and %Y1% which represent the x and y coordinates of the marker respectively. These can be combined with other text in any suitable manner. For example: ‘Voltage @ %X1%S = %Y1%’ might resolve to something like ‘Voltage at 10u = 2.345’. The values of %X1% and %Y1% will automatically update if you move the marker.

You can also use expressions in the same manner as for cursor dimensions. See “Label” previously.

Background Colour

Select button Edit Colour... to change background colour. To change the default colour select Command Shell menu **File | Options | Colour**... then select item **Text Box**. Edit colour as required.

Font

Select button **Edit Font**... to change font. To change the default font select Command Shell menu **File | Options | Font**... then select item Legend Box. Edit font as required.

Text Box

Placing

Select menu **Annotate | Text Box**. Enter required text then **OK**. You can use the symbolic constants %date%, %time%, and %version% to represent creation date, creation time and the product version respectively.

Moving

Place cursor inside the box and drag to new location.

Editing Properties

Double click on the box or select then menu **Annotate | Edit Selected Object**. A dialog like the one shown for legend boxes (see above) will be displayed Note when editing the label, you can use the symbolic constants as detailed in “Placing” above.

Caption and Free Text

The Caption and Free Text objects are essentially the same, the only difference is their initial font size and position.

Placing

Select menu **Annotate | Caption** or **Annotate | Free Text**. Enter required text then Ok. You can use the symbolic constants %date%, %time%, and %version% to represent creation date, creation time and the product version respectively.

Moving

Place cursor inside the box and drag to new location.

Editing Properties

Double click on the box or select then menu **Annotate | Edit Selected Object**. This will open a dialog similar to the one shown for curve markers but without the Automatic option for text alignment.

Copying to the Clipboard

Overview

Pulsonix Spice offers facilities to copy both graph data and the graph's graphical image to the system clipboard. This provides the ability to export simulation results to other applications. The data - for example - may be exported to a spreadsheet application for custom processing, while the graphical image may be exported to a word processor for the preparation of documents.

Pulsonix Spice may also import data in a tabulated ASCII format. This feature may be used to display data from a spreadsheet allowing, for example, a comparison between measured and simulated data.

As well as the system clipboard, Pulsonix Spice also uses an internal clipboard to which graph curves may be copied. This provides an efficient method of moving or copying curves to a new graph sheet.

Copy Data to the Clipboard

1. Select the graphs you wish to export
2. Select the menu Edit | Copy ASCII Data

The data will be copied in a tabulated ASCII format. The first line will contain the names of the curves, while the remaining lines will contain the curves' data arranged in columns

Copying Graphics to the Clipboard

There are three different ways a graph can be copied to the clipboard. Use the menus under Edit | Copy Graphics. These are detailed below

Colour	Copies graph to clipboard in full colour. The curve legends identify the curves using coloured squares similar to how the graph is displayed on the screen.
Monochrome	Copies graph to clipboard in monochrome. Curves are distinguished using varying markers and line styles. Curve legends distinguished curves with a straight line example

Colour with markers	Copies graph to clipboard in full colour but also differentiates curves using markers and line styles. Curve legends distinguished curves with a straight line example.

Paste Data from the Clipboard

Pulsonix Spice can plot curves using tabulated ASCII data from the clipboard. The format is the same as used for exporting data. See “Copy Data to the Clipboard” above for more details.

Using the Internal Clipboard

The menus **Edit | Copy, Edit | Cut and Edit | Paste** all use the internal clipboard. These menus are intended to allow the moving or copying of curves to new graphs. Note that these menus do not use the system clipboard at all. See above sections for details on how to copy and paste from the system clipboard.

The internal clipboard uses an efficient method for transferring curves that uses very little memory even if the curve is large. Also, if you copy a curve, the data itself is not copied internally; the two curves just reference the same data. This makes copying a memory efficient operation.

▶ To Move a Curve to a New Graph Sheet

1. Select the curve or curves you wish to move.
2. Select menu **Edit | Cut**.
3. Either create a new graph sheet to receive the new curves (use F10) or switch to an existing graph sheet.
4. Select menu **Edit | Paste**.

▶ To Copy a Curve to a New Graph Sheet

1. Select the curve or curves you wish to move.
2. Select menu **Edit | Copy**.
3. Either create a new graph sheet to receive the new curves (use F10) or switch to an existing graph sheet.
4. Select menu **Edit | Paste**.

Exporting Graphics

To export waveform graphics to a file, select the graph menu **File** | **Save Picture...** then select the format of your choice using the Save as type: drop down box. The choices are:

1. Windows Meta File (.EMF and .WMF). Nearly all windows applications that support graphics import will accept this format. Note that this is a scalable format and therefore suitable for high resolution printing.
2. Scalable Vector Graphics (.svg). This is a relatively new format and is not supported by many applications.
3. Bitmap - default image size (.png, .jpg, .bmp) These are widely supported by graphics applications but these are not scalable formats and so do not offer good

quality when printed using high resolution printers. PNG is the default format if you do not choose a file extension and generally this is the format that provides the best image quality/file size trade off. To choose JPG (JPEG format) or BMP (windows bitmap format) you must explicitly enter .jpg or .bmp file extensions respectively. With this option the image size will match the image size currently displayed on screen. If you wish to specify a different image size, use next option.

4. Bitmap - specify image size (.png, .jpg, .bmp). As 3 above but you must explicitly define the image resolution in pixels. You will be prompted for this when you close the file selection dialog box.

Saving Graphs

You can save a graph complete with all its curves, cursor settings and annotations to a binary file for later retrieval. Note that all the graph data is stored not just that needed for the current view. If a long run was needed to create the graph, the file could be quite large.

Saving

Select Command Shell menu **File | Graph | Save As**... or graph menu **File | Save As**... to save a graph that has never been saved before. To update a saved graph use Command Shell menu **File | Graph | Save** or graph menu **File | Save**

Restoring

Select Command Shell menu **File | Graph | Open**... or, if a graph window already exists, graph menu **File | Open**...

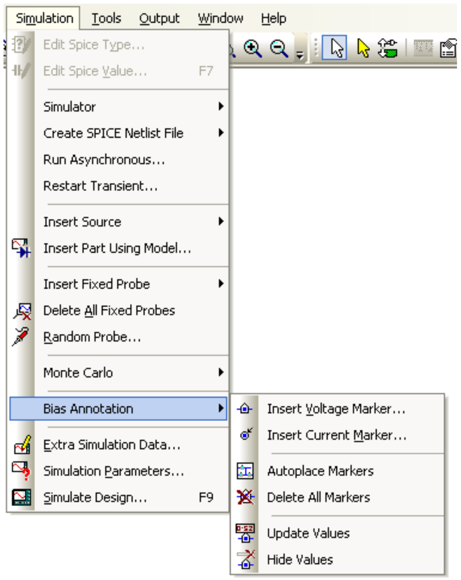
Viewing DC Operating Point Results

Schematic Annotation

Bias Markers

You can annotate the schematic with the results of a DC operating point analysis. This requires special markers to be placed on the schematic. You can instruct Pulsonix to place voltage markers at every node or you can place voltage or current markers manually.

Individual bias markers can be added from the Pulsonix SPICE library using the **Simulation menu** and **Bias Annotation** sub-menu. Bias Voltage Markers should have their pin attached to the net you wish to display the value for, and Bias Current Markers should have a connection from their pin to the device pin.



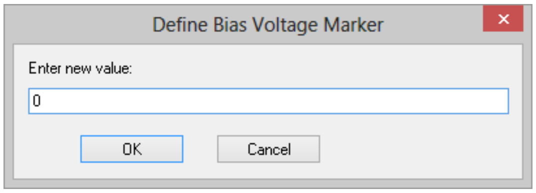
Use the **Insert Voltage/Current Markers** and **Autoplace Markers** options to automatically add a voltage marker component to each net in the schematic design. First all existing Markers on the Schematic Page will be removed. Then a Bias Marker will be added to all nets on the page. This does however clutter up the schematic and you may prefer to place them manually.

If you have already run a Simulation, you may now update the values displayed on the markers by using the **Update Values** option. These values are automatically updated after each simulation run.

Use the **Delete All Markers** option to automatically remove all Bias Voltage Markers from the current page of Schematic design.

Use the **Hide Values** option from the **Simulation** menu to automatically hide the values on all Bias Markers.

You may hide and restore values individually by selecting markers and using the **<F7>** option.



0

The values displayed on all the markers can be restored using the **Update Values** option.

List File Data

A great deal of information about each device in the circuit can be obtained from the list file. Use Command Shell menu **Graphs and Data | View List File** or **Graphs and Data | Edit List File** to see it.

Other Methods of Obtaining Bias Data

You can also display a voltage or current in the Pulsonix Spice Command Shell without placing bias markers on the schematic.

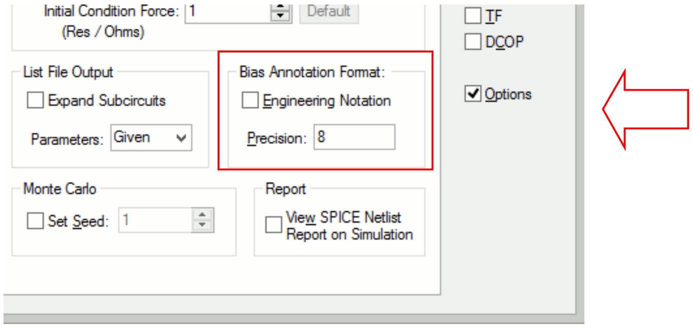
After running simulation, select the Pulsonix **Random Probe** option from the **Simulation** menu.

A probe cursor will appear indicating that you are using the interactive Random Probe tool. The current probe type is displayed on the right hand side of the Status Bar. A prompt stating what type of item you can pick is displayed on the left hand side of the Status Bar. As you move the cursor over the design, nets and pins that are suitable for probing with the current probe type are highlighted.
Right click to use the shortcut menu to change the type of probe you are using to **Show Net Voltage** or **Show Pin** Current.

Left click on the net or pin to show the bias point information in the Command Shell window.

Bias Annotation Display Precision

By default, bias annotation values are displayed with a precision of 6 digits. To change this use the **Simulation Parameters** menu from the schematic **Simulation** menu. On the **Options** page you will find the **Bias Annotation Format** section.


Initial Condition Force: 1
(Res / Ohms)
Bias Annotation Format:
Engineering Notation ☐
Precision: 8
Options ☑

Use these controls to alter the format that will be used for the DC Operating point values displayed on **Bias Voltage Markers** in the design. Enabling the **Engineering Notation** check box sets the format to use the standard multiplier suffixes (e.g. m=1e-3 u=1e-6 etc).
Simulation will need to be run again to change the values shown in the design.

Bias Annotation and Long Transient Runs

If you are running a long transient analysis and plan to use bias annotation extensively, you might like to set a simulator option that will make this process more efficient. The simulator option is:

```
.OPTIONS FORCETRANOPGROUP
```

This forces a separate data group and separate data file to be created for the transient analysis bias point data. Unless tstart>0 bias point data is usually taken from t=0 values. The problem with this approach is that to view a single value, the entire vector has to be loaded from the data file to memory. This isn't a problem if the run is only a 100 points or so, but could be a problem if it was 100,000 points. It can take a long time to load that amount of data. By specifying this option, the bias point data is stored separately and only a single value needs to be read from the file. This is much more efficient.

Future versions may set this option by default.

Saving Data

Saving the Data of a Simulation

The simulator usually saves all its data to a binary file stored in a temporary location. This data will eventually get deleted. To save this data permanently, select menu **File | Data | Save**.... You will be offered two options:

Move existing data file to new location

This will move the data file to location that you specify and thus change its status from temporary to permanent. As long as the new location is in the same volume (=disk partition) as the original location, this operation will be very quick. However, if the data is from the most recent simulation, Pulsonix Spice needs to ‘unhook’ it in order to be able to move the file. This will make it impossible to resume the simulation (if paused) or restart the simulation (transient only).

Note that if you specify a location on a different volume as the original data, then the file’s data has to be copied and for large data files, this will take a long time.

Make new copy

This makes a fresh copy of the data. This option does not suffer from the drawbacks of moving the file but if the data file is large can take a very long time.

Restoring Simulation Data

Select menu **File | Data | Lo**ad.... Navigate to a directory where you have previously saved data files.

You can also reload data from temporary files using menu **File | Data | Load Temporary Data**.... Whether or not there will be any files available to opened depends on the temporary data file delete options.

The error “The process cannot access the file because it is being used by another process” means that the temporary data file is still in use. Unless the file is in use by another instance of Pulsonix Spice you will be able to use its data by selecting its associated group. Use menu **Graphs and Data Analysis | Change Data Group**....

Performance Analysis and Histograms

Overview

When running multi-step analyses which generate multiple curves, it is often useful to be able to plot some characteristic of each curve against the stepped value. For example, suppose you wished to investigate the load response of a power supply circuit and wanted to plot the fall in output voltage vs transient current load. To do this you would set up a transient analysis to repeat a number of times with a varying load current. After the run is complete you can plot a complete set of curves, take cursor measurements and manually produce a plot of voltage drop vs. load current. This is of course can be quite a time-consuming and error prone activity.

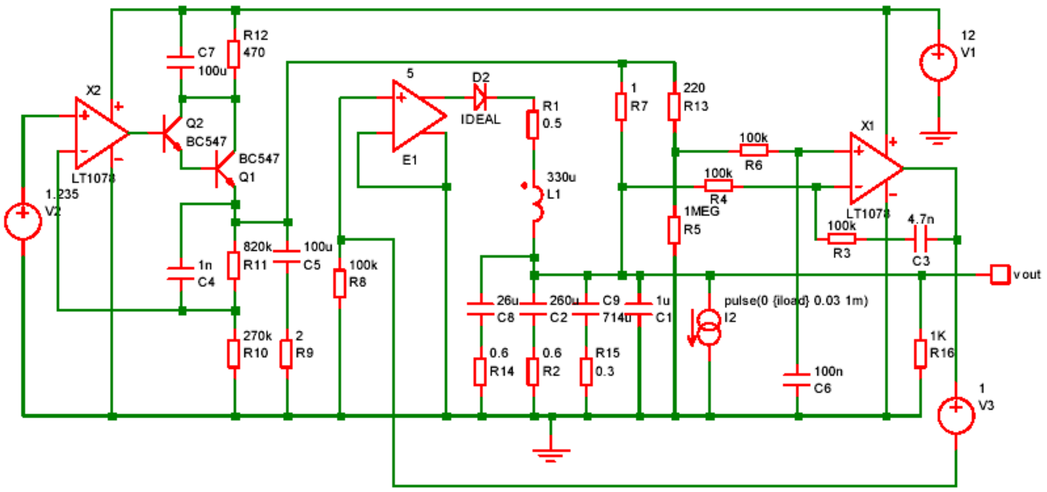
Fortunately Pulsonix Spice has a means of automating this procedure. A range of functions sometimes known as *goal functions* – are available that perform a computation on a complete curve to create a single value. By applying one or a combination of these functions on the results of a multi-step analysis, a curve of the goal function versus the stepped variable may be created.

This feature is especially useful for Monte Carlo analysis in which case you would most likely wish to plot a histogram.

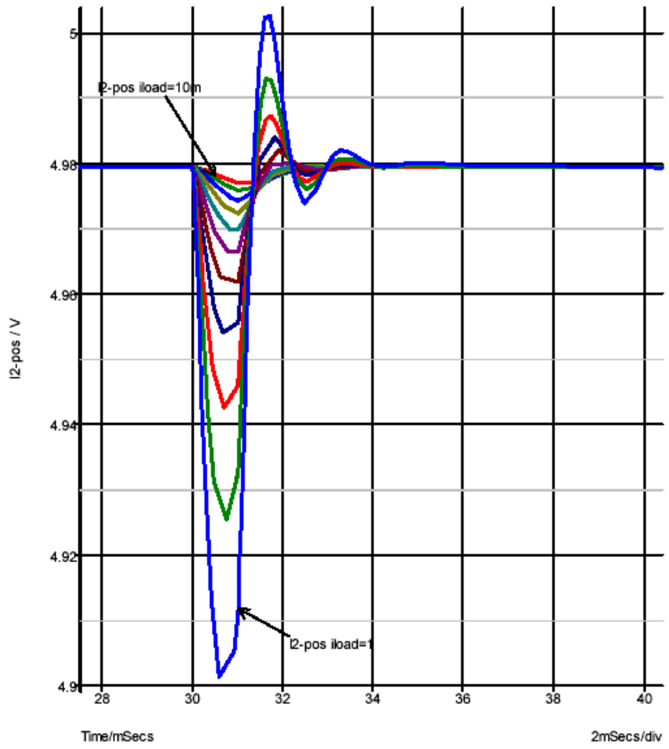
We start with an example and in fact it is a power supply whose load response we wish to investigate.

Example

The following circuit is a model of a hybrid linear-switching 5V PSU. See Examples\Spice\HybridPSU\5vpsu_v1.sxsch



I2 provides a current that is switched on for 1mS after a short delay. A multi-step analysis is set up so that the load current is varied from 10mA to 1A. The output for all runs is:

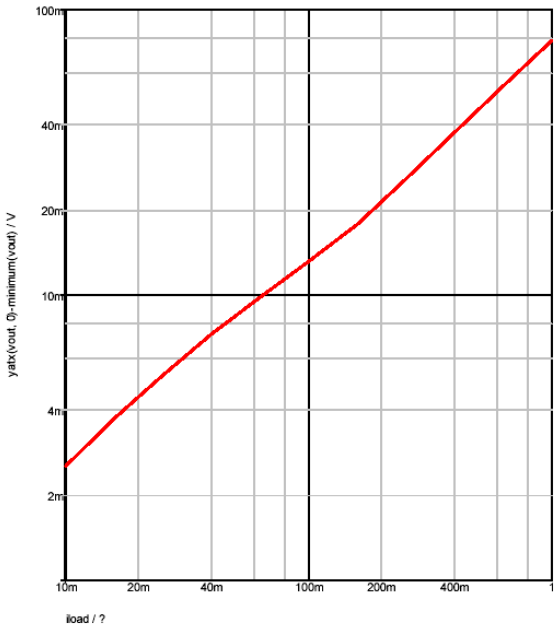


We will now plot a graph of the voltage drop vs the load current. This is the procedure:

1. Select **Random Probe** from the **Simulation** menu and use **Performance Analysis** from the shortcut menu.
2. You will see a dialog box very similar to that shown in “Plotting an Arbitrary Expression”. In the expression box you must enter an expression that resolves to a single value for each curve. For this example we use:

```
yatx (vout, 0) –minimum (vout)
```

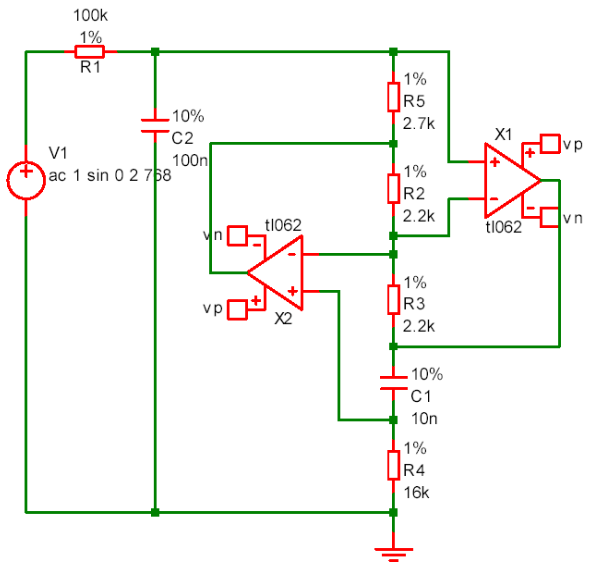
yatx(vout, 0) returns the value of vout when time=0. minimum(vout) returns the minimum value found on the curve. The end result is the drop in voltage when the load pulse occurs. Press **OK** and the following curve should appear:


100m
40m
20m
10m
4m
2m
yatx(vout, 0)-minimum(vout) / V
10m
20m
40m
100m
200m
400m
1
iload / ?

Histograms

The procedure for histograms is the same except that you should use the Simulation menu option MonteCarlo|Plot|Histogram… instead. Here is another example.

This is a design for an active band-pass filter using the simulated inductor method. See Examples\Spice\Montecarlo\768Hz_bandpass.sxsch. We want to plot a histogram of the centre frequency of the filter.

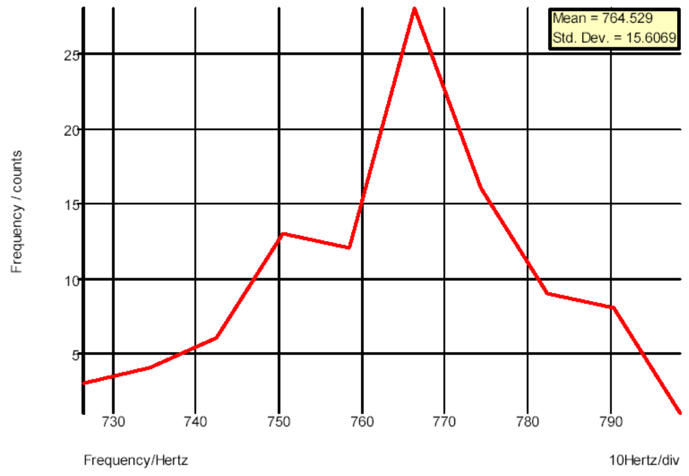


The example circuit has been set up to do 100 runs. This won’t take long to run, less than 10 seconds on most machines. This is the procedure:

1. Run the simulation using <**F9**> or **Simulate Design** option
2. Select menu Simulation | Monte Carlo | Plot\Histogram
3. On the schematic find the net name for the output of the filter. This is the junction of R1 and C2.
4. Enter the net name N0000 in the expression box. We must now modify this with a goal function that returns the centre frequency. The function CentreFreq will do this. This measures the centre frequency by calculating the half-way point between the intersections at some specified value below the peak. Typically you would use 3dB.

   Modify the value in the expression box so that it reads:

   ```
   CentreFreq (N0000, 3)
   ```
5. At this stage you can optionally modify the graph setting to enter your own axis labels etc. Now close the box. This is what you should see:

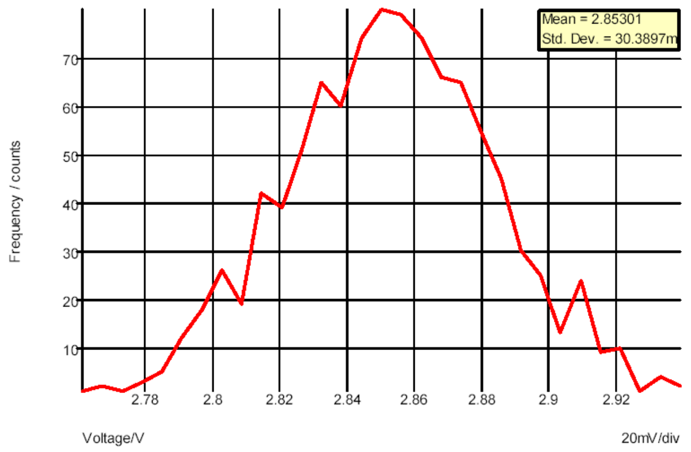


Note that the menu and standard deviation are automatically calculated.

Histograms for Single Step Monte Carlo Sweeps

An example of this type of run is shown in the AC Sweep section of the Analysis Modes chapter. These runs produce only a single curve with each point in the curve the result of the Monte Carlo analysis. With these runs you do not need to apply a goal function, just enter the name of the signal you wish to analyse.

1. Open the example circuit Examples\Spice\Sweep\AC_Param_Monte
2. Run simulation
3. Select menu Simulation|MonteCarlo|Plot Histogram
4. Type VPOS_Uneg in the expression box. “Vpos” is the net on the “+” pin of the differential amplifier “E1” and “Vneg” is the net on the “-“ pin of E1.
5. Press **OK**. You should see something like this:



This is a histogram showing the distribution of the gain of the amplifier at 100kHz.

Goal Functions

A range of functions are available to process curve data. Some of these are *primitive* and others use the user defined function mechanism. Primitive functions are compiled into the binary executable file while user defined functions are defined as scripts and are installed at functions at start up. User defined functions can be modified and you may also define your own. For more information refer to the *Script Reference Manual*.

The functions described here aren't the only functions that may be used in the expression for performance analysis. They are simply the ones that can convert the array data that the simulator generates into a single value with some useful meaning. There are many other functions that process simulation vectors to produce another vector for example: log; sqrt; sin; cos and many more. These are defined in "Function Reference" in the Simulators help pages.

Of particular interest is the Truncate function. This selects data over a given X range so you can apply a goal function to work on only a specific part of the data.

Primitive Functions

The following primitive functions may be used as goal functions. Not all actually return a single value. Some return an array and the result would need to be indexed. Maxima is an example.

Name	**Description**
Maxima(real [, real, string])	Returns array of all maximum turning points.
Maximum(real/complex [, real, real])	Returns the larges value in a given range.
Mean(real/complex)	Returns the mean of all values, (You should not use this for transient analysis data as it fails to take account of the varying step size. Use Mean1 instead.)
Mean1(real [, real, real]) interval	Finds the true mean accounting for the between data points.
Minima(real [, real, string])	Returns array of all minimum turning points.
Minimum(real/complex)	Returns the largest value in a given range.
RMS1(real [, real, real])	Finds RMS value of data
SumNoise(real [, real, real])	Integrates noise data to find total noise in the specified range.
XfromY(real, real [, real, real])	Returns an array of X values at a given Y value.
YfromX(real, real [, real])	Returns an array of Y values at a given X value.

User Defined Functions

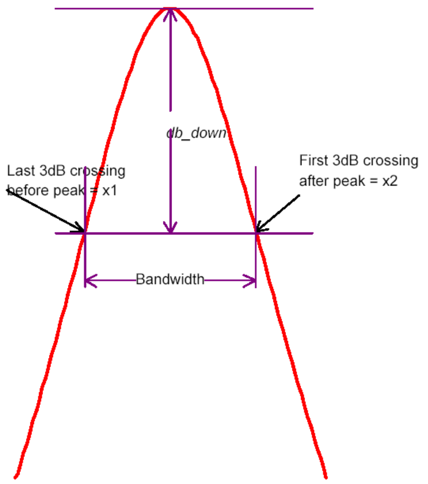
The following functions are defined using the user defined functions mechanism. They are defined as scripts but behave like functions.

Name	Description
BPBW(data, db_down)	Band-pass bandwidth.
Bandwidth(data, db_down)	Same as BPBW
CentreFreq(data, db_down)	Centre frequency
Duty(data, [threshold])	Duty cycle of first pulse
Fall(data, [start, end])	Fall time
Frequency(data, [threshold])	Average frequency
GainMargin(data, phaseInstabilityPoint)	Gain Margin
HPBW(data, db_down)	High pass bandwidth
LPVW(data, db_down)	Low pass bandwidth
Overshoot(data, [start, end])	Overshoot
PeakToPeak(data, [start, end])	Peak to Peak
Period(data, [threshold])	Period of first cycle
PhaseMargin(data, phaseInstabilityPoint)	Phase Margin
PulseWidth(data, [threshold])	Pulse width of first cycle
Rise(data, [start, end])	Rise time
XatNthY(data, yValue, n)	X value at the Nth Y crossing
XatNthYn(data, yValue, n)	X value at the Nth Y crossing with negative slope
XatNthYp(data, yValue, n)	X value at the Nth Y crossing with positive slope
XatNthYpct(data, yValue, n)	X value at the Nth Y crossing. y value specified as a percentage
YatX(data, xValue)	Y value at xValue
YatXpct(data, xValue)	Y value at xValue specified as a percentage

BPBW, bandwidth

BPBW9*data, db_down)*

Finds the bandwidth of a band pass response. This is illustrated by the following graph:


db_down
Last 3dB crossing before peak = x1
First 3dB crossing after peak = x2
Bandwidth

Function return x2-x1 as shown in the above diagram.

Note that *data* is assumed to be raw simulation data and may be complex. It must not be in dBs.

Implemented by built-in script uf_bandwidth. See install CD for source.

CentreFreq, CenterFreq

CentreFreq(*data, db_down)*

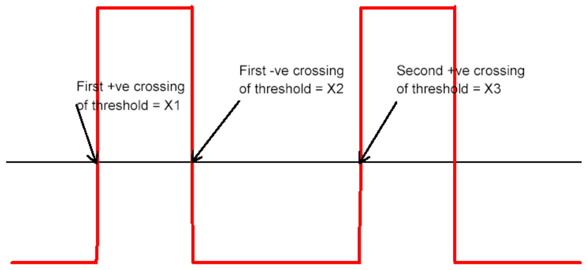
See diagram in "BPBW, Bandwidth" above. Function returns (x1_x2)/2

Both British and North American spellings of centre (center) are accepted.

Implemented by built-in script uf_centre_freq. See install CD for source.

Duty

Duty*(data*,[*threshold*])


First +ve crossing of threshold = X1
First -ve crossing of threshold = X2
Second +ve crossing of threshold = X3

Function returns (X2-X1)/(X3-X1)
X1, X2 and X3 are defined in the above graph.

Default value for *threshold* is

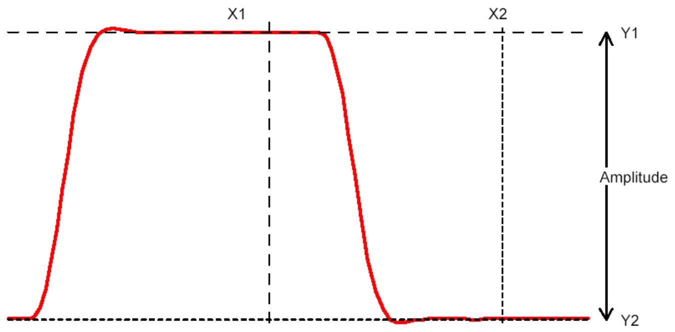
(Ymax+Ymin)/2

Where Ymax = largest value in *data* and Ymin is smallest value in *data*.

Implemented by built-in script uf_duty. See install CD for source.

Fall

Fall(*data*, [*xStart*, *xEnd*])



Function returns the 10% to 90% fall time of the first falling edge that occurs between x1 and x2. The 10% point is at y threshold Y1 + (Y2-Y1*0.1 and the 90% point is at y threshold Y1 _ (Y2-Y1)*0.9.

If *xStart* is specified, X1=*xStart* otherwise X1 = *x* value of first poin in data.
If *xEnd* is specified, X2-*xEnd* otherwise X2 = *x* value of last point in data.
If *xStart* is specified, Y1=*y* value at *xStart* otherwise Y1 = maximum *y* value in data.
If *xEnd* is specified, Y2=*y* value at *xEnd* otherwise Y2 = minimum *y* value in data.

Implemented by built-in script uf_fall. See install CD for source.

Frequency

Frequency(*data*, [*threshold*])

Finds the average frequency of *data.*

Returns:

$(n-1)/(x_n-x_1)$

Where:
n = the number of positive crossings of *threshold*
x_n = the x value of the nth positive crossing of *threshold*
x_1 = the x value of the first positive crossing of *threshold*

If *threshold* is not specified a default value of (ymax+ymin)/2 is used where ymax is the largest value in data and ymin is the smallest value.

Implemented by built-in script uf_frequency. See install CD for source.

GainMargin

GainMargin(*data*, [*phaseInstabilityPoint*])

Finds the gain margin in dB of *data* where *data* is the complex open loop transfer function of a closed loop system. The gain margin is defined as the factor by which the open loop gain of a system must increase in order to become unstable.

phaseInstabilityPoint is the phase at which the system becomes unstable. This is used to allow support for inverting and non-inverting systems. If *data* represents an inverting system, *phaseInstabilityPoint* should be zero. If *data* represents a noninverting system, *phaseInstabilityPoint* should be -180.

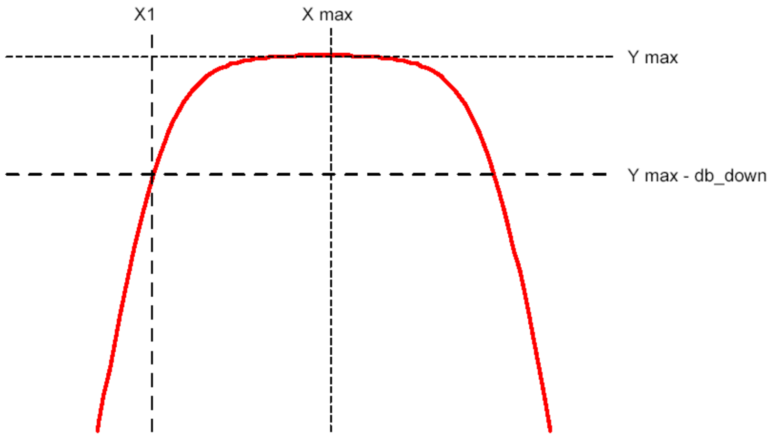
The function detects the frequencies at which the phase of the system is equal to *phaseInstabilityPoint*. It then calculates the gain at those frequencies and returns the value that is numerically the smallest. This might be negative indicating that the system is probably already unstable (but could be conditionally stable).

If the phase of the system does not cross the *phaseInstabilityPoint* then no gain margin can be evaluated and the function will return an empty vector.

HPBW

HPBW(*data, db_down*)

Finds high pass bandwidth.



Returns the value of X1 as shown in the above diagram.

Y max is the y value at the maximum point.
X max is the x value at the maximum point.
X1 is the x value of the first point on the curve that crosses (Y max – db_down) starting at X max and scanning *right to left*.

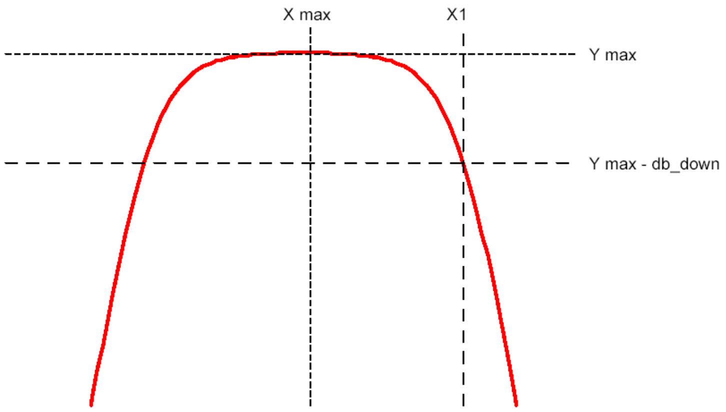
Note that *data* is assumed to be raw simulation data and may be complex. It must not be in dBs.

Implemented by built-in script uf_hpbw. See install CD for source.

LPBW

LPBW(*data, db_down*)

Finds low pass bandwidth



Returns the value of X1 as shown in the above diagram.

Y max is the y value at the maximum point.
X max is the x value at the maximum point.
X1 is the x value of the first point on the curve that crosses (Y max – db_down) starting at X max and scanning left to right.

Note that *data* is assumed to be raw simulation data and may be complex. It must not be in dBs.

Implemented by built-in script uf_lpbw. See install CD for source.

Overshoot

Overshoot(*data*, [*xStart*, *xEnd*])

Finds overshoot in percent.

Returns:

(yMax-yStart)/(yStart-yEnd)

Where

yMax is the largest value found in the interval between *xStart* and *xEnd*.
yStart is the y value at *xStart*.
yEnd is the y value at *xEnd*.

If *xStart* is omitted it defaults to the x value of the first data point.
If *xEnd* is omitted it defaults to the x value of the last data point.

Implemented by built-in script uf_overshoot. See install CD for source.

PeakToPeak

PeakToPeak(*data*, [*xStart*, *xEnd*])

Returns the difference between the maximum and minimum values found in the data within the interval *xStart* to *xEnd*.

If *xStart* is omitted it defaults to the x value of the first data point.
If *xEnd* is omitted it defaults to the x value of the last data point.

Implemented by built-in script uf_peak_to_peak. See install CD for source.

Period

Period(*data*, [*threshold*])

Returns the period of the data.

Refer to diagram for the "Duty" function. The Period function returns:

X3 – X1

Default value for *threshold* is

(Ymax+Ymin)/2

Where Ymax = larges value in *data* and Ymin in smallest value in *data*.

Implemented by built-in script uf_period. See install CD for source.

PhaseMargin

PhaseMargin(*data*, [*phaseInstabilityPoint*])

Finds the phase margin in dB of *data* where *data* is the complex open loop transfer function of a closed loop system. The phase margin is defined as the angle by which the open loop phase shift of a system must increase in order to become unstable. *phaseInstabilityPoint* is the phase at which the system becomes unstable. This is used to allow support for inverting and non-inverting systems. If *data* represents an inverting system, *phaseInstabilityPoint* should be zero. If *data* represents a non-inverting system, *phaseInstabilityPoint* should be -180.

The function detects the frequencies at which the magnitude of the gain is unity. It then calculates the phase shift at those frequencies and returns the value that is numerically the smallest. This might be negative indicating that the system is probably already unstable (but could be conditionally stable).

If the gain of the system does not cross unity then no phase margin can be evaluated and the function will return an empty vector.

PulseWidth

PulseWidth(*data*, [*threshold*])

Returns the pulse width of the first pulse in the data.

Refer to diagram for the "Duty" function. The PulseWidth function returns:

X2 – X1

Default value for *threshold* is

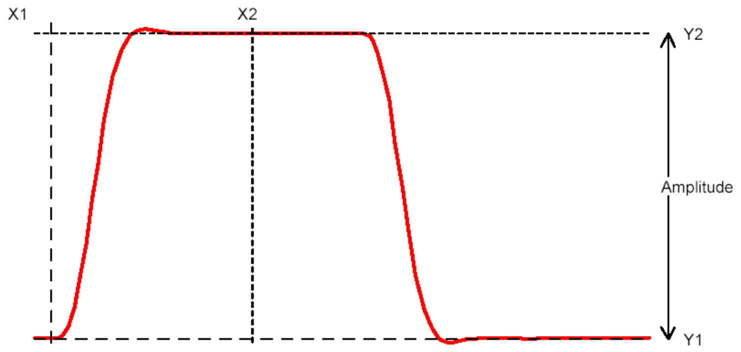
(Ymax+Ymin)/2

Where Ymax = largest value in *data* and Ymin in smallest value in *data*.

Implemented by built-in script uf_pulse_width. See install CD for source.

Rise

Rise(*data*, [*xStart*, *xEnd*])



Function returns the 10% to 90% rise time of the first rising edge that occurs between x1 and x2. The 10% point is at y threshold Y1 + (Y2-Y1)*0.1 and the 90% point is at y threshold Y1 + (Y2-Y1)*0.9.

If *xStart* is specified, X1=*xStart* otherwise X1 = x value of first poin in data.
If *xEnd* is specified, X2=*xEnd* otherwise X2 = x value of last point in data.
If *xStart* is specified, Y1=y value at *xStart* otherwise Y1 = maximum y value in data.
If *xEnd* is specified, Y2=y value at *xEnd* otherwise Y2 = minimum y value in data.

XatNthY

XatNthY(*data*, *yValue*, *n*)

Returns the x value of the data where it crosses *yValue* for the *n*th time.

XatNthYn

XatNthYn (*data*, *yValue*, *n*)

Returns the x value of the data where it crosses *yValue* for the *n*th time with a negative slope.

XatNthYp

XatNthYp(*data*, *yValue*, *n*)

Returns the x value of the data where it crosses *yValue* for the *n*th time with a positive slope.

XatNthYpct

XatNthYpct(*data*, *yValue*, *n*)

As XatNthY but with *yValue* specified as a percentrage of the maximum and minimum values found in the data.

YatX

YatX(*data*, *xValue*)

Returns the y value of the data at x value = *xValue.*

YatXpct

As YatX but with *xValue* specified as a percentage of the total x interval of the data.

Data Import and Export

Pulsonix Spice provides the capability to export simulation data to a file in test form and also to import data from a file in text form. This makes it possible to process simulation data using another application such as a spreadsheet or custom program.

Pulsonix Spice may also import data in SPICE3 raw file format and CSDF format. Some other simulation products can output in one or both of these formats.

Importing SPICE3 Raw and CSDF Files

▶ To import Raw Spice files

1. Select command shell menu **File|Data|Load…**
2. In **Files of type** select **SPICE3 Raw Files** or **CSDF Files** as required.
3. Select file to import

Pulsonix Spice will read the entire file and write its data out to a temporary .sxdat file in the same way as it does when saving its own simulation data.The data read from the raw file is buffered in RAM in order to maximise the efficiency of the saved data. Pulsonix Spice will use up to 10% of system RAM for this purpose.

Importing Data

Importing Tabulated ASCII Data

Pulsonix Spice can import data in a tabulated ASCII format allowing the display of data created by a spreadsheet program. There is a no menu for this, but this can be done using the OpenGroup command with the /text switch. E.g. at the command line type:

```
OpenGroup /text data.txt
```

This will read in the file data.txt and create a new group called text*n*. See “Data Files Text Format” below for details of format.

Note that if you create the file using another program such as a spreadsheet, the above command may fail if the file is still open in the other application. Closing the file in the other application will resolve this.

Exporting SPICE3 Raw Files

Pulsonix Spice can export all simulation data to a SPICE3 raw file. This format may be accepted by third party waveform viewers.

To export a SPICE3 raw file, proceed as follows:

1. Select menu **File|Data|Save…**
2. Under **Save as type:** choose “SPICE3 Raw Files”.

Note that various applications use slightly different variants of this format. By default, Pulsonix Spice outputs the data in a form that is the same as the standard unmodified SPICE3 program. This can be modified using the option setting “ExportRawFormat”. Use the Set command to set this value. Set this value to ‘spice3’, ‘spectre’ or ‘other’.

Exporting Data

To export data, use the Show command with the /file switch. E.g.

```
Show /file data.txt vout r1_p q1#c
```

Will output to data.txt the vectors vout, r1_p, and q1#c. The values will be output in a form compatible OpenGroup /text.

Launching Other Applications

Data import and export makes it possible to process simulation data using other applications. Pulsonix Spice has a facility to launch other programs using the Shell command. You could therefore write a script to export data, process it with your own program then read the processed data back in for plotting. To do this you must specify the /wait switch for the Shell command to force Pulsonix Spice to wait until the external application has finished. E.g.

```
Shell /wait procdata.exe
```

will launch the program procdata.exe and will not return until procdata.exe has closed.

Data Files Text Format

Pulsonix Spice has the ability to read in data in text form using the OpenGroup command. This makes it possible to use Pulsonix Spice to graph data generated by other applications such as a spreadsheet. This can be useful to compare simulated and measured results.

There are two alternative formats.
The first is simply a series of values separated by white space. This will be read in as a single vector with a reference equal to its index.

The second format is as follows:

A text data file may contain any number of *blocks*. Each *block* has a *header* followed by a list of *datapoints*. The *header* and each *datapoint* must be on one line.

The *header* is one of the form:
reference_nameydata1_name[*ydata2_name …]*

Each *datapoint* must be of the form:
reference_valueydata1_value[*ydata2_value …*]

The number of entries in each *datapoint* must correspond to the number of entries in the *header*.
The *reference* is the x data (e.g. time or frequency).

Example

Tvol	Voltage1	Voltage2
`Time`		
0	14.5396	14.6916
1e-09	14.5397	14.6917
2e-09	14.5398	14.6917
4e-09	14.54	14.6917
8e-09	14.5408	14.6911
1.6e-09	14.5439	14.688
3.2e-08	14.5555	14.6766
6.4e-08	14.5909	14.641
1e-07	14.6404	14.5905
1.064e-07	14.6483	14.5821

If the above was read in as a text file (using OpenGroup /text), a new group called textn where n is a number would be generated. The group would contain three vectors called time, V*oltage1* and *Voltage2.* The vectors *Voltage1* and *Voltage2* would have a *reference* of *Time. Time* itself would not have a *reference.*

To read in complex values, enclose the real and imaginary parts in parentheses and separate with a comma. E.G.

Frequency	:VOUT
1000	(-5.94260997, 0.002837811)
1004.61579	(-5.94260997, 0.00285091)
1009.252886	(-5.94260996, 0.002864069)
1013.911386	(-5.94260995, 0.002877289)
1018.591388	(-5.94260994, 0.00289057)
1023.292992	(-5.94260993, 0.002903912)
1028.016298	(-5.94260992, 0.002917316)
1032.761406	(-5.94260991, 0.002930782)
1037.528416	(-5.9426099, 0.00294431)
1042.317429	(-5.94260989, 0.0029579)
1042.128548	(-5.94260988, 0.002971553)

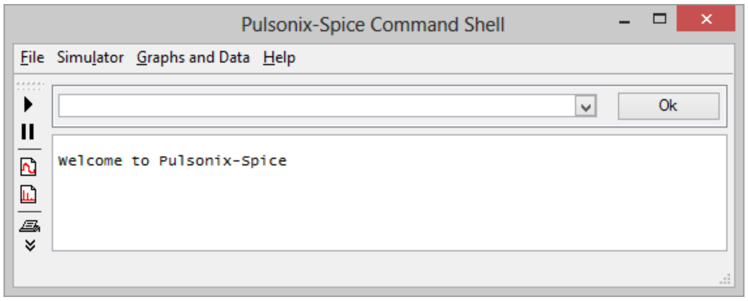
Function Reference

All functional references can be found in the Pulsonix Spice online help.

Chapter 8. Command Shell

Command Line

The command line is at the top of the Command Shell.


Pulsonix-Spice Command Shell
File Simulator Graphs and Data Help
Ok

```
Welcome to Pulsonix-Spice
```

The vast majority of operations with **Pulsonix Spice** can be executed from menus or pre-defined keys and do not require the use of the command line. However, a few more advanced operations do require the use of the command line. From the command line you can run a script or an internal command. You can also define a new menu to call a script, command or series of commands. In fact all the built in menu and keys are in fact themselves defined as commands or scripts. These definitions can be changed as well as new ones defined.

Details of some of the available commands are given later in the chapter The remainder are documented in the *Pulsonix Spice Script Users Guide* supplied installed as a PDF file and accessed via the simulator's hep menu.

Command History.

A history of manually entered commands is available from the drop down list. (select arrow to the right of the command line). Some commands entered via the menus are also placed in the command history. This allows you to repeat the command easily. Although all menus are executed via commands only a few are placed in the history.

Multiple commands on one line

You can place multiple commands on the same line separated by a semi-colon - ; This is the only way a menu or key can be defined to execute more than one command.

Scripts

Pulsonix Spice features a comprehensive scripting language. Full details of this are supplied with the "Pulsonix Spice Script Users Guide" installed into the Spice directory.

Command line editing.

The command line itself is a windows edit control. The cursor keys, home and end all work in the usual way. You can also copy (Ctrl-C), cut (Ctrl-X) and paste (Ctrl-V) text. There is also a right click popup menu with the usual edit commands.

Maximum line length

There is a maximum number of characters that can be entered on the command line but this is very large. The limit is related to the number of words as well as characters and is typically in excess of 2000 words. It is possible to exceed this limit with *braced substitutions* where the result of an expression is placed in the command line. Note that lines larger than 512 characters will not be inserted into the command history. (The drop down list box attached to command line).

Command Line Switches

Many commands have *switches*. These are always preceded by a '/' and their meaning is specific to the command. There are however four global switches which can be applied to any command. <u>These must always be placed immediately after the command</u>. Global switches are as follows:

1. /e Forces command text to copied to command history
2. /ne Inhibits command text copying to command history
3. /quiet Inhibits error messages for that command. This only stops error message being displayed. A script will still be aborted if an error occurs but no message will be output
4. /noerr Stops scripts being aborted if there is an error. The error message will still be displayed

Message Window

The *message window* is the main window within the command shell. The majority of messages, including errors and warnings, are displayed here. The window can be scrolled vertically with the scroll bar.

Up to 2000 lines of messages will be retained for viewing at any time.

You can copy a line of text from the *message window* to the command line by placing the cursor on the line and either double clicking the left mouse key or pressing the Insert key.

You can clear the message window using **File | Windows | Clear Messages**.

Colours and Fonts

Colours

Colours for graph curves and graph grids may be customised using the colour dialog box. This is opened using the **File Options | Colour** menu item.

Select the object whose colour you wish to change then select **Edit** button to change it. The colours you select are stored persistently and will remain in effect for future sessions of Pulsonix Spice.

Fonts

Fonts for various components of Pulsonix Spice may be selected using the font selection dialog box. This is opened using the **File | Options | Font**... menu item.

Select the item whose font you wish to change the press **Edit** to select new font. Items available are:

Font object name	Where font used
Command Line	Command line at top of **Command Shell**
Graph	Graph windows
Graph Caption	Graph Caption objects placed using **Annotate | Add Caption**
Graph Free Text	Graph Free Text object placed using **Annotate | Add Free Text**
Legend Box	Graph Legend Box object placed using **Annotate | Add Legend Box**
Message Window	Bottom part of Command Shell
Print Caption	Font used at base of printed graph
View File Window	Window opened for viewing files - such as the simulator list file or netlist file
View Model Text	Model display window in View model dialog box.

Saving and Restoring Sessions

Overview

You can save the current session for later restoration. This is useful in the situation where you are in the middle of studying simulation results, but you need to interrupt this work maybe at the end of a working day. While in some situations you might simply be able to leave your computer switched on and logged in, or maybe use a “Hibernate” mode, these methods are not always practical or indeed reliable.

The Pulsonix Spice save session feature will save the current state of all open graphs and any simulation data so that it can be restored at a later time.

Saving a Session

Select menu File | Save Session.

Restoring a Session

You can only restore a session if all graphs are closed and there is no current simulation data loaded. This is the normal state when Pulsonix Spice has just been started. If you wish to restore a session when Pulsonix Spice has been in use since first starting, you can either shut down and restart, or close all windows and graphs then select menu Graphs and Data | Delete Data Group..., press Select All, then Ok.

To restore the session, select menu File | Restore Session

Where is Session Data Stored?

Session data is stored in the following directory:

application_data/session

where *application_data* is the Pulsonix-Spice application data directory.

Configuration Settings

Overview

This simulator, in common with most applications, needs to store a number of values that affect the operation of the program. These are known as configuration settings. Included among these are the locations of installed symbol libraries, installed model libraries, font preferences, colour preferences and default window positions.

Default Configuration Location

By default, the simulator stores configuration settings in a single file. This file is located at: *simetrix_app_data_dir*\config\Base.sxprj (windows)

See “Application Data Directory” below for location of *simetrix_app_data_dir*.

Application Data Directory

Pulsonix Spice stores a number of files in its *application data directory*.

This location is as follows:

sys_application_data_dir\Pulsonix\Pulsonix_Spice*xx* ((full production versions)
sys_application_data_dir\Pulsonix\Pulsonix_SpiceIntro*xx* (*demo version*)

where *xx* is a three digit code representing the simulator version number, e.g. 540 for version 5.40. *sys_application_data_dir* is a system defined location.

The following table shows typical locations for all supported Windows systems:

Operating System

Windows 2000	C:\Documents and Settings\username\Application Data Windows XP
Windows Vista	C:\Users\username\AppData\Roaming

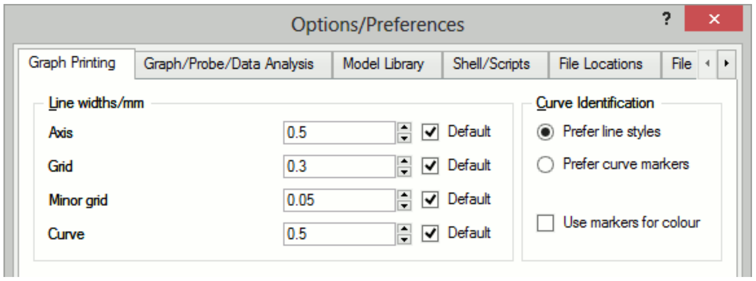
username is the log on name currently being used. The above are only typical locations on English language versions of Windows. The user or system administrator may move them and also the names used may be different for non-English versions of Windows.

Options

There are a number of options affecting all aspects of Pulsonix Spice**.**

Using the Options dialog

This is invoked from the **Command Shell** window and the **File** menu option **Options** and **General**



Line widths/mm

Axis	0.5	[✓ Default]
Grid	0.3	[✓ Default]
Minor grid	0.05	[✓ Default]
Curve	0.5	[✓ Default]

Curve Identification

● Prefer line styles
○ Prefer curve markers
☐ Use markers for colour

Graph Printing

Axis line width	Width in mm of printed axis
Grid line width	Width in mm of printed grid lines
Curve line width	Width in mm of printed curves
Curve identification	When printing on monochrome printers or if "Use markers for colour" is selected, curves are differentiated using different line styles (solid, dashed etc.) and marker shapes (circles, squares etc.). For a large number of curves both methods are used but for just a few you can use this option to state your preference
Prefer line styles	Printed curves will first be differentiated using line styles
Prefer curve markers	Printed curves will first be differentiated using curve markers
Use markers for colour	Even if printing to a colour printer, curves will still be printed using markers and variable line styles to differentiate them. They will also be printed in colour. This is useful when creating on-line documents (e.g. using Adobe PDFWriter) which might subsequently be viewed on-line or printed out.

Graph/Probe/Data Analysis

Probe update times	Plots created from fixed probes are updated on a regular basis. This controls how frequently and when it starts.
Period	Update period in seconds
Start	Start delay in seconds
Temporary data file delete	Simulation data is stored in data files that are placed in the "Temp data directory" (see file locations below). These options control when these data files are deleted.

Never	Temporary files are never deleted but will be overwritten in subsequent sessions. Not recommended unless you only ever do short simulations.
When program starts	All temporary files are deleted when Pulsonix Spice starts.
When program closes	All temporary files are deleted when Pulsonix Spice is shut down. While using Pulsonix Spice, you can recover earlier simulation runs. Normally, only the 3 most recent are kept but earlier ones can be recovered from the TEMPDATA directory using File|Data|Load...
When data is no longer needed	This is the most aggressive delete method and is recommended if you do many long runs or/and have limited disc space. By default, the 3 most recent runs are kept but with the other options above, the data files are not deleted when the data is not needed but links to the data in them are released. (See explanation below). If this option is set, the data files are deleted as soon as they become out of date, optimising use of disc space at the expense of not being able to recover old data.
Sizes	
Curve weight	Thickness of displayed curves. Curves display much quicker if this value is set to 1 but are clearer (but can lose detail) if set to 2.
Digital Axis Height	Sets height of axes (in mm) used to plot digital traces.
Min grid height	When a grid is added to a graph window, existing grids are reduced in height to accommodate the new one. But they won’t be reduced to a height lower than specified by this setting. When this limit is reached, the vertical space will be increased by allowing the window to scroll.
Cursor readout	Controls where cursor values are displayed.
On graph	Values are displayed on the graph itself
Status bar	Values are displayed in status bar boxes at the bottom of the graph window
Both	Displayed in both locations as described above.
Histogram style	Controls the curve style used for histogram displays:
Stepped	Displays a flat line for the width of each bin. Similar to a bar graph
Smooth	Joins the centre of each bin with a straight line to display a continuous curve

How data is stored. For info only

Simulation data is stored in temporary data files as explained above. The data is not read into system memory until it is needed - say - to plot a graph. However, the location in the file of the various vectors is always in memory so that the data can be extracted from the file as rapidly as possible. It is this latter location data that is destroyed when *a simulation run gets out of date. The file containing the data gets deleted at a time set by the above options, not necessarily when the data is "no longer needed". As long as the file exists, the data can be recovered by using File|Data|Load* or *File|Data|Load Temporary Data which re builds the location data.*

Model Library

Library Diagnostics	Determines whether messages are displayed when models are found in the library.
Model read in	"Don't abort on unknown parameters". If cleared, the simulation will abort if models are found that have parameters unsupported by Pulsonix Spice

File Locations

Locations in your files system of various files and folders needed for correct operation of Pulsonix Spice.

Scripts	Location of script folder used to hold any scripts you run. Changing this has no effect until Pulsonix Spice is restarted.
Start Up Script	Name of script that is automatically run on start up. You can place custom key definitions in this file.
Built-in Scripts	First location that Pulsonix Spice searches for scripts. Internal scripts can be overwritten by placing scripts of the same name in this folder. Changing it has no effect until Pulsonix Spice is restarted.
Start up	Current working directory on start up.
Temp Data	Location of temporary simulation data files. Changing this setting has no effect until you restart Pulsonix Spice. Note this must always be a local folder.
Editor	Text editor called by EditFile command as used by a number of menus. Default is notepad.

Shell/Scripts

Script Options	
Echo all messages	If set, all script lines will be displayed in the message window. This will result in a great deal of output and will slow down the whole program operation
Don't abort scripts on error	Normally scripts abort if an error is detected. Check this box to disable this behaviour.
Menu/Keys disable	The built-in menu and key definitions can be disabled, allowing you to define your own.
Disable standard menus	Disables menu definitions and any key defined as shortcuts to a menu. Does not take effect until you restart Pulsonix Spice.
Disable standard key definitions	Disable key definitions. Note that many of the key assignments are defined as menu shortcuts (their name appears in the menu text). These are not disabled by this option. Does not take effect until you restart Pulsonix Spice.

File Extensions

Defines extensions used for the various files used by Pulsonix Spice.

Note that the default settings for three file types have five letters rather than the usual three. This is to avoid conflict with other applications where file associations are involved. Some file servers on network systems do not support five letter extensions. If you are using such a system you will need to change these settings.

For each setting, the supported extensions are separated by a semi-colon. The first in the list is the default. So, for example, the default data file extension is sxdat so when you save a simulation data without giving an extension, it will automatically be given the extension sxdat.

Data Files	Extensions used for simulation data files
Text Files	Supported extensions for text files. (**File|Scripts|Edit File** will list all files with these extensions)
Logic Def. Files	Files used for logic definitions for the arbitrary logic block. If the extension is omitted in the model (FILE parameter) this will be used.
Device Models	SPICE model files. For future expansion, currently unused.
Device Catalogs	Extension for Device Catalogs used by the associate model and part system.
Scripts	Default extension for scripts if called without an extension.

Using the “Set” and “Unset” commands

All options have a name and many also have a value. These are set using the Set command and can be cleared with UnSet. When an option is cleared it is restored to its default value. A complete listing of available options with possible values is given below. Note that option settings are *persistent*. This means that their values are stored in the system registry and automatically restored at the start of each subsequent **Pulsonix Spice** session.

List of Options

Upper and lower case letters have been used in the following listing only for clarity. Option names and their values are not in fact case sensitive. Many of the options described below are supported by the Options dialog box in which case they are noted accordingly.

Unsupported Options

Some options in the following list are marked as ‘unsupported’. This means that they may be withdrawn in the future or their functionality changed.

Name	**Type**	**Default value**	**Description**	**User interface support**
AxisPrintWidth	Numeric	0.5mm	Width of printed axis in mm. See also CurvePrintWidth and GridPrintWidth	Options dialog
BiScriptDir	Text		This is the first location that Pulsonix Spice searches for scripts. See File Locations section of options dialog above for more info.	Options dialog
CloseSimStatus	Boolean	off	If set, simulator status dialog box is closed on completion of run. This option is automatically set if the "Close on completion" check box in this dialog box is enabled.	Automatic
CursorColour	Text	black	Possible values: "inverted", "white" and "black". This refers to the colour of the cross-hair cursor used for the graph window. "inverted" means that the cursor will be the inverse of the background. This option takes effect on next session. (SETUP program sets this to "inverted" at installation)	No
CursorDisplay	Text	Graph	Controls initial graph cursor readout display. *Graph* Display on graph only *StatusBar* Display on status bar only *Both* Display on both graph and status bar	Options dialog
CurvePrintWidth	Numeric	0.50mm	Width of printed graph curves in mm. See also GridPrintWidth.	Options dialog
CurveWeight	Numeric	1	Sets the line width in pixels of graph curves. Note that although widths greater than 1 are clearer they normally take considerably longer to draw. This does however depend on the type of adapter card and display driver you are using.	Options dialog

DataGroupDelete	Text	OnStart	Determines when temporary simulation data is deleted. Possible values, "Never", "OnStart", "OnClose" and "OnDelete". See Graph/Probe/Data Analysis tab of options dialog for details.	Options dialog
Degrees	Boolean	off	Affects whether trig functions (such as ph() for phase) work in degrees or radians. (Although default is off, it is automatically switched on the first time the program is run after installation)	No
DigAxisHeight	Numeric	0.8mm	Height of digital axis in mm. (screens are typically 75pixels/inch)	Options dialog
DisplaySimProgressMessage	Boolean	flse	If true, a message will be displayed in the command shell indicating the start and end of a simulation.	
EchoOn	Boolean	off	When set, all commands are echoed to the *message* window. This is used primarily for script debugging.	Options dialog
Editor	Text	Notepad	Default text editor.	Options dialog
ExportRawFormat	Text	SPICE3	Possible values, SPICE3, SPECTRE and OTHER. Controls format of raw output. See “Exporting SPICE3 Raw Files” section.	No
GraphPrintLandscape	Boolean	off	Graph print page orientation set to Landscape. This is set/cleared automatically by the Print Graph Setup... dialog box.	Automatic
GridPrintWidth	Numeric	0.30mm	Width of printed graph grid lines in mm. See also CurvePrintWidth	Options dialog
GroupPersistence	Numeric	3	Sets the number of groups that are kept before being deleted.	No
HighlightIncrement	Numeric	1	Highlighted graph curves are thicker than normal curves by the amount specified by this option	No
HistoCurveStyle	Text		Sets histogram curve style.	Options dialog
InvertCursors	Boolean	false	If true, graph mouse cursors are modified to be suitable for use on a black background	

InterpOrder	Numeric	2	Interpolation order for FFT menus.	Options dialog
InterpPts	Numeric	512	Number of interpolated points for FFT menus.	Options dialog
LibFile*n* (*n* can be any number. E.g. LibFile100. *n* may also be omitted)	Text	none	Name of file that will be searched if any model or subcircuit name could not be found in the netlist or cache. The file specified by "LibFile" will be searched before any files specified with the .lib simulator control.	Model library installation
LibraryDiagnostics	Text	Full	Possible values, "Partial", "None" and "Full". Affects progress information displayed during model library searching.	Options dialog
MinorGridPrintWidth	Numeric		Print width in mm of dialog graph’s minor grid.	Options dialog
MRUSize	Numeric	4	Number of items in File|Reopen menu.	Options dialog
NoAutoFile	Boolean	off	If specified, all simulation data will be directed to RAM rather than to disk. This may slightly improve speed under Windows NT	No
NoInitXaxisLimits	Boolean	false	Inverse of default value of the initxlims parameter of the .GRAPH statement. See *Simulator Reference Manual* for details.	
NoKeys	Boolean	off	If on, the default key definitions will be disabled. Note this will not take effect until the *next* session of Pulsonix Spice.	Options dialog
NoMenu	Boolean	off	If on, the default menu definitions will be disabled and no menu bar will appear. As for "NoKeys" this will not take effect until the *next* session of Pulsonix Spice.	Options dialog
NoStopOnError	Boolean	off	If disabled, scripts and multi-command lines (i.e. several commands on the same line separated by ' ;') are aborted if any individual command reports an error.	Options dialog

NoStopOnUnknownParam	Boolean	off	If on, a warning rather than an error is given if an unknown device model parameter is found. The simulation will continue if there are only warnings but will not if there are errors.	Options dialog
OpenIntro	Boolean	off	If set the "Welcome" dialog box is opened on start up.	Check box in "Welcome" dialog
Precision	Numeric	10	Precision of numeric values displayed using Show command.	No
PrintInfo	Boolean	off	Extra diagnostic information about simulation is displayed when this option is on. Limited at present.	No
PrintOptions	Text		Options set in print dialog	Print dialog
ProbeStartDelay	Numeric	1 second	Delay after start of simulation run before fixed probe graphs are first opened.	Options dialog
ProbeUpdatePeriod	Numeric	0.5 seconds	Update period for fixed probe graphs	Options dialog
ScriptDir	Text	none	Directory used to search for scripts and symbol files if not found in the current directory. Changes to this option do not take effect until next session.	Options dialog
StartUpDir	Text	none	Current directory set at start of session.	Options dialog
StartUpFile	Text	startup.sxscr	Script that is automatically run at start of each session.	Options dialog
StatusUpdatePeriod	Numeric	0.2 seconds	Minimum delay in seconds between updates of simulator status window during run.	No
TempDataDir	Text	current working directory	Directory where temporary data files are placed. Although the default is to the current working directory, the first time the program is run it is automatically set to the TEMPDATA directory created by set up program.	Options dialog
TranscriptErrors	Boolean	false	If true, incorrectly typed commands will be entered in the history box. (The drop down list in the command line that shows previously entered commands).	

UpdateCurvesNoDeleteOld	Boolean	false	If true, old curves are not deleted when using the Update Curves feature.	Plot | Update Curve Settings
UpdateCurvesNoFixSelected	Boolean	false	If true update includes selected curves when using the Update Curves feature.	Plot | Update Curve Settings
UseNativeXpSplitters	Boolean	false	The standard ‘splitter’ bar in windows XP is flat and usually not visible. In some simulator windows the standard style has been bypassed in order to make these visible. For example the legend panel in graphs. Set this option to true to revert to standard XP behaviour. You may need to use this if using a non-standard XP theme.	No
UseAltGraphPrintStyles	Boolean	off	Determines method of differentiating curves on monochrome hardcopies. See Graph Printing sheet in options dialog.	Options dialog
UseOldOpenSave	Boolean	off	If specified the old style dialog box is used for File open and close operations. This is faster for some systems.	No
VectorBufferSize	Numeric	256	Each simulation vector is written to a temporary buffer in RAM before being written to a file. This option specifies the size of the buffer in bytes. A larger value will improve the speed of retrieving data at the expense of RAM usage. It is not recommended to make it smaller. You should always enter a value that is a binary power (512, 1024 etc.). Changes do not take effect until Pulsonix Spice is restarted.	No

VertTextMode	Text	Alt	Controls vertical text display when copying graphs to the Windows clipboard. Default setting has been found to be reliable and it isn't usually necessary to change it. If you find a target application does not display the y-axis labels correctly, try values of: Normal (use a different method to rotate text), Hide (hides vertical text) or Horiz (displays vertical text horizontally).	No
WarnSubControls	Boolean	false	Controls vertical text display when copying graphs to the Windows clipboard. Default setting has been found to be reliable and it isn't usually necessary to change it. If you find a target application does not display the y-axis labels correctly, try values of: Normal (use a different method to rotate text), Hide (hides vertical text) or Horiz (displays vertical text horizontally). If true, a warning will be issued if unexpected simulator commands are found in subcircuits.	No

File Extension

The following options set default file extensions. See options dialog for more details.

CatalogExtension	Text	cat	Catalog files	Options dialog
DataExtension	Text	sxdat;dat	Data files	Options dialog
LogicDefExtension	Text	ldf	Arbitrary block logic definition files	Options dialog
ModelExtension	Text	lb;lib;mod;cir	SPICE model files	Options dialog
ScriptExtension	Text	sxscr;txt	Scripts	Options dialog
TextExtension	Text	txt;net;cir;mod;ldf;sxscr;lib;lb;cat	Text files	Options dialog

For Example:

To set curve weight to 2 pixels, type:

```
set CurveWeight=2
```

To set the cursor colour to white, type:

```
set CursorColour=white
```

Note in the above example that the text white need not be enclosed in quotation marks as it does not contain any spaces.

To return curve weight to default value, type:

```
unset CurveWeight
```

Keyboard

The following keys definitions are built in.

Key	Unshifted	Shift	Control	Alt
F1				
F2	Step script	Pause script	Resume script	
F3	More analysis functions			
F4				Quit/Close window
F5	Cursor to next peak	Cursor to previous peak		
F6	Cursor to next trough	Cursor to previous trough		
F7	Ref cursor to next peak	Ref cursor to previous peak		
F8	Ref cursor to next trough	Ref cursor to previous trough		
F9				
F10	New graph sheet			
F11				
F12	Zoom out graph	Zoom in graph		
Insert				
Delete				
Home	Zoom full graph	Zoom full selected axis		
End				
Page Up				
Page Down				
Up	Scroll up graph	Scroll up selected axis		
Down	Scroll down graph	Scroll down selected axis		
Left	Scroll left graph	Scroll left selected axis		
Right	Scroll right graph	Scroll right selected axis		
SPACE				
TAB	Step main cursor	Step reference cursor		
ESC	Abort macro, cancel operation or pause simulation			

Chapter 9. The Simulator

Simulator Control Reference

Overview

Simulator controls instruct the simulator how to read in and simulate the circuit. All simulator controls begin with a period (.) .

The simulator receives its input from the netlist which contains the circuit configuration along with simulator controls.

The schematic editor supports some of the controls described in this chapter but not all. For example pole-zero analysis (.PZ) is not available. A pole-zero analysis can, however, be defined by adding the .PZ control to the schematic's netlist. See section *Adding Extra Netlist Lines* in the Circuit Definition chapter.

.MODEL and .SUBCKT controls may also appear in model library files (in fact that is where they would usually reside) see the next chapter *Device Library and Model Management*. The .alias control may only appear in model library files.

The following simulator controls are recognised by Pulsonix Spice.

.ac

.dc

.endf

.ends

.file

.func

.global

.graph

.ic

.inc

.keep

.lib

.model

.nodeset

.noise

.op

.options

.param

.print

.p2

.sens

.subckt

.temp

.tf

.trace

.tran

The following control is only recognised in model library files.

.alias

General Sweep Specification

Overview

Pulsonix Spice features a common sweeping algorithm which is used to define the swept analysis modes: .DC, .AC, .NOISE and (now) .TF, along with multiple analyses such as Monte Carlo.

The sweep algorithm has 6 modes:

- Device. Sweeps a single default value of a specified device. E.G. voltage of a voltage source, resistance of a resistor or the capacitance of a capacitor.
- Temperature
- Parameter. Parameter can be referenced in an expression for a model or instance parameter.
- Model parameter. Named model parameter.
- Frequency. (Not applicable to .DC)
- Monte Carlo. Perform a specified number of steps with distribution functions (i.e tolerances) enabled.

Standard SPICE only provides a subset of the above. .DC can only sweep voltage and current sources, .AC and .NOISE can only sweep frequency while .TF can't be swept at all.

As well as providing 6 modes, each of the modes can sweep in four different ways. These are linear, decade, octave and list.

Syntax

All the swept analysis modes use the same general syntax to specify the sweep parameters. However, to maintain compatibility with SPICE and its derivatives including earlier versions of Pulsonix Spice, each analysis mode allows variations to this standard syntax. The general syntax is described below while the variations allowed for each analysis mode are described in the section dedicated to that analysis mode.

All of the analysis modes can optionally be entered in a similar manner to .MODEL statements i.e. as an unordered list of parameter names followed by their values. For example, the following is a perfectly legal noise analysis specification:

```
.noise V-vout DEVICE-V1 VN=0 F=1k LIN= (100 -10m 10m)
+ INSRC=V1
```

In the various forms of the syntax described in the following sections, some of the parameter names may be omitted as long as they are entered in a particular order. It is sometimes, however, easier to remember parameter names rather than a default order, so the method described above may be more convenient for some users.

General syntax for swept analyses

.AC|.DC|.NOISE|.TF *sweep_spec* [analysis specific parameters]

sweep_spec:
One of the following:

DEVICE *device_name step_spec* F *frequency*

TEMP *step_spec* F *frequency*

PARAM *param_name step_spec* F *frequency*

MODEL *model* [PARAM] *mod_param_name step_spec* F *frequency*

FREQ *step_spec*

MONTE *num_step* F *frequency*

Where

device_name	Name of device to be swept. The following components may be swept: Capacitors, all controlled sources, fixed current source, fixed voltage source, inductors and resistors.
Param_name	Name of parameter used in expression. Expressions may be used to define an instance or model parameter and may also be used in arbitrary sources.
Model	Name of model containing parameter to be swept
Mod_param_name	Name of model parameter
Num_steps	Number of steps to be performed for Monte Carlo sweep.
Frequency	Specified frequency for which .NOISE, .AC and .TF analyses are to be performed. May be zero for .AC and .TF

Step_spec:
One of the following:

STP *start stop step*

LIN *num_points start stop*

DEC *num_points_decade start stop*

OCT *num_points_octave start stop*

LIST *val1* [*val2* …]

Where:

start	First value
stop	Last value (inclusive)
step	Interval
num_points	Total number of points
num_points_decade	Number of points per decade
num_points_octave	Number of points per octave

STP and LIN modes are both linear sweeps but specified differently. STP specifies start, stop and a step size, while LIN specifies start, stop and the total number of points.

Multi Step Analysis

Overview

The sweep specification described in “General Sweep Specification” can also be applied to define multiple analysis including Monte Carlo analysis. This can be applied to the swept modes .DC, .AC, .NOISE and .TF along with .TRAN. The analyses .SENS, .PZ and .OP cannot be run in multi-step mode. A multi-step .OP is in fact the same as .DC so this is not required. Monte Carlo analysis is the subject of its own section but it is invoked in the same way as other multi-step modes. As well as the standard 6 sweep modes, small-signal multi-step analyses can be run in snapshot mode which uses snapshots created by a previous transient analysis.

Syntax

The general form is:

.analysis_name analysis parameters SWEEP

+ [*sweep_spec*]

Where:

.analysis_name	Dot control for analysis
analysis_parameters	Specific parameters for that analysis
sweep_spec	See “General Sweep Specification”

Examples

Run 10 Monte Carlo runs for 1mS transient analysis

```
    .TRAN !m SWEEP MONTE 10
```

Sweep V1 from 0 to 5 volts in 0.1V steps for 200us transient

```
    .TRAN 200u SWEEP DEVICE V1 STP 0 5 0.1
```

AC sweep of voltage source V5 from -300mV to 300mV. Repeat 6 times for parameter restail from 450 to 550.

```
    .AC DEVICE=V5 LIN 100 -300m 300m F=100000
    + SWEEP PARAM=restail LIN 6 450 550
```

.AC

.AC *inner_sweep_spec* [F *frequency*] [SWEEP *outer_sweep_spec*]

Spice compatible frequency sweep:
.AC DEC|LIN|OCT *num_points start stop*

Instructs the simulator to perform a swept small signal AC analysis. Pulsonix Spice AC analysis is not limited to a frequency sweep as it is with generic SPICE and derivatives. See “General Sweep Specification” and examples below for more details.

frequency	Frequency at which analysis will be performed for non-frequency sweeps. Default 0.
inner_sweep_spec	See “General Sweep Specification” for syntax. Defines sweep mode. FREQ keyword is optional.
outer_sweep_spec	If specified, analysis will be repeated according to this specification. See “General Sweep Specification” for syntax
num_points	LIN: total number of points DEC: number of points per decade OCT: number of points per octave
start	Start frequency for SPICE compatible mode
stop	Stop frequency for SPICE compatible mode

Except for frequency sweep, the frequency at which the analysis is being performed should be specified. If omitted, the frequency will be assumed to be zero.

For non-frequency sweeps, a new dc operating point may be calculated at each step depending on what is being swept. If a capacitor, inductor or an ‘AC only’ model parameter is being swept, then no new dc operating point will be required. Otherwise one will be performed. An ‘AC only’ parameter is one that does not affect DC operating point such as device capacitance.

Notes

An AC analysis calculates the small signal frequency response of the circuit about the dc operating point. The latter is automatically calculated prior to commencing the frequency sweep. One or more inputs may be specified for AC analysis by using voltage or current sources with the AC specification (See “Voltage Source”) The results of an AC analysis are always complex.

Examples

SPICE compatible. Sweep frequency from 1kHz to 1Meg

```
.AC DEC 25 1k 1MEG
```

Sweep voltage source V1 100 points from -100mV to 100mV. Frequency = 100kHz

```
.AC DEVICE V1 LIN 100 -100m 100m F=100k
```

Sweep parameter Rscale from 0.5 to 3 in steps of 0.1. Frequency=20Meg

```
.AC PARAM Rscale STP 0.5 3 0.1 F=20Meg
```

Sweep resistor R1 with values 10k 12k 15k 18k 22k 27k 33k, Frequency=1.1KHz

```
.AC DEVICE R1 LIST 10k 12k 15k 18k 22k 27k 33k F=1.1k
```

Monte Carlo sweep 100 steps. Frequency – 10K.
This is useful if – say- you are interested in the gain of an amplifier at one frequency and it needs to lie within a defined tolerance. Previously you would need to repeat an AC sweep at a single frequency to achieve this which could take a long time especially if the circuit has a difficult to find operating point. The analysis defined by the following line will take very little time even for a large circuit.

```
.AC MONTE 100 F=10K
```

Examples of Nested Sweeps

As Monte Carlo above but repeated from 0 to 100C

```
.AC MONTE 100 F=10K SWEEP TEMP STP 0 100 10
```

…and at a number of frequencies

```
.AC MONTE 100 SWEEP FREQ DEC 5 1k 100k
```

.alias

.alias *alias_name device_name device_type*

This control may only be used in device model library files. It is not recognised by the simulator. It permits a device model or subcircuit to be referenced by a different name. This allows one model definition to be used for multiple part numbers.

alias_name	Alias name
device_name	Device to which alias refers
device_type	Type of device to which alias refers. Must be one of the following C, D, LTRA, NJF, NMF, NMOS, NPN, PJF, PMF, PMOS, PNP, R, SW or SUBCKT. see .model for more details.

Example:

```
.model BC547C npn ( IS=7.59E-15 VAF=19.3 BF=500 IKF=0.0710 NE=1.3808
+    ISE=7.477E-15 IKR=0.03 ISC=2.00E-13 NC=1.2 NR=1 BR=5 RC=0.75
+    CJC=6.33E-12 FC=0.5 MJC=0.33 VJC=0.65 CJE=1.25E-11 MJE=0.55
+    VJE=0.65 TF=4.12E-10 ITF=0.4 VTF=3 XTF=12.5 RB=172 IRB=0.000034
+    RBM=65
.alias BC549C BC547C NPN
```

The above would provide identical definitions for both BC547C and BC549C bipolar transistors.

Notes: *.alias definitions will recognise models defined in other files provided the file in which the alias resides and the file in which the model definition resides are part of the same library specification. A library specification is a single pathname possibly with a wildcard ('?' or '*') to refer to multiple files. E.g. \Pulsonix Spice\models\*.mod is a library specification and refers to all files with the extension '.mod' in the directory \Pulsonix Spice\models.*

Aliases must refer directly to a model or subcircuit definition and not to other aliases.

.DC

.DC *inner_sweep-spec* [SWEEP *outer_sweep_spec*]

Spice compatible:
.DC *device_name start stop step*

The remainder are Pulsonix Spice Version 1 compatible:
.DC TEMP *start stop step*
.DC PARAM *param_name start stop step*
.DC MODEL *model* [PARAM] *mod_param_name start stop step*

Instructs simulator to perform a DC sweep analysis. A dc analysis performs a dc operating point analysis for a range of values according to the sweep specification. Pulsonix Spice DC analysis is not limited to sweeping a voltage or current source as with generic SPICE. Any mode defined by the general sweep specification may be used although frequency sweep has no useful purpose.

inner_sweep_spec	See “General Sweep Specification” for syntax. Defines sweep mode. DEVICE keyword is optional.
outer_sweep_spec	If specified, analysis will be repeated according to this specification. See “Multi Step Analyses” for details
device_name	Component reference of voltage source, current source, resistor or controlled source to be swept. (Only voltage and current sources are SPICE compatible)
start	Start value for sweep
stop	Stop value for sweep
step	Increment at each point

param_name	Parameter name. This would be used in an expression to define a component or model value.
model_name	Model name e.g. Q2N2222
model_parameter_name	Model parameter name e.g. IS

If *start* is arithmetically greater than *stop* then *step* must be negative.

It is not necessary to declare parameters with .PARAM if using parameter sweep.

Examples

SPICE compatible. Sweep V1 from 0 to 5 volts in steps of 0.1 volt

```
.DC V1 0 5 0.1
```

Pulsonix Spice Version 1 compatible temperature sweep

```
.DC TEMP 0 100 2
```

Decade (i.e. logarithmic) sweep. Sweep V1 from 1mV to 1V with 25 points per decade

```
.DC V1 DEC 25 1m 1v
```

Note that the DEVICE keyword has been omitted. This is the default sweep mode for .DC.

Do 1000 Monte Carlo steps. This performs the same task as a Monte Carlo analysis applied to a DC operating point. In other products and earlier versions of Pulsonix Spice this task would take a long time as the operating point is solved from scratch each time. With the mode described by the following example, the operating point need only be calculated from scratch once. all subsequent steps are seeded by the previous one and usually require only a few iterations. The end result is a sometimes spectacular increase in speed.

```
.DC MONTE 1000
```

Examples of Nested Sweeps

Decade sweep at temperatures 0 to 100 in steps of 10

```
.DC V1 DEC 25 1m 1v SWEEP TEMP STP 0 100 10
```

Note the STP keyword is necessary to signify the start-stop-step method of defining a linear sweep. Alternatively LIN can be used which defines the sweep in terms of the total number of points. The following is equivalent to the above:

```
.DC V1 DEC 25 1m 1v SWEEP TEMP LIN 11 0 100
```

Do 100 run Monte Carlo analysis for temperature sweep

```
.DC TEMP 0 100 2 SWEEP MONTE 100
```

.ends

```
.ends
```

Terminates a subcircuit definition. See .subckt for more details.

.file and .endf - Embedding files in netlist

.file *filename*

file_contents

.endf

The .file control allows the contents of a file referenced in a .MODEL control to be placed directly in the netlist. Files are referenced in arbitrary logic blocks, PWLFILE voltage sources digital sources and digital state machines. Each of these may refer to files defined using .file and .endf.

Example

```
.model COUNT_8 d_logic_block file=counter_def


.file counter_def
PORT (DELAY = 10n)CountOut out[0:7] ;


EDGE (DELAY=5n, WIDTH=8, CLOCK=in[0])   Count ;


Count = Count + 1 ;


CountOut = count ;
.endf
```

The .MODEL control refers to a file called "counter_def". This could be a real disk file called counter_def or counter_def.ldf, but in the above example it is instead defined directly in the netlist using .file and .endf

Important Note

.file and .endf will not be recognised in library files.

.func

.FUNC *name (arglist) {body}*

name	Name of function. Must begin with a letter and not match one of the built in functions.
arglist	List of comma separated argument names
body	Body of function. This is an expression referring to the names in arglist that defines the operation performed by the function

.FUNC defines a function that can be used in a model or device parameter expression, a parameter defined using .PARAM or in an arbitrary source expression.

Examples

```
.FUNC FREQ(V) { (V)*120K }

.FUNC SWEEP(V) { SIN(TIME*FREQ(v)*2*PI) }
```

Optimiser

Any expression that uses a function defined with .FUNC will be automatically processed by an optimisation algorithm. For more information see “Optimisation” section.

The optimiser attempts to speed simulations by making the expression evaluation more efficient. The optimiser is effective when .FUNC is used to create very complex expressions perhaps to develop a semiconductor device. In simple applications it may not make a noticeable improvement to performance. The optimiser can be enabled for all expressions and can also be disabled completely. To enable for all expressions use:

```
.OPTIONS optimise=2
```

To disable:

```
.OPTIONS optimise=0
```

.global

.GLOBAL *node* [*node…*]

Identifies nodes as global. This allows nodes specified at the top level of a circuit to be accessed within a subcircuit definition. For more information see “Subcircuits” section.

.graph

```
.graph signal_name|"expression"
    [ persistence = persistence ]
    [ axisname = axisname ]
    [ graphname = graphname ]
    [ axistype = digital|grid|axis|auto ]
    [ curvelabel = curvelabel ]
    [ xlabel = xlabel ]
    [ ylabel = ylabel ]
    [ xunit = xunit ]
    [ yunit = yunit ]
    [ xmin = xmin ]
    [ ymin = ymin ]
    [ xmax = xmax ]
    [ ymax = ymax ]
    [ analysis = analyses_list ]
    [ ylog = lin|log|auto ]
    [ xlog = lin|log|auto ]
    [ nowarn = true|false ]
    [ initXLims = true|false]
    [ complete = true|false ]
```

.GRAPH instructs Pulsonix Spice to plot a graph of the specified signal or expression. The graph can be plotted incrementally as the simulation proceeds or may be delayed until the run is complete.

Parameter name	Type	Description
signal_name | expression	string	Specifies item to be plotted. If this is an expression, then it must be enclosed in double quotation marks or curly braces.
persistence	integer	Number of curves to be displayed at once. On repeated runs, any curves from earlier runs remain until the number of curves exceeds this value at which point the oldest is deleted automatically. If this parameter is absent or zero, the curves are never deleted.
graphname	string	If specified, the curves will be directed to their own graph sheet within the current window. The value of *graphname* is arbitrary and is used to identify the graph so that multiple .graph controls can specify the same one. It works in a similar way to *axisname* an example of which is given below. This name is not used as a label for display purposes but simply as a means of identification.
axistype	string	Can be one of four values to specify type of y-axis: • DIGITAL. Use a digital axis. This is a small axis that carries only one curve. It is intended for digital signals but may also carry analog curves. • GRID. Use a separate grid stacked on top of the main one. The AXISNAME parameter may be used to identify a particular grid used by another .GRAPH control. • AXIS. Use a separate y-axis alongside the main one. The AXISNAME parameter may be used to identify a particular axis used by another .GRAPH control. • AUTO. This is the default value, a suitable axis is chosen automatically.
axisname	string	This is only used if AXISTYPE is specified. The value of AXISNAME is arbitrary and is used to identify the axis so that multiple .graph controls can specify the same one. An example of this is given below. This name is not used as a label for display purposes but simply as a means of identification. Axes can be labelled using ylabel and xlabel.
curvelabel	string	Label for curve displayed in graph legend. If omitted, the label will be the signal name or expression. You can use component value SpiceProbeCurveLabel to add these labels in the schematic.
xlabel	string	Label for x-axis. Default is reference of curve being plotted (E.g. time, frequency etc.)
ylabel	string	Label for y-axis. If there is only a single curve, this will default to the label for the curve otherwise the default is blank.
xunit	string	Units for x-axis. Default is units of reference.
yunit	string	Units for y-axis. Default is units of curves plotted provided they are all the same. If any conflict, the default will be blank

xmin	real	Minimum limit for x-axis. Must be used with xmax.
xmax	real	Maximum limit for x-axis. Must be used with xmin.
ymin	real	Minimum limit for y-axis. Must be used with ymax.
ymax	real	Maximum limit for y-axis. Must be used with ymin.
analysis	string	Specifies for what analysis modes the plot should be enabled. By default it will be enabled for all analysis modes. Any combination of the following strings, separated by a pipe ('|') symbol. • TRAN. Transient analysis • AC. AC analysis • DC. DC sweep analysis • NOISE. Noise analysis Other analysis modes do not produce results that can be plotted.
ylog	string	One of three values LIN Use linear axis LOG Use log axis AUTO Axis will be log if x values are log spaced. (E.g. for decade AC sweep) and all values are positive. Default if omitted: LIN
xlog	string	One of three values LIN Use linear axis LOG Use log axis AUTO Axis will be log if x values are log spaced. (E.g. for decade AC sweep) and all values are positive. Default if omitted: AUTO
nowarn	boolean	If true, no warnings are given if an attempt is made to plot a non-existent signal. Default: false.
initXLims	boolean	When this is TRUE (the default), the x-axis limits are initialised according to the analysis. E.g. if the analysis is transient and runs from 0 to 1mS, the x-axis will start with these limits. If set to FALSE, the x-axis limits are calculated to fit the curve and updated incrementally. You should set this to FALSE if you are plotting an expression whose x values are not the same as the x values for the analysis e.g. using the XY() function for an X-Y plot.
complete	boolean	If true, the plot is not produced until the analysis is completed. Otherwise the plot is updated at a rate determined by the global option ProbeUpdatePeriod. This can be set using the options dialog box (File|Options|General...). Default: false.

The .GRAPH control is the underlying simulator mechanism used by the schematic's *fixed probes*.

.GRAPH supersedes the older and less flexible .TRACE. The latter is, however, still supported and may sometimes be convenient for specifying multiple signals on one line.

Using multiple .GRAPH controls

If specifying several .GRAPH controls to plot a number of curves on the same graph, you should make sure that the various parameters are consistent. If for example, a conflict arises if you specify xmin and xmax for two .GRAPH's that plot curves in the same graph sheet, and the values for xmin and xmax are different for each. You can specify xmin and ymin for just one of the .GRAPH controls or you can specify for all and make sure they are all the same. The same applies to other non-boolean parameters i.e. ymin, ymax, xlabel, ylabel, xunits and yunits. The parameter initXLims, however must be specified with the same value for all .GRAPH controls specifying the same graph sheet.

Conflicting values of ylog and xlog are resolved by plotting the curves on separate axes or graph sheets respectively.

Creating X-Y plots

To create an X-Y plot, use the XY() function. You should also specify "initXLims=false". E.g.

```
.GRAPH "XY( imag(vout), real(vout) )" initXLims=false xlog=LIN complete=true
```

The above will create a nyquist plot of the vector "VOUT".

Using .GRAPH in subcircuits

.GRAPH maybe used in a subcircuit in which case a plot will be produced for all instances of that subcircuit. Note, however, that it will only work for single values and not for expressions when inside a subcircuit. The value of the **curveLabel** parameter will be prefixed with the instance name so that the displayed curves can be correctly identified.

Examples

```
.graph C2_P curveLabel="Amplifier output" nowarn=true
```

Plots the vector C2_P and gives it the label "Amplifier output". As NOWARN is TRUE, no warning will be given if C2_P does not exist.

```
.graph vout_quad
+ axisType="grid"
+ axisName="grid1"
+ persistence=2
+ curveLabel="Quadrature"
+ nowarn=true
+ analysis = TRAN|DC

.graph vout
+ axisType="grid"
+ axisName="grid1"
+ persistence=2
+ curveLabel="In Phase"
+ yLabel="Filter Outputs"
+ nowarn=true

+ analysis = TRAN|DC
```

The above illustrates the use of the parameters AXISTYPE and AXISNAME. Both the vectors specified by the above .GRAPH controls will be plotted on the same but separate grid. Because both grids have been given the AXISNAME "grid1", each curve will be plotted on the same one. If the values of axisname for the above were different, each curve would be plotted on a *different* grid. The ANALYSIS parameter has been specified in both cases, so plots will only be created for transient and dc sweep analyses.

Using Expressions in .graph

You can enter an expression as well as single vectors to be plotted. A problem arises when plotting expressions incrementally that are regularly updated while the simulation is running. Pulsonix Spice versions prior to version 2 could not incrementally evaluate expressions, so each time the plot of an expression was updated, the expression had to be recalculated from the beginning. This was inefficient and it has always been recommended that the **complete=true** flag was added in these circumstance to inhibit incremental plotting.

Pulsonix Spice now has the ability to incrementally evaluate some expressions and there is no longer a recommendation to set **complete=true**. However, certain expression cannot be incrementally evaluated and when such expressions are entered, incremental plotting will automatically be disabled and the plot won't appear until the run is complete.

A notable example of expressions that cannot be incrementally evaluated is anything containing the phase() function. This is because the phase() function uses a state machine to determine when the phase wraps from -180 to 180 and back. An offset is then applied to make the phase plot continuous. Because of the state machine, it is always necessary to evaluate this function from start to finish which makes incremental evaluation difficult. An alternative is to use instead the arg() function. This is the same as phase, but does not have the state machine and always gives an output that lies between +/- 180 degrees.

Plotting Spectra with .GRAPH

You can use .GRAPH to create spectrum plots using FFTs or Fourier. However, FFT is quite difficult to use on its own as it needs interpolated data. So, a new user defined function called Spectrum() has been developed that is especially designed for use with

.GRAPH. Usage is:

Spectrum (vector, numPoints [, start [, stop]])

Where:

vector	Vector or expression
numPoints	FFT size - must be a binary power of 2
start	Start time - default = 0
stop	Stop time - default = end of data

Spectrum() cannot be incrementally evaluated and so incremental plotting will automatically be disabled for any .GRAPH control that uses it.

.ic - Initial Conditions

.ic V(*node1*)=*val1* [V(*node2*)=*val2*]...

OR

.ic *node1 val1* [*node2 val2*]

This control sets transient analysis initial conditions.

node1, node2 etc.	Name of circuit node (or net) to which initial condition is to be applied. See notes below.
val1, val2 etc.	Voltage to be applied to net as initial condition.

If the UIC parameter is specified with the .tran control no DC operating point will be calculated so an initial condition will set the bias point in the same way as an IC=... parameter on a BJT, capacitor, diode, JFET or MOSFET.

If the UIC parameter is absent from the .tran control then a DC operating point is calculated before the transient analysis. In this case the net voltages specified on the .ic control are forced to the desired initial values during the DC operating point solution. Once transient analysis begins this constraint is released. By default the voltage force is effectively carried out via a 1Ω resistor. This can be changed with the option setting ICRES. (See .OPTIONS section).

To add an initial condition to a schematic there is a special component called “Initial Condition”.

Alternative Initial Condition Implementations

An initial condition can also be specified using a voltage source with the DCOP parameter specified. E.g.

```
VIC1 2 3 3.5 DCOP
```

Will force a voltage of 3.5 volts between nodes 2 and 3 during the DC operating point solution. This has 2 advantages over .IC:

1. It has zero force resistance
2. It can be applied differentially

You can also use a capacitor with the BRANCH parameter set to 1. E.g.:

```
C1 2 3 10u BRANCH=1 IC=3.5
```

This will behave identically to the voltage source in the above example during the DC operating point but during a subsequent small-signal or transient analysis will present a 10μF capacitance to nodes 2 and 3.

.inc

.inc *pathname*

Insert the contents of the specified file.

pathname File system pathname for file to be included

The .inc control is replaced by the specified file in its entirety as if was part of the original file. .inc controls may also be nested i.e. there may be .inc controls within the included file. Nesting may be to any level.

.keep

```
.KEEP signal_spec [signal_spec….]
```

This control tells the simulator what values to store during a simulation. By default, all top level voltages and currents and digital data are stored.

signal_spec	/**TOP** | /**SUBS** | /**NOV** | /**NOI** | /**NODIG** | ***V** | ***I** | ***D** | ****V** |****I** | ****D** | *subref*.***V** | *subref*.***I** | *subref*.***D** | *subref*.****V** |
Subref	.****I** | *subref*.****D** | ^*wildcard_filter* | *signal_name*
subref	Sub-circuit reference
/**NOV**	Don't store top level (i.e. not in a subcircuit) voltages.
/**NOI**	Don't store top level currents
/**NODIG**	Don't store top level digital data.
/**SUBS**	Store all subcircuit data.
/**TOP**	Overrides /subs. This is to inhibit storing signals in child schematics in hierarchical designs.
^wildcard_filter	General specification that selects values to store based on their name alone. Would usually use one of the special characters '*' and '?'. '*' means 'match one or more characters' while '?' means 'match a single character'. Some examples:
*	matches anything
X1.*	matches any signal name that starts with the three letters: X1.
X?.*	matches any name that starts with an X and with a '.' for the third letter.
*.q10#c	matches any name ending with q10#c.
***V**	Store all top level voltages. This is actually implicit and need not be specified at the top level of the netlist. It can be usefully used in sub-circuit definitions - see notes.
***I**	Store all top level currents. This is actually implicit and need not be specified at the top level of the netlist. It can be usefully used in sub-circuit definitions - see notes.
***D**	Store all top level digital data. This is actually implicit and need not be specified at the top level of the netlist. It can be usefully used in sub-circuit definitions - see notes.
****V**	Store all voltages including those inside sub-circuits descending to all levels
****I**	Store all currents including those inside sub-circuits descending to all levels

****D**	Store all digital data including those inside sub-circuits descending to all levels
subref.***V**	Store all voltages within sub-circuit *subref* excluding voltages within children of *subref*.
subref.***I**	Store all currents within sub-circuit *subref* excluding currents within children of *subref*.
subref.***D**	Store all digital data within sub-circuit *subref* excluding digital data within children of *subref*.
subref.****V**	Store all voltages within sub-circuit *subref* including voltages within children of *subref* descending to all levels.
subref.***I**	Store all currents within sub-circuit *subref* including currents within children of *subref* descending to all levels.
*subref***.*D**	Store all digital data within sub-circuit *subref* including digital data within children of *subref* descending to all levels.
signal_name	Explicit voltage or current.

This control instructs the simulator what values to store during a simulation. By default, all voltages and currents not within subcircuits are stored.

/**SUBS**	Store all available voltages and currents including subcircuit data. (Overrides /NOV and /NOI).
/**NOV**	Don't store currents in voltage sources and inductors or any voltages.
/**NOI**	Don't store currents. (Except those in voltage sources and inductors)
/**TOP**	If present, overrides /**SUBS**.
signalname.	Name of explicit signal (e.g. voltage or current) to store.

Notes

.KEEP may be used inside a sub-circuit definition in which case .KEEP operates at a local level. For example .KEEP *v inside a sub-circuit definition specifies that all voltages within that subcircuit (for all instances) will be saved. .KEEP **v does the same but also includes any descendant sub-circuit instances. In earlier versions, *voltage* also meant currents in voltage sources and inductors as

these values are obtained as part of the matrix solution. This is no longer the case, *voltage* and *current* mean what they say.

Examples

Store only voltages and currents in sub-circuit X1 excluding descendants.

```
      .KEEP /noi /nov /top X1.*v X1.*i
```

Store only voltages and currents in sub-circuit X1 including descendants.

```
      .KEEP /noi /nov /top X1.**v X1.**i
```

Store voltages within U3.U12 along with VOUT and VIN

```
      .KEEP /noi /nov /top U3.U12.*v VOUT VIN
```

Store all top level voltages and currents in U7

```
.KEEP /noi /top U7.*i
```

.lib - Referencing libraries

.lib *pathname*

This control specifies a pathname to be searched for model and subcircuit libraries. Any number of .lib controls may be specified and wildcards (i.e. * and ?) may be used.

If a model or subcircuit is called up by a device line but that definition was not present in the netlist, Pulsonix Spice will search for it in the files specified using the .LIB control.

Pulsonix Spice will also search for definitions for unresolved parameters specified in expressions. These are defined using .PARAM

Example:

The following control instructs the simulator to search all files with the .mod extension in c:\simlib for any required subcircuits or device models.

```
.lib c:\simlib\*.mod
```

.model - Defining model parameters

.model *modelname modeltype* (*param1=val1* [*param2=val2*]...)

This control specifies a set of model parameters that are used by one or more devices. .model controls usually reside in model libraries.

modelname	Model name. Any text string to uniquely identify model. Must begin with a letter but may contain any legal ASCII character other than a space and period '.' .
modeltype	Model type. Must be one of the following:

XSPICE models

ad_converter	analog-to-digital converter
adc_bridge	analog-to-digital interface bridge
adc_schmitt	analog-to-digital schmitt trigger
cm_cap	Capacitor with voltage initial condition
cm_ind	Inductor with current initial condition
d_and	digital n-input and gate
d_buffer	digital one-bit-wide buffer
d_cap	Digital capacitor
d_dff	digital d-type flip flop
d_dlatch	digital d-type latch
d_fdiv	digital frequency divider
d_init	Digital initial condition

d_inverter	digital one-bit-wide inverter
d_jkff	digital jk-type flip flop
d_logic_block	arbitrary logic block
d_nand	digital n-input nand gate
d_nor	digital n-input nor gate
d_open_c	digital one-bit-wide open-collector buffer
d_open_e	digital one-bit-wide open-emitter buffer
d_or	digital n-input or gate
d_osc	controlled digital oscillator
d_pulldown	digital pulldown resistor
d_pullup	digital pullup resistor
d_pulse	digital pulse
d_ram	digital random-access memory
d_res	Digital resistor
d_source	digital signal source
d_srff	digital set-reset flip flop
d_srlatch	digital sr-type latch
d_state	digital state machine
d_tff	digital toggle flip flop
d_tristate	digital one-bit-wide tristate buffer
d_xnor	digital n-input xnor gate
d_xor	digital n-input xor gate
da_converter	digital-to-analog converter
dac_bridge	digital-to-analog interface bridge
s_xfer	s-domain transfer function block

SPICE Models

C	Capacitor
CORE	Jiles-Atherton core model
CORENH	Simple core model
D	Diode
LTRA	Lossy transmission line
NIGBT	IGBT
NJF	N-channel JFET
NMF	N-channel GaAsFET
NMOS	N-channel MOSFET
NPN	NPN BJT (Bipolar junction transistor)

PJF	P-channel JFET
PMF	P-channel GaAsFET
PMOS	P-channel MOSFET
PNP	PNP BJT
R	Resistor
SW	Voltage controlled switch

param1, param2 etc.	Parameter name. Valid values depend on the model type. (See "Device Reference" section)
val1, val2 etc.	Parameter value.

Example

The following is a model for a 1N5404 diode.

```
.model D1n5404  D(Is=15.48f Rs=7.932m Ikf=0 N=1 Xti=3 Eg=1.11 Cjo=150p
+               M=.3 Vj=.75 Fc=.5 Isr=120n Nr=2 Bv=525 Ibv=100u)
```

It is possible to define SOA limits within the .MODEL statement. To do this, add one or more parameters in the following format:

LIMIT(*name*)=(*min, max, xwindow*)

name	Name of quantity to test. See format for access variables useable when MODEL is specified for a .SETSOA control. This is described in section: ".SETSOA" in the section below. E.g. use 'LIMIT(vcb)' to specify the limits for the collector-base voltage of a BJT.
min, max	As described in ".SETSOA" below.
xwindow	As described in ".SETSOA" below.

.nodeset - Convergence assist

.nodeset v(*node1*)=*val1* [v(*node2*)=*val2*]...

OR

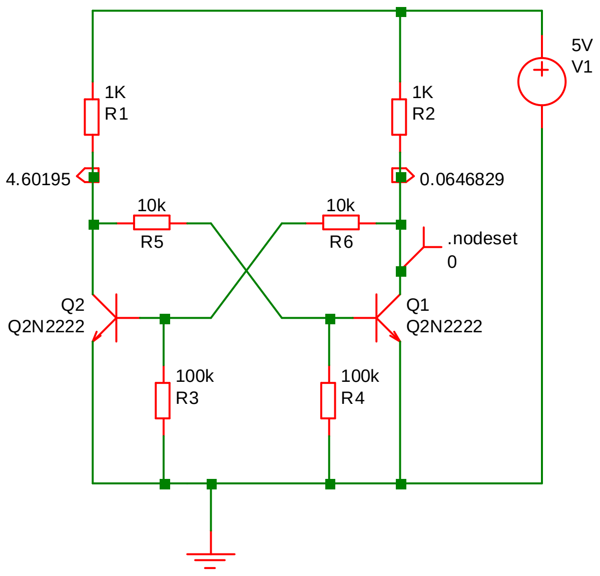
.nodeset *node1 val1* [*node2 val2*]

This control sets an initial guess voltage at the specified node for the dc operating point solution.

node1, node2 etc.	Name of circuit node (or net) to which nodeset is to be applied. See notes below.
val1, val2 etc.	Nodeset voltage to be applied.

Initially nodesets work exactly the same way as initial conditions. The nodeset voltage is applied via a 1 Ohm (by default) resistor and the solution is completed to convergence (by any of the methods). The nodeset is then released and the solution repeated. If the nodeset voltage is close to the actual solution the convergence of the second solution should be rapid.

Nodesets can be used to force a particular solution for circuits that have more than one stable state. Consider the following circuit:



A nodeset has been applied to the collector of Q1. This has forced Q1 to be on and Q2 to be off. If the nodeset were absent the solution would actually leave both Q1 and Q2 partially on. In real life this would not be stable but is numerically accurate.

The other application of nodesets is to help convergence for the DC bias point. With Pulsonix Spice, it is rarely necessary to use nodeset's to find the DC solution of a circuit. They can, however, be useful for speeding up the operating point analysis for circuits that have already been solved. Pulsonix Spice features a method of creating nodesets for this purpose using the SaveRhs command.

Nodeset's should not be confused with initial conditions. (see *.ic* section). Initial conditions tie a node to a particular voltage and keep it there throughout the DC operating point analysis. Nodeset's merely "suggest" a possible solution but do not force it.

To add a nodeset to a schematic there is a special component called “Nodeset”.

.NOISE

.NOISE *inner_sweep-spec* [V] *pos_node* [VN] *neg_node*
+ [[INSRC] *in_source*]
+ [F *frequency*] [SWEEP *outer_sweep_spec*]

Spice Compatible

.NOISE V(*pos_node* [, *neg)_out_node*]) *in_source*
+ DEC|LIN|OCT *num_points start stop interval*

This control instructs the simulator to perform a small signal noise analysis.

pos_node	Node on circuit at which noise is measured.
neg_node	Node to which *outputnode* is referenced. Defaults to ground if omitted.
in_source	Input source (i.e. voltage or current) to which the input noise measurement is referred.
inner_sweep_spec	See “General Sweep Specification” for syntax. Defines sweep mode. FREQ keyword is optional.
outer_sweep_spec	If specified, analysis will be repeated according to this specification. See “General Sweep Specification” for syntax.
LIN	Analysis points are linerly spaced.
DEC	Analysis points are logarithmically spaced in decades
OCT	Analysis points are logarithmically spaced in octaves
num_points	LIN: Total number of points DEC: Number of points per decade OCT: Number of points per octave
start	Start frequency
stop	Stop frequency
interval	Currently does nothing. Provided for backward compatibility.

Notes

During noise analysis the simulator calculates the total noise measured between *pos_node* and *neg_node* at each frequencly point. It also calculates and outputs this noise referred back to an input specified by *in_source.* As for all other analysis modes a DC operating point analysis is carried out first but, unlike AC analysis, the results of this analysis are not made available. The simulator outputs vectors covering the contribution from each noise generating device to the total output noise. The names of these vectors begin with the component reference of the device followed by a suffix to indicate the source of the noise within the device. A listing of the suffixes is given below. It is important to note that it is not the noise being generated by each device that is output but the proportion of that noise that is propagated to the output.

It is not necessary to specify a separate AC analysis alongside the noise analysis as it is with SPICE2 and commercial derivatives of SPICE2

The magnitude of any AC independent voltage or current source on the circuit has no effect on the results of a noise analysis. Unlike SPICE and earlier versions of Pulsonix

Spice it is not necessary to specify an AC parameter for the source used for the noise input source. For the first form shown above, the input source is in fact optional. If it is omitted the input referred noise will not be calculated.

All noise results are in V/√Hz except input noise referred back to a current source which is in A/√Hz. In standard SPICE3 the noise values produced for MOS2 and BSIM3 devices are in V^2/√Hz. For consistency, these have now been changed to V/√Hz. The original SPICE3 behaviour can be restored by setting the simulator option OldMosNoise (see “OPTIONS”)

Device vector name suffixes

Device Type	Suffix and Description
BJT	#rc Noise due to collector resistance #rb Noise due to base resistance #re Noise due to emitter resistance #ic Shot noise in collector #ib Shot noise in base #1overf Flicker (1/f) noise no suffix Total transistor noise
Diode	#rs Noise due to series resistance #id Shot noise #1overf Flicker (1/f) noise no suffix Total diode noise
JFET and MOSFETs level 1-3 and BSIM3	#rd Noise due to drain resistance #rs Noise due to source resistance #id Shot noise in drain #1overf Flicker (1/f) noise no suffix Total FET noise
Philips MOS9 (all types see “Philips Compact Models”	#Sfl Flicker (1/f) noise #Sth Drain thermal noise #Sig Gate thermal noise #Sigth Gate-drain correlated thermal noise no suffix Total FET noise
Resistor	#therm Resistor thermal noise #1overf Flicker (1/f) noise #noise total resistor noise
voltage controlled switch	no suffix Total switch noise

Creating Noise Info File

Noise analysis generates vectors in the same way as all other swept analyses. Individual vectors may also be tabulated in the list file using the .PRINT control. A noise output file may also be created from the front end. Select the command shell menu Graphs and Data|Create Noise Output File to create a text file with a summary of noise results. Included is a list of the integrated noise output for every device listed in order of magnitude. Select Graphs and Data|View Noise Output File to view the file. Note that this is a front end feature and is not implemented by the simulator.

Examples

Run noise analysis from 100Hz to 1MHz with 25 points per decade. Calculate noise at node named vout and noise referred back to voltage source vin:

```
      .NOISE V ( vout ) vin dec 25 100 1meg
```

Decade sweep resistor Rsource from 100 to 10K with 25 points per decade. Frequency = 1kHz

```
      .NOISE DEVICE Rsource DEC 25 100 10k F=1K
```

.op - Operating Point Analysis.

```
.op
```

This control instructs the simulator to perform a DC operating point analysis. Note that a DC operating point analysis is carried out automatically for transient (unless the UIC parameter is specified), ac, dc, transfer function and noise analyses.

DC operating point analysis attempts to find a stable bias point for the circuit. It does this by first applying an initial guess and then uses an iterative algorithm to converge on a solution. If it fails to find a solution by this method the simulator then attempts three further strategies.

For the first, a method known as "source stepping" is employed. For this all voltage and current sources in the circuit are initially set to near zero and the solution found. The sources are then gradually increased until they reach their final value.

If this approach fails a second strategy "GMIN stepping" is invoked. This conditions the solution matrix by increasing the diagonal term such that it is dominant. If large enough, convergence is virtually guaranteed. If successful then the diagonal term is reduced and a further solution sought using the previous solution as a starting point. This procedure is repeated until the diagonal term is returned to its correct value. Increasing the diagonal term is in a way similar, but by no means identical, to placing a small resistance at each node of the circuit.

If source stepping fails a final strategy, "pseudo transient analysis" is invoked. This is the most powerful technique employed and nearly always succeeds. However, it is also the slowest which is why it it left until last. For more information on DC convergence see later in this chapter on *Convergence* and section on *DC Operating Point - Overview.*

If the final approach fails then the analysis will abort.

IMPORTANT: *It is not necessary to include .OP if other analyses are specified.* All other analysis modes will perform an operating point anyway so including .OP will simply cause it to be done twice. However, with .NOISE, .TF, .SENS and .PZ the results of the operating point analysis are not output. If the bias point of the circuit is required when running one of these analysis modes, a .OP will be needed.

"OFF" Parameters

Some semiconductor devices feature the device parameter "OFF". If there are devices in the circuit which specify this parameter, the bias point solution is found in two stages. In stage 1 the devices with "OFF" specified are treated as if their output terminals are open circuit and the operating point algorithm completes to convergence. In stage 2, The "OFF" state is then released and the solution restarted but initialised with the results of stage 1.

The result of this procedure is that "OFF" devices that are part of latching circuits are induced to be in the OFF state. Note that the "OFF" parameter only affects circuits that have more than one possible DC solution such as bistables. If the "OFF" parameter is specified in - say - an amplifier circuit - with a unique solution, the final result will be the same. It will just take a little longer to arrive at it.

Nodesets

Nodesets work in a similar way to the "OFF" parameter in that the solution is found in two stages. In the first the "nodeset" is applied and the solution found. It is then released and convergence continues. Nodeset are an aid to convergence and, like the "OFF" parameter, can coerce a particular solution if there is more than 1 stable state.

Initial Conditions

Initial conditions force a particular voltage at a circuit node during bias point solution. The force is released for any subsequent analysis.

Operating point output file

During the operating point analysis, a list file is created which provides information on the operating values of every device in the circuit. This information is not usually output for other analysis modes unless explicitly requested. Three simulator options control the name and creation of this file. These are:

NOOPINFO	If set, the operating point info file is not created for .OP analysis
OPINFO	If set, the operating point info file is created for other analyses as well as .OP (except .SENS)
OPINFOFILE	Sets name of file to receive operating point info. Outputs to list file if this option is not specified.

See Also

Analysis Modes previously

Viewing DC operating point results later on

.options - Simulator options

```
.options [ opt1 [=val1]] ...
```

This control allows the setting of various options specific to the simulator.

opt1	Option name. Must be one specified in list below.
val1	Option value. Note, boolean options do not have a value. They are assigned “true” if the name is present and “false” if not.

List of simulator options

Option name	Default value	Units	Description
ABSTOL	1p	A	The absolute current error tolerance. It is sometimes desirable to increase this for circuits that carry large currents (>1A) to speed the solution and aid convergence.
ABSTOLMAX	1µ	A	ABSTOL selectively relaxed to this value if needed to allow transient analysis to continue. See later in this chapter *Convergence and accuracy issues*
ALLACI	false		Instructs simulator to save all device currents in an AC analysis. Usually, only currents for simple devices are evaluated and stored. Equivalent to “.KEEP /allaci”, but unlike .KEEP, .OPTION values can be defined on the Run command line.
ACCT	false		Full simulation timing statistics are generated if this is enabled.
BINDIAG	false		If enabled, a report about selection of binned models will be output to the list file. See Model Binning previously discussed.
CHGTOL	1e-14	Coulombs	The absolute charge tolerance. It is not usually necessary to change this value.
DCOPSEQUENCE	gmin|source| pta		Operating point strategy sequence order.
DEFL	100µ	metres	Default value for MOS channel length
DEFW	100µ	metres	Default value for MOS channel width (metres)
DEFAD	0	m^2	Default value for MOS drain diffusion area.
DEFAS	0	m^2	Default value for MOS source diffusion area m^2.
DIGMINTIME	1pS	Secs	Minimum digital resolution. Not yet fully supported
DEFNRD	0		As DEFAD but for NRD parameter
DEFNRS	0		As DEFAD but for NRS parameter
DEFPD	0	metres	As DEFAD but for PD parameter
DEFPS	0	metres	As DEFAD but for PS parameter
DEVACCT	False		If true, the simulator will measure load times for each device type during a simulator run. This information can be obtained using the GetDeviceStats() script function.

EXPAND	False		The netlist with subcircuits expanded is output to the list file if this is specified.
EXPANDFILE			Only applies if EXPAND also specified. Specifies a file instead of the list file to receive the expanded netlist
FASTPOINTTOL	1.0	off	Value for POINTTOL used during ‘Fast transient start’. See POINTTOL below.
FASTRELTOL	0.001		Value for RELTOL used during ‘Fast transient start’.
FLUXTOL	1e-11	V.secs	The absolute flux tolerance for inductors. It is not usually necessary to change this value.
FORCETRANOPGROUP	Off		Forces a separate data group to be created for transient analysis operating point data. This happens anyway if tstart>0. Use this option when simulating a large circuit and you wish to make extensive use of schematic bias annotation.
DISABLESUBCKTMULTIPLIER	False		If true, the subcircuit multiplier parameter, M, will be disabled. See “Subcircuit Instance”
FULLEVENTREPORT	False		If true, the simulator will save all event information. When false, only major events are recorded. Events can be obtained from the GetSimulatorEvents() script function and can be useful for diagnosing failed runs.
GMIN	1e-12	Siemens (mhos)	The minimum conductance allowed by the program. This has the effect of placing a resistor = 1/GMIN in parallel with every branch of the circuit.
GMINMAXITERS	1000		Maximum total number of iterations allowed for GMIN stepping operating point algorithm.
GMINMULT	10		Multiplier at each step of GMIN stepping method.
GMINSTEPS	10		Number of steps used for GMIN stepping method of finding dc operating point. See *.op (Simulator Control).*
GMINSTEPITERLIMIT	20		Iteration limit for each step in GMIN stepping.
ICRES	1	Ohms	Initial condition resistive force.
ITL1	100		DC iteration limit.
ITL2	50		DC sweep iteration limit.
ITL4	10		Normal transient timepoint iteration limit.

ITL7	40		Upper transient timepoint iteration limit. This is specific to Pulsonix Spice. See *Convergence and accuracy issues*
LOGICTHRESHIGH	5	V	Output voltage for logic high level. Used for & | and ~ operators for arbitrary source.
LOGICTHRESLOW	0	V	Output voltage for logic low level. Other comments as for LogicThreshHigh
LOGICHIGH	2.2	V	Upper threshold for logic inputs. Other comments as for LogicThreshHigh
LOGICLOW	2.1	V	Lower threshold for logic inputs. Other comments as for LogicThreshHigh
LOGPARAMEXPRESSIONS	False		If true, a log of parameter expressions in use will be output to the list file
MATCHEDSUBCIRCUITS	off		If set, components within subcircuits are treated as matched for Monte Carlo analysis. See “Monte Carlo Analysis”.
MAXEVTITER	0 (sets Internal default)		Maximum number of event driven passes allowed at each step. It is not usually necessary to change this value.
MAXOPALTER	0 (sets Internal default)		Maximum number of alternations between analog and event-driven. iterations. It is not usually necessary to change this value.
MAXORD	2		Maximum integration order. For "METHOD=TRAP" max value is 2. For "METHOD=GEAR" max value is 6. There are few advantages to changing this value.
MAXVDELTAABS	0.5		with MAXVDELTAREL, sets a limit on the amount of change per timestep for each node. Inactive if MAXVDELTAREL less than or equal to zero. See “Voltage Delta Limit” for further details.
MC_ABSOLUTE_RECT	Off		If set Monte Carlo distribution will be rectangular for absolute tolerances. Otherwise the distribution will be Gaussian.
MC_MATCH_RECT	Off		If set Monte Carlo distribution will be rectangular for matched tolerances. Otherwise the distribution will be Gaussian.
MCLOGFILE	Mclog.txt		File name to receive Monte Carlo log.
MAXVDELTAREL	0.0		See MAXVDELTAABS above.
METHOD	trap		Numerical integration method. Either "TRAP" (default) or "GEAR". More info: See later *Convergence and accuracy issues*

MINBREAK	5e-5 X *tmaxstep*	secs	Minimum time between transient analysis breakpoints. A breakpoint is a point in time when an analysis is forced regardless of whether it is required by the timestep selection algorithm. Typically they are set at known turning points such as the start and end of a rising pulse. If two breakpoints are closer than MINBREAK they are merged into one. (there are exceptions to this e.g if the two breakpoints were generated by a single rising edge). Increasing MINBREAK can help convergence and simulation speed especially if transmission lines are used. (*tmaxstep* is the last parameter of the .tran control.)
MINGMINMULTIPLIER	1.000001		In GMIN stepping, the step size is multiplied by variable factor at each step. This step is reduced if convergence fails. If it is reduced below this value, the simulation will abort.
MinTimeStep	1e-9 * Max time step	Secs	Minimum transient time step. Simulation will abort if it reaches this value.
MOSGMIN	GMIN		Value of GMIN used between drain and source of MOSFETs.
NEWGMIN	False		Changes the implementation of GMIN for 'old' MOS devices i.e. LEVELs 1-3. When this option is set, GMIN is implemented as a conductance between source and drain. Otherwise two conductance's are added between drain and bulk and source and bulk.
NOCUR	False		Equivalent to ".KEEP /noi". Inhibits the saving of current data.
NODELIMIT	1e50	Volts	Maximum value allowed for circuit node during iteration. If exceeded, iteration will abort. (This does not usually mean the analysis will abort). Reducing this value can sometime solve floating point exceptions or unexplained singular matrices.
NODESETRES	1.0	Ohms	Driving resistance of node set force.
NOECHO	False		Inhibits display of netlist in list file.
NOMCLOG	False		If specified, no Monte Carlo log file will be created.
NOMOD	False		If specified, no model parameter report will be output to the list file.

NOMS9GATENOISE	False	If specified, the drain induced gate noise model for MOS9 devices will be disabled.
NOOPALTER	False	If specified, only a single pass will be made to resolve the operating point for event driven devices.
NOOPINFO	off	Switches off creation of operating point info file for .OP analyses.
NOOPITER	off	Use GMIN stepping for DC operating point analysis first. (i.e skip normal iteration method)
NORAW	off	Output transient analysis values at intervals of *tstep* only. See *.tran (Simulator Control)*
NOSENSFILE	off	Switches off creation of sensitivity analysis data file.
NOVOLT	False	Equivalent to ".KEEP /nov". Inhibits the saving of voltage data.
NOWARNINGS	False	Inhibits simulation warnings
NUMDGT	10	Column width used for display of all values in list file and Monte Carlo log file. Minimum value is 8, maximum is 30. Note this value is column width not the number of significant digits.
OLDLIMIT	Off	If set SPICE 2 MOS limiting algorithm is used.
OLDMOSNOISE	Off	MOS2 and BSIM3 devices return device noise in V^2/Hz for SPICE3 whereas other device's noise is returned in V/√Hz. With Pulsonix Spice all devices return noise in V/√Hz. Setting this option restores to behaviour of standard SPICE..
OPINFO	Off	If set DC operating point info file is created for all analyses. Normally it is created only for .OP analyses.
OPINFOFILE	Off	Specify name of operating point info file. This is OP.TXT by default.
OPTIMISE	1	Controls expression optimiser. 0=off, 1=on for .FUNC defined expressions, 2=on always. See "Optimisation" for details.

PARAMLOG	Given	Control amount of detail for parameter log in list file. Choices: None: no parameters listed Brief: only parameters specified using an expression are listed Given parameters explicitly specified in the netlist are listed Full all parameters are listed
PIVREL	1e-3	This affects the matrix solution. Setting this parameter to a high value e.g. 0.99 can sometimes fix convergence problems but will slow down the simulation. Valid values lie between 0 and 1. Mathematicians may like to know that it is the ratio of the largest column entry and an acceptable pivot value and is sometimes known as *drop-tolerance*.
PIVTOL	1e-13	The absolute minimum value for a matrix entry to be accepted as a pivot. Unexplained "Singular matrix" errors can sometimes be overcome by lowering this value. (But note that "Singular matrix" errors are usually caused by errors in the circuit such as floating nodes or shorted voltage sources).
POINTTOL	0.001	A factor used to control the extent to which the maximum value attained by a signal is used to control its tolerance. This is new from release 2; set it to zero for pre-release 2 behaviour. Increasing this value will speed up the simulation at the expense of precision.
PTAACCEPTAT	0	If > 0, specifies a time when pseudo transient analysis results will be accepted unconditionally. This is useful when a circuit comes close to convergence during pseudo transient, but doesn’t quite make it due to an oscillation. See “Pseudo Transient Analysis”.
PTACONFIG	0	Integer from 0 to 15 sets internal parameters for pseudo transient algorithm used to find DC operating point. See “Pseudo Transient Analysis”.
PTAMAXITERS	20000	Maximum total number of iterations allowed for pseudo transient algorithm used to find DC operating point. See “Pseudo Transient Analysis”.

PTAOUTPUTVECS	false	If specified, signal vectors will be output during pseudo transient analysis. This may be used to diagnose a failure. See “Pseudo”
RELTOL	0.001	This is the relative tolerance that must be met for each analysis point. Reducing this number will improve accuracy at the expense of simulation time or/and convergence reliability. Simulation results cannot be relied upon if its value is increased beyond 0.01. A more detailed discussion is given in the section on *Convergence and accuracy issues*
RELTOLMAX	0.01	RELTOL selectively relaxed to this value if needed to allow transient analysis to continue. See the section on *Convergence and accuracy issues*
RESTHRESH	1e-6	Resistance threshold. If a resistor is specified that is below this value, Pulsonix Spice will use a voltage based implementation (V=IR) instead of the conventional current based implementation (V=I/R). Voltage based resistors are slightly less efficient but allow R=0 without numerical overflow.
RSHUNT	Infinite	If specified a resistor of the specified value is placed from every node to ground. This can resolve problems with floating nodes.
SEED	0	Integer value. If non-zero will be used to initialise random number generator used for Monte Carlo analysis distribution functions. See *Seeding the Random Number* Generator
SENSFILE	SENS.TX T	Specify name of sensitivity data file.
SOADERATING	1.0	Scales min and max values used in “.SETSOA” specification. This allows a de-rating policy to be globally applied to SOA limits.
SOAEND	-1	Specifies end time point for SOA. Use with SOASTART. -1 means end of simulation.

SOAMODE	off		Controls the Safe Operating Area (SOA) test mode. See “.SETSOA” below for details on how to define a SOA test.	
			Can set to:	
			Off	SOA testing is not enabled. In this mode .SETSOA controls will be read in and any errors reported, but no SOA testing will be performed during the run.
			Summary	SOA testing enabled and results given in summary form with only the first violation for each expression given being output.
			Full	SOA testing enabled with full results given. Every violation will be reported in this mode.
SOAOUTPUT	list		Can be:	
			msg	Results displayed in command shell message window, or console if run in “non-GUI” mode
			list	Results output to list file
			msg|list	Results output to both list file and command shell message window
			none	Results not output to either list file or message window.
			Note that all results are always stored for retrieval using the script function GetSOAResults described below. So even if “none” is specified the SOA data is always available.	
SOURCEMAXITERS	1000		Maximum total number of iterations permitted for source stepping algorithm. Set to zero to disable limit	
SOASTART	0.0		Specifies start time for SOA. Use with SOAEND	
SOAWRITEDEFS	False		If specified, SOA definitions are written to the list file.	
TEMP	27	(°C)	Operating temperature of circuit. Note this value can be overridden locally for some devices. You can also use .TEMP for this.	

TIMESTATS	Off		Full simulation timing statistics are generated if this is enabled. Note that this can slow down runs for small circuits on Windows 95 and 98 platforms.
TLMINBREAK	See note		Minimum break point for transmission lines. Works in the same way as MINBREAK but only for break points generated by lossless transmission lines. Default = MINTIMESTEP * 5e-5
TNOM	27	(°C)	Temperature at which model parameters are defined. This can be overridden in the model control.
TRTOL	7		This only affects transient analysis. It is a relative value that is used to determine an acceptable value for the "local truncation error" before an analysis point is accepted. Reducing this value cause the simulator to model the effects of energy storage elements more accurately at the expense of simulation time.
TRYTOCOMPACT	Off		Forces compaction of data for lossy transmission lines. This speeds up simulation at the expense of accuracy.
VNTOL	1μ	V	The absolute voltage error tolerance. Circuits with large voltages present (>100) may benefit from an increase in this value.
VNTOLMAX	1m	V	VNTOL selectively relaxed to this value if needed to allow transient analysis to continue. See the section on *Convergence and accuracy issues*
WIDTH	80		Number of columns used for list file output. This may be set to any reasonable value and not limited to the choice of 80 or 132 as with SPICE2
WIRETABLE	None		Define file containing wire table used for the digital simulator’s wire delay.

.param - Simulator parameters

.PARAM *parameter_name* [=] *parameter_value* [*parameter_name* [=] *parameter_value*]...

Defines a simulation variable for use in an expression. Expressions may be used to define device parameters, to define model parameters, for arbitrary sources and to define variables themselves. See “Using Expressions” for details.

parameter_name	Sequence of alpha-numeric characters. Must begin with a letter or underscore. May not contain spaces.

parameter_value	Either: A constant OR An expression enclosed by '{' and '}'.

.PARAM Examples

```
.PARAM Vthresh 2.4
.PARAM Vthresh {(Vhigh+Vlow)/2}
.PARAM F0 1k Alpha 1 C1 {2*c2}
.PARAM R1 {2/(2*pi*freq*C1*alpha}
```

Constants

The following constant values may be used

E	2.71828
PI	3.14159
TRUE	1
FALSE	0

Note that these values may not be used in arbitrary source expressions unless enclosed by '{' and '}'.

Global Simulation Parameters

The parameter TEMP may be used. This returns the circuit temperature in Celsius. The value is updated during temperature sweeps.

How parameter expressions are evaluated

The following describes the sequence in which netlist expressions are evaluated.

1. When the netlist is read in, an attempt is made to evaluate any expressions making only the values of command line parameters available. If the evaluation is successful, the expression is substituted with the result of the evaluation. If it is not the expression is passed on to the simulator.

2. When the simulation starts all constant .PARAM values are stored, then any .PARAM controls which themselves are an expression are evaluated. At this stage, any swept parameters (in a .DC control) are not made available nor is the global TEMP parameter for a temperature sweep.

3. At this stage, unless the analysis is a temperature sweep or parameter sweep, all expressions should have been successfully evaluated. If any remain, an error will be flagged and the simulation will abort. For the case of temperature or parameter sweeps some expressions that depend on the swept parameter will remain unresolved. These will be evaluated at each stage of the sweep.

Parameter Rules

Expressions that depend only on command line values can be placed anywhere in the netlist except for the device name; they can even be used for node names (although we would advise great caution if doing this). They don't have to be numeric expressions they can resolve to a text string. Expressions that depend on global (i.e. TEMP) or .PARAM parameters may only be used for instance parameters, model parameters and parameters passed to subcircuits.

Parameters defined in .PARAM controls may be used in arbitrary source expressions directly. E.g.

```
.PARAM Vthresh 2.3

B1 n1 n2 v=(v(n4)-Vthresh)*v(n3)
```

Parameters defined on the command line may not be used in this way although the following is legal:

In script or command line:

```
Let Vthresh=2.3
```

In netlist

```
B1 n1 n2 v=(v(n4)- {Vthresh} )*v(n3)
```

.PARAM netlist order

.PARAM controls that resolve to a constant are order independent ; they can be placed anywhere in a netlist. They can even be placed after another .PARAM expression that depends on its value (but note this does not apply in subcircuits). .PARAM controls that are defined as an expression that depends on other .PARAM's also defined as an expression must be placed in sequential order. For example, the following is OK:

```
.PARAM C2 {C1*alpha*alpha/4}

.PARAM C1 1n

.PARAM alpha 1

.PARAM R1 {2/(2*PI*F0*C2*alpha}
```

The first .PARAM depends on `alpha` and `C1` which are defined later in netlist. This is OK (as long as it is not in a subcircuit) because `alpha` and `C1` are constants. The fourth .PARAM depends on `C2` which is defined as an expression. The definition for must - and does in the above example - come before the definition of `R1`. The following would yield an error as the definition for `C2` comes after the definition of `R1`:

```
.PARAM R1 {2/(2*PI*F0*C2*alpha}

.PARAM C1 1n

.PARAM alpha 1

.PARAM C2 {C1*alpha*alpha/4}
```

Note that .PARAM's inside subcircuits are local to the subcircuit. This is explained in next section.

Subcircuit Parameters

Parameters may be declared within sub circuits. E.g.

```
.subckt ADevice n1 n2 n3 n4

.PARAM Vthresh 3.5
```

```
...
...
ends
```

In the above example, in reference to Vthresh within the subcircuit would use the value declared by the .PARAM declared inside the subcircuit. That value would not be available outside the subcircuit definition. Parameters may also be passed to subcircuits. E.g.

```
X1 1 2 3 4 ADevice : threshold=2.4
```

or

```
X1 1 2 3 4 ADevice params: threshold=2.4
```

Any reference to `threshold` within the subcircuit definition would use that value.

Default values for parameters may also be specified in subcircuit definition:

```
.subckt ADevice n1 n2 n3 n4 params: threshold=2.4
...
.ends
```

If that subcircuit is called without specifying `threshold` the default value of 2.4 will be used. Note that it is not compulsory to declare default values.

.PARAM in Libraries

.PARAM controls may be included in libraries specified using .LIB or by global definitions. Pulsonix Spice will search such libraries for any parameters used in expressions that are not found in the netlist.

Using .PARAM in Schematics

.PARAM controls may be appended to the netlist created by the schematic editor.

.POST_PROCESS

.POST_PROCESS *scriptname* [*arguments*]

Invokes the Pulsonix Spice script *scriptname* at the end of a successful simulation. If present *arguments* will be passed to the script as a single string.

scriptname may the name of an embedded file defined using .FILE and .ENDF. For example, the following will cause the text “Simulation Complete” to be displayed in the command shell when the run is complete:

```
    .FILE on_complete
    Echo "Simulation Complete"
    .ENDF
    .POST_PROCESS on_complete
```

.POST_PROCESS may be used to perform measurements on simulation results for display in the command shell or written to a file.

.POST_PROCESS scripts will function even if the simulator is operating in a standalone mode in which case any displayed messages created from, for example, Echo or Show, will be directed to the simulator's output device. In console mode, this would be the console or terminal and in standalone GUI mode, this would be the message window in the simulator status box. As there is no environment available in the standalone mode, not all script commands and functions will be available.

.PRINT

.PRINT TRAN|AC|DC|NOISE|TF *vector*| *{expression}* ….

Instructs the simulator to output selected simulation data to the list file in tabulated form.

Where:

vector	Name of vector to print. May be in Pulsonix Spice native format or traditional SPICE format (see notes below).
expression	Arithmetic expression of vectors

Notes

A traditional SPICE2 command, this was not supported by Pulsonix Spice until release 2.0. It is SPICE2 compatible but also supports some additional features:

- NOISE and TF results may be output as well as TRAN, AC and DC
- You can put expressions as well as single values enclosed in '{' and '}'. E.g.

```
.PRINT TRAN {vout-q5_c}
```

You can use the SPICE2 style method of accessing single voltages, differential voltages and device currents. These are of the form:

Single ended voltage

```
funcname(nodename)
```

Differential voltage

```
funcname(nodename, nodename)
```

Device current

```
funcname(device_name)
```

Where:

Funcname	Function to be applied. For available list, see below.
nodename	Node name as specified in the netlist.
device_name	Name of device for current.

Available functions:

Function Name	**Argument**	**Analysis mode**	**Meaning**
V	node name	Transient	Voltage at node
V	node name	AC	Voltage magnitude at node
VM	node name	AC	Voltage magnitude at node
VP	node name	AC	Voltage phase at node
VR	node name	AC	Real voltage at node

VDB	node name	AC	dbV at node
VG	node name	AC	group delay at node
I	two term. device name	TRAN	Current in device
IB	BJT name	TRAN	Base current
IB	MOSFET name	TRAN	Bulk current
IC	BJT name	TRAN	Collector current
ID	MOSFET/JFET name	TRAN	Drain current
IE	BJT name	TRAN	Emitter current
IG	MOSFET/JFET name	TRAN	Gate current
IS	MOSFET/JFET name	TRAN	Source current
IS	BJT name	TRAN	Substrate current
IM	Two term device	AC	Device current
IP	Two term device	AC	Current phase
IR	Two term device	AC	Current real part
II	Two term device	AC	Current imaginary part
IDB	Two term device	AC	Current dB
IG	Two term device	AC	Current group delay

.PRINT controls may be placed inside a subcircuit definition in which case the device and node names refer to local devices and nodes. Output will be listed *for every instance of the subcircuit.*

For transient analysis the results are displayed at the interval specified by the time step parameter on the .TRAN control. If this is zero or omitted, it defaults to (tstop-tstart)/ 50. The data is created by interpolation unless the NORAW option is specified in which case a time step is forced at the time step interval.

Examples

```
.PRINT TRAN V(VOUT)
.PRINT TRAN VOUT
.PRINT TRAN V(VPos, VNeg)
.PRINT TRAN {Vpos-VNeg}
.PRINT AC VDB(VOUT)
```

.pz (Pole-zero analysis)

.PZ *N1 N2 N3 N4* CUR|VOL POL|ZER|PZ

N1, N2	Input nodes
N3, N4	Output nodes
CUR	transfer function is of the type (output voltage)/(input current)
VOL	transfer function is of the type (output voltage)/(input voltage)

Usually the last parameter would be PZ which instructs the simulator to find both poles and zeros. The alternatives instruct it to find one or the other. This may be used if the simulator aborts because it didn't converge on poles or on zeros, at least it can be instructed to find the other.

To view the results of the pole-zero analysis select the command shell menu **Graphs and Data|List Pole-zero|Transfer Function results**. The poles and zeros will be listed in complex form.

.sens - Sensitivity Analysis

.sens v(*nodename* [,*refnodename*])| i(*sourcename*)

This control instructs the simulator to perform a DC sensitivity analysis. In this analysis mode, a DC operating point is first calculated then the linearised sensitivity of the specified circuit voltage or current to every model and device parameter is evaluated. The results are output to a file (OUTPUT.TXT by default but can be changed with SENSFILE option) and they are also placed in a new data group. The latter allows the data to be viewed in the message window (type Display) at the command line and can also be accessed from scripts for further analysis.

.SETSOA

.SETAOA [LABEL=*label*][MODEL=*modelname* | INST=*instname*]

[DEVICE=*device*] [DERATING=*derating*] [MEAN]

[ALLOWUNUSED][ALLOWWILD]

expr1=(min1, max1[, xwinow1])

[*expr2*=(min2, max2[, xwindow2])…]

Defines a Safe Operating Area (SOA) specification. If SOA testing is enabled the simulator will check simulated results against this specification and record any violations.

The results of SOA testing are output to the list file by default and can optionally also be displayed in the command shell message window, or console window if run in non-GUI mode. They are also always available via a script function GetSOAResults(). See .OPTIONS setting “SOAOUTPUT” for more details.

label	Optional label that will be included in every violation report. You can use the following symbolic values in this label: %INST% - substituted with the instance name that violated the specification. This is only meaningful if MODEL or INST are specified. (See below). %MODEL% - substituted with the model name that violated the specification. Only meaningful if MODEL is specified. (See below). %EXPR% - substituted with the expression that violated the specification. %SUBCKT% - applicable if the .SETSOA command is located within a .SUBCKT definition. Value is substituted with the subcircuit instance reference.
instname	If specified the expression or expressions supplied in expr1 etc. are applied to the specified instance (e.g. Q23, M10, R56). In this case the expression may refer to node voltages and pin currents of the specified instance. See details under expr1, expr2...
modelname	If specified the expression or expressions supplied in expr1 etc. are applied to every instance belonging to modelname. In this case the

	expression may refer to node voltages and pin currents for each instance processed. See details under expr1, expr2...
device	If INST or MODEL is specified using a wildcard specification, only instances of the specified device type will be processed. For example: .SETSOA INST=* DEVICE=resistor... will be applied to all resistors in the circuit.
Derating	Derates limit specification by specified factor. Default is 1.0 which means no derating. Value must be greater than 0. An expression containing values defined using .PARAM may be used.
expr1, expr2...	Expression to be evaluated and compared against minimum and maximum specs. This expression can access simulation results using access variables. The format and scope of these variables depends on whether MODEL, INST or neither is specified. If neither is specified, the expression can use the global access variables defined below:

Syntax	**Function**	**Example**
nodename	Voltage on node	VOUT – voltage on node VOUT
n(nodename)	Voltage on node	n(VOUT) - voltage on node VOUT
instname#param	Instance parameter	M2#vdsat – vdsat value for M2 Q23#c – current in collector of Q23
paramname	Parameter defined using .PARAM	

If there is a clash between a paramname and nodename, that is if the same name could refer to either a node or a parameter, then the parameter name takes precedence. To access the node in this case, use the n(*nodename*) syntax.

Use the following values if MODEL or INST is specified. In each case (excepting the global access variable) the variable accesses a quantity for the instance being processed. With INST this will be the single instance specified by instname. With MODEL all instance belonging to the model specified by modelname will be processed.

Syntax	**Function**	**Example**
pinname	Current in pin	c - current in collector of transistor
Ipinname	Current in pin	Ic - current in collector of transistor
Vpinname	Voltage on pin	Vc - voltage on collector of transistor
n(pinname)	Voltage on pin	n(c) - voltage on collector of transistor
Vxy Where x = pin name 1, y= pin	Voltage between x and y.	Vbc – voltage from base to collector

name 2. Both x and y must be single letters		
pow	Power in device	
param	Readback parameter	vdsat - ‘vdsat’ for MOSFET
#global_name	Global node voltage or pin current	#VOUT – voltage on net called VOUT #q23#c – current in collector of q23
paramname	Parameter defined using .PARAM	

Note that currently the use of V() and I() is not accepted and will result in an error message being displayed.

min, max	Minimum and maximum values respectively. A violation message will be produced if the value of the associated expression is less than min or greater than max. Use ‘*’ if the limit is to be ignored. E.g. (*, 15) will test a maximum value of 15 but the minimum value will not be tested. min and max values may be scaled using the SCADERATING .OPTIONS setting (see below for details). Currently, only constant values are accepted for min and max.These values may be entered as expressions containing variables defined using .PARAM.
xwindow	Optional value specifies a minimum window that must be surpassed before limit violations are registered. For example if 10u is specified for xwindow for a transient analysis, then the limit must be exceeded continuously for at least 10uS before the violation is recorded. These values may be entered as expressions containing variables defined using .PARAM.
ALLOWUNUSED	If INST or MODEL are specified, an error will result if no instances to be processed are found. If INST is specified the error will occur if instname doesn’t exist. If MODEL is specified, the error will occur if there are no instances using modelname even if modelname itself is valid. This error will be inhibited if ALLOWUNUSED is specified
ALLOWWILD	If specified, wildcards can be used for modelname and instname. In this case Pulsonix Spice will search for all devices that match the wildcard specification. Use ‘*’ to match any sequence of characters and ‘?’ to match a single character.
MEAN	If specified all tests will be on the mean of the test expression over the whole simulation run.

Examples

Test the voltage on the ‘p’ pin of R1. Will fail if it exceeds 0.5V

```
.setsoa INST=R1 vp=(*,0.5)
```

Test the power dissipation of R2. Fails if it exceeds 0.5mW

```
.setsoa INST=R2 pow=(*,0.5m)
```

Test the current into pin ‘p’ of R3. Fails if it exceeds 0.5mA

```
.setsoa INST=R3 ip=(*,0.5m)
```

Test the voltage across R4. Fails if it exceeds 0.85V for at least 100uS. Will be reported using label “%INST%, high”, which resolves to “R4, high”

```
.setsoa LABEL="%INST%, high" INST=R4 vd=(*,0.85,100u)
```

Test the voltage across R4. Fails if it exceeds 0.7V for at least 500uS

```
.setsoa LABEL="%INST%, low" INST=R4 vd=(*,0.7,500u)
```

Tests voltage between ‘c’ and ‘e’ pins for all instances of model N1. Fails if voltage drops below -0.5V or exceeds 25V

```
.setsoa MODEL=N1 vce=(-0.5,25)
```

Tests power all devices of type resistor. Fails if this exceeds 0.25W.

```
.setsoa INST=* ALLOWWILD DEVICE=resistor pow=(*,0.25)
```

Tests the mean power in instance Q1. Fails if it exceeds “2*bjtderating”. “bjtderating” must be defined using a .PARAM statement.

```
.setsoa LABEL="%INST%, pow(q1)" INST=Q1 MEAN pow=(*,2)
derating=bjtderating
```

Calculates the expression “n(c)*(q1#c-d1#p)+n(b)*q1#b+n(e)*(q1#e+d1#p)” and fails if its mean exceeds 1.0. Violations will be reported using label “%SUBCKT%, power”. Statement is intended to be placed in a subcircuit definition block and “%SUBCKT%” will resolve to the reference of the subcircuit call.

```
.setsoa LABEL="%SUBCKT%, power" MEAN
"n(c)*(q1#cd1#p)+n(b)*q1#b+n(e)*(q1#e+d1#p)"=(*,1)
```

.subckt

.subckt *subcktname n1* [*n2*]... [params: *param_name1 param_value1* [*param_name2 param_value2*]...]

This control begins a subcircuit definition.

subcktname	Subcircuit name. This must begin with a letter but may contain any legal ASCII character except any whitespace (space, tab) or '.'. The name must be unique i.e. no other subcircuits may have the same name.
n1, n2 etc.	Node names available externally. Must not be zero.
param_name1,2 etc. param_value1,2 etc.	Parameter name and value. This sets default values for parameters used within the subcircuit. These values can be overridden for each subcircuit instance. Note that it is not compulsory to declare default values for subcircuit parameters.

IMPORTANT: Either the *params:* specifier or the first '=' may be omitted *but not both*. If both are omitted it becomes impossible for the netlist scanner to tell the difference between parameter names and node names.

.ENDS [*subcktname*]

Terminates a subcircuit definition. *subcktname* may be added for clarity but will be ignored by Pulsonix Spice.

A subcircuit consists of a .subckt control followed by a series of device or model descriptions and terminating in a .ends control. A subcircuit is a circuit that can be called into the main circuit (or indeed another subcircuit) by reference to its name. The .subckt control is used to define the subcircuit while a subcircuit call - an 'X' device - is used to create an instance of that subcircuit. Subcircuits have a number of uses:

To repeat a commonly used section of circuit.

To hide detail from the main circuit to aid circuit readability.

To distribute models of integrated devices such as op-amps.

For a detailed discussion see chapter on *Device Reference* and *Subcircuits*.

Subcircuit definitions usually reside in a text file and are read in as libraries.

.temp

.temp *temperature*

This control sets the default simulation temperature. Some devices can override this on a per instance basis. Units are degrees centigrade.

.TF

.TF *inner_sweep_spec* [V] *pos_out_node* [VN] *neg_out_node*
+ [[INSRC] *in_source*] [F frequency] [SWEEP *outer_sweep-spec*]

.TF *inner_sweep_spec* I *source* [INSRC *in_source*]
+ [F *frequency*] [SWEEP *outer_sweep-spec*]

Spice Compatible:
.TF V(*pos_out_node* [, *neg_out_node*]) *in_source*

.TF I (*source*) [INSRC] *in_source*

This control instructs the simulator to perform a small signal transfer function analysis.

pos_out_node	Output node.
neg_out_node	Output reference node. Defaults to ground if omitted for standard SPICE syntax.
in_source	Name of input source to which input noise will be referred.
inner_sweep_spec	See General Sweep Specification” for syntax. Defines sweep mode.
outer_sweep_spec	If specified, analysis will be repeated according to this specification. See “General Sweep Specification” for syntax.
frequency	Frequency at which analysis will be performed for non-frequency sweeps. Default 0

source	Voltage source to specify output current.

Notes

The Pulsonix Spice transfer function analysis remains syntax compatible with the SPICE version but is substantially enhanced. The SPICE version performs the analysis at a single point with frequency = 0. The Pulsonix Spice implementation performs a swept analysis using the same sweep algorithm used for AC, DC and NOISE>

Transfer function analysis is similar to AC analysis in that it performs a swept small signal analysis. However, whereas AC analysis calculates the response at any circuit node from a (usually) single input source, transfer function analysis calculates the individual responses from each source in the circuit to a single specified output node. This allows, for example, the series mode gain, common mode gain and power supply rejection of an amplifier to be measured in one analysis. The same measurements could be performed using AC analysis but several of them would need to be run. Transfer function mode also calculates output impedance or admittance and, if an input source is specified, input impedance.

The names of the output vectors will be of the form

Input voltage, output voltage
source_name#Vgain

Input voltage, output current
source_name#Transconductance

Input current, output voltage
source_name#Transresistance

Input current, output current
source_name#Igain

Output impedance for voltage out will be called Zout. For a current output, the output admittance will be calculated and will be named Yout.

If an input source is specified the input impedance will be calculated and called Zin.

Note that although the syntax for .TF retains compatibility with SPICE and earlier versions of Pulsonix Spice the output provided is slightly different. firstly, the data is complex even if F=) and secondly the names of the output vectors are different as detailed above.

Examples

SPICE compatible. Outputs results at DC.

```
.TF V (Vout) Vin
```

As above but decade sweep from 1k to 100k

```
.TF FREQ DEC 25 1K 100K V(Vout, 0) Vin
```

Note that in the above example the ‘0’ in V(Vout, 0) is compulsory. If it is omitted, Vin will be assumed as the reference node.

.trace

```
.trace vector_name  [vector_name ...] graph_id
```

Set up a *trace*. This is graph plot that is updated as the simulation runs.

Where *vector_name* is the name of a net or pin
graph_ id is an integer between 1 and 999 to specify which graphs traces should use - see explanation below

graph_ id is an arbitrary number that makes it possible to direct traces to different graphs. Two traces with the same id will be always be put in the same graph. Traces from subsequent simulations with that id will also go to that graph if it still exists otherwise a new one will be created. To force two traces to go to separate graphs, use different id's. Note that it doesn't matter what the id's value actually is - it could be 1 or 100 - as long as traces that must go to the same graph use the same value.

Note that the "AutoAxis" feature available for normal plotting also works for Traces. So if a current and voltage trace are both directed to the same graph, separate axes will be created for them.

Examples

.trace v1_p 1 q1#c 1

In the above example a voltage - v1_p - and a current - q1#c - will both be traced on the same graph. As they have different units, the AutoAxis feature will force the curves to two different y axes.

.trace v1_p 1 q1#c 2

In this example the voltage and current traces will be directed to different graph sheets.

Notes

The .TRACE control has now been largely superseded by the .GRAPH control, which is much more flexible. However, the .TRACE control is still useful for specifying multiple traces on a single line. .GRAPH can only specify one signal at a time.

.TRAN

.TRAN *tstop*

OR

.TRAN*tstep tstop* [*tstart* [*tmaxstep*]] [UIC]
+ [FAST=*fast_start*]
+ [SWEEP *sweep_spec*]

This control instructs the simulator to perform a transient analysis. In this mode the simulator computes the behaviour of the circuit over the specified time interval. The circuit’s currents and voltages are calculated at discrete time points separated by a variable time step. This time step is adjusted automatically by the simulator according to circuit activity. The circuit may contain any number of time varying voltage and current sources (stimuli) to simulate external signals, test generators etc.

tstep	This defines the interval for tabulated results specified by the .PRINT control. It also defines the output interval for all data if the NORAW option is specified. If there are no .PRINT controls in the netlist and NORAW is not being used, this can be set to zero or omitted altogether as in form 1 above. If set to zero it defaults to (*tstop*-*tstart*)/50 *tstep* is also used to define default values for pulse and exponential stimuli. Note that if *tstep* and NORAW are specified a time point is forced at *tstep* intervals to calculate the output. This differs from other SPICE programs which generate output at *tstep* by interpolation. *tstep* does not control the time step used by the simulator. This is controlled automatically according to circuit activity.
tstop	Stop time. Note that if running in GUI mode, a transient analysis can be restarted from the front end using the RestartTran command or from the schematic using the Restart Transient option.
tstart	Start time. This is the time at which the storage of transient analysis outputs commences. It is not the time at which the analysis begins; this is always zero. *tstart* is zero if it is omitted.
tmaxstep	Maximum time step. The simulator uses the largest time step possible to achieve the required accuracy but will not increase it beyond this value. If not specified it is set to (*tstop-tstart)*/50.
UIC	If specified a DC operating point is not calculated and initial condition specifications are used instead
fast_start	If specified, the simulation will run at reduced accuracy but higher speed for the time specified by this parameter. The reduced accuracy is implemented by altering a number of tolerances and internal parameters. See notes below for more details.
sweep_spec	If specified, analysis will be repeated according to this specification. See "General Sweep Specification" for syntax.

Fast Start

If the FAST parameter is specified, the simulation will begin with a number of tolerances and internal parameters altered to speed up the simulation at the expense of accuracy. Just before the end of the fast start period, these tolerances and parameters will be gradually restored to their normal values. Fast start is an aid for simulating circuits such as switching power supplies and oscillators for which the initial start-up period is not of interest but takes a long simulation time. Note that although the fast start interval can run sometimes as much as twice as quickly as normal, the fact that accuracy is impaired can mean that the final steady state reached may not be very accurate. This means that after the fast start period, an additional settling time may be required for full accuracy to be reached.

Start sets the values of POINTTOL and RELTOL according to the value specified by FASTPOINTTOL and FASTRELTOL respectively.

Convergence

Overview

In transient and DC analyses, an iterative method is used to analyse the circuit. Generally, iterative methods start with an initial guess for the solution to a set of equations and then evaluate the equations with that guess. The result of that evaluation is then used to derive a closer estimate to the final solution. This process is repeated until a solution is found that is within the error tolerance required. Pulsonix Spice and SPICE use a technique known as Newton-Raphson iteration (also known as "Newton's Method") which usually converges extremely rapidly. However, there are occasions when this process is either unreasonably slow or fails altogether. Under these circumstances the simulation will abort.

Pulsonix Spice offers superior convergence to all other products in its price bracket and possibly all PC based simulators generally. In a test on 57 public domain benchmark circuits[1], Pulsonix Spice passed all netlist tests, whereas standard SPICE3 achieves about 62%. This performance has been achieved as a result of the following developments to the simulator core.

- Automatic pseudo transient analysis algorithm for operating point solution. See below for details.
- Enhancements to GMIN and source stepping algorithms to use a variable step size. (The standard SPICE3 variants use a fixed step).
- Junction GMIN DCOP convergence method
- Proprietary enhancements to transient analysis algorithm.
- Improvements to device models to remove discontinuities.

With these improvements, convergence failure with Pulsonix Spice is extremely rare. However, it is impossible to eliminate this problem altogether and there still remain some circuits which fail.

[1] CircuitSim90 Benchmarks

In this chapter we explain some of the causes of non-convergence and some of the strategies Pulsonix Spice uses to prevent it. Also explained is what to do in the rare event that convergence fails.

DC Operating Point - Overview

Pulsonix Spice has four different algorithms at its disposal to solve the DC operating point. For this analysis mode to fail, and assuming the default settings are being used, all four algorithms must fail.

The following sections describe the possible reasons for failure of each mode and what can be done about them.

The general procedure is as follows:

1. Check your circuit. Check that all components are the correct way around and have the correct values. Make sure you haven't used 'M' when you meant 'Meg'.
2. Refer to section *DC Operating Point – Source and GMIN Stepping* and see if GMIN or source stepping can be made to work.
3. Refer to section *DC Operating Point – Pseudo Transient Analysis* to get pseudo transient analysis converging.
4. Contact technical support. We don't officially offer a "convergence fixing" service and reserve the right to decline help. However, we are always interested in non-converging circuits and usually we will look at your circuit to see if we can identify the problem.

DC Operating Point - Source and GMIN stepping

By default, if these modes fail, Pulsonix Spice will carry on and attempt pseudo transient analysis. It will not do so only if instructed not to using the dcop Sequence option Pseudo transient analysis usually succeeds but sometimes can take a long time so you may prefer to get one of these methods working instead. Also, if pseudo transient analysis fails it is desirable to first see if GMIN or source stepping can be made to work.

There are a few options you can set to encourage these modes to converge. These are:

Name	Default	Suggest set to:	What it does
gminStepIterLimit	20	100	The number of iterations attempted for each GMIN step
gminMaxIters	1000	0 (equiv. to infinity)	Total number of iterations allowed for GMIN stepping
sourceMaxIters	1000	0 (equiv. to infinity)	Total number of iterations allowed for source stepping

It is only worth changing gminMaxIters or sourceMaxIters if the iteration limit is actually being reached. Often GMIN and source stepping fail to converge before the iteration limit is reached. To find out, select the command shell menu "Simulator|Show

Statistics". This displays, amongst other things, the number of iterations used for GMIN and/or source stepping. If they exceed 1000 then the iteration limit has been reached. This means that GMIN/source stepping may have succeeded if it had been given a chance.

DC Operating Point - Pseudo Transient Analysis

Pseudo transient analysis is the most powerful method known and it is rare for it to fail. It is not however infallible and can go wrong for the following reasons:

1. The transient analysis itself failed to converge, (this is rare)
2. The circuit oscillates.

Convergence failure in pseudo transient analysis

You will get the error message

```
Cannot find DC operating point
No convergence in pseudo transient analysis
```

The reasons why this may happen are the same as for transient analysis and are covered in *Fixes for Transient non-Convergence.*

Circuit oscillation

You will see the message

```
Cannot find DC operating point
Iteration limit exceeded in pseudo transient analysis
```

The circuit can oscillate because:

1. It is designed to i.e. it is or has an oscillator
2. It is supposed to be stable but passes an unstable region during supply ramping
3. It is supposed to be stable but has a fault in its design
4. It is stable but is made unstable by the capacitors added during the pseudo transient analysis.

If the circuit is an oscillator

If 1. then you must disable the oscillator during the DC solution. You can do this by one of the following methods:

1. Apply an initial condition to a point on the circuit that will break the oscillator's feedback loop.
2. Use the capacitor/inductor PTAVAL parameter to change its value during pseudo transient analysis. This parameter can be applied to a component or components that form part of the oscillator. In the netlist the parameter is applied at the end of the component line, e.g. for a capacitor:

   C12 N2 N6 1.2n PTAVAL=1
   In the above a 1.2n capacitor will take the value of 1 farad during pseudo transient analysis.

The circuit is not supposed to be an oscillator

If the circuit does not have any intentionally unstable elements then diagnosis of the problem is a little harder. Firstly, you need to rule out 4. above as a possible cause. Pulsonix Spice adds its own capacitors to your circuit during pseudo transient analysis in order to overcome potential problems with regenerative action. The problem is that these added capacitors can themselves make a circuit unstable. So the first thing to try is to inhibit the addition of these capacitors. To do this, add the following line to the netlist (See section "Adding Extra Netlist Lines" to find out how to add to a schematic).

```
.options ptaConfig=1
```

then re-run the simulation.

The circuit is not supposed to be an oscillator but it is

If this fails, then life gets even more complicated! If it fails with the message "`Iteration limit exceeded in pseudo transient analysis`" then it is very likely that the circuit is oscillating or entering an unstable region. If a different message is displayed go to the next section headed "The circuit doesn't oscillate but still doesn't converge". To allow diagnosis of what is happening Pulsonix Spice provides a method of analysing the circuit during the pseudo transient ramp. By default, no data is output during pseudo transient analysis but this can be changed as follows:

1. Set the analysis mode to DC operating point only.
2. Add the simulator option "ptaOutputVecs" by adding the following line to the netlist:

```
.options ptaOutputVecs
```

3. Now run the simulation for a while or until it stops.

You can now probe the circuit in the normal way to see what is oscillating. Once the oscillation has been fixed, you should be able to simulate the circuit successfully.

The circuit doesn't oscillate but still doesn't converge

As there are no added capacitors, there is a risk that pseudo transient analysis can fail for the same reason that GMIN and source stepping sometimes fail. In this case you will get the message "No convergence in pseudo transient analysis". If this happens your only recourse is the final desperation measure. This is to repeat the simulation with all valid values of ptaConfig from 2 to 15. (You can skip 7 as this is the default). ptaConfig is a simulator option that controls some of the parameters used in pseudo transient analysis. Most circuits pass for all settings but a few are more selective.

Accept Pseudo Transient Unconditionally

You can specify pseudo transient analysis to be accept unconditionally at some time after it has started. This is often a good solution to problems caused by circuit oscillation especially if the oscillation is small and unintended. To accept pseudo transient unconditionally, set the option:

.OPTIONS PTAACCEPTAT=*time*

Specify a *time* value that is adequate for the circuit state to settle as much as possible.

DC Operating Point - Junction Initialised Iteration

By default, this is the first method to be tried. If it fails, Pulsonix Spice will then attempt source stepping, GMIN stepping and finally pseudo transient analysis. Usually one of these other methods will succeed and it is not worth spending time getting this method to work.

If it does work, however, this is usually the fastest method and this can be put to good use for repetitive runs e.g. Monte-Carlo. It can be made to succeed using nodesets (see next section) and with a wisely chosen selection it is possible to speed up repetitive runs. Assuming one of the other methods does complete to a solution, the best way of creating nodesets is by using the SaveRHS command. This is explained in the next section.

DC Operating Point - Using Nodeset's

Nodesets have two uses, one to aid convergence and the other to bias the solution in multi-stable circuits.

Initially nodesets work exactly the same way as initial conditions. The nodeset voltage is applied via a 1 Ohm (by default) resistor and the solution is completed to convergence (by any of the methods). The nodeset is then released and the solution repeated. If the nodeset voltage is close to the actual solution the convergence of the second solution should be rapid.

With Pulsonix Spice, it is rarely necessary to use nodesets to find the DC solution of a circuit. They can, however, be useful for speeding up the operating point analysis for circuit that have already been solved. You may wish to do this for a Monte-Carlo analysis, for example.

Pulsonix Spice provides a means of creating nodesets using the SaveRHS command. To make use of this, proceed as follows:

1. Run a DC operating point analysis
2. Save the solution to a file using the SaveRhs command as follows:

```
SaveRhs /nodeset RHS.TXT
```

This will save to the file RHS.TXT a .nodeset control specifying the solution at each node.

3. Paste the contents of RHS.TXT to the netlist. Alternatively, include the file using the .INC control.

If you now repeat the DC analysis, you should now find that the solution is very rapid. Depending on the nature of your circuit, you may also find that the solution is found easily even if you modify the circuit. This is not, however, guaranteed.

Transient Analysis

What causes non-convergence

Fundamentally there are four reasons for convergence failure in transient analysis.

1. There is no solution to the circuit within the numerical range of the computer (approx. +/- 10^{308}).
2. One or more device models contains a discontinuity or (less of a problem) a discontinuity in its first derivative.
3. The circuit has a discontinuity caused by undamped regenerative action.
4. The solution matrix is ill-conditioned and the machine does not have sufficient accuracy to solve it to the required tolerance.

1. and 3. above are circuit problems. A trivial example of 1. is a PN junction biased by a large voltage. Without any series resistance, the voltage does not need to be very high for the current in the device to exceed the range of the machine. An example of 3. is a bistable circuit where the device capacitances are not modelled. The action of switching state would theoretically occur in zero time, a situation the simulator cannot be guaranteed to handle. Note also that negative valued components can cause 1. or 3. to occur.

2. is usually a software problem that the user can do little about. However, we are not aware of any discontinuities in the standard devices and have removed the ones we have found in the original SPICE3 code. It is possible to create a device with the arbitrary source that contains discontinuities. If your circuit has any of these devices in it you should check the equations for discontinuous behaviour. In particular the functions SGN() and U() are discontinuous and should be avoided.

4. is probably the most common cause of convergence failure in most other SPICE products but is rare in Pulsonix Spice. This is because we have done an extensive amount of work to eradicate the problem.

Nevertheless we still occasionally see circuits that fail. Usually they have one, or more likely, a combination of the following:

- Very small resistors especially if they are not connected to ground.
- Very large capacitors especially if they are not connected to ground.
- Very large inductors especially if they are not connected to ground.
- Circuit forced to use very small time steps perhaps because of fast rise/fall times.
- Very large currents/voltages
- Very high gain loops.

By "very" in the above we mean extreme. 1000V is not a very large voltage but 1000MV volts is. 1mΩ is not particularly small but 1pΩ is. If your circuit has any extreme values, try and moderate them. Don't use non-physical values of components if you can avoid it.

Fixes for Transient Non-convergence

- As with DC operating point, check your circuit. In particular, check that you are not doing anything which might cause numerical difficulties such as forward biasing a zero resistance pn junction with a large zero source impedance voltage source.
- Do anything that will prevent small time steps being needed. Gross non-linearities, regenerative loops and high gain loops all require small time-steps if not well damped. It may be that you have left out damping components to simplify the circuit and speed the simulation. It is worth noting that an accurately modelled circuit frequently converges more easily than an "idealised" circuit.
- Avoid using unrealistically large capacitors or inductors and unrealistically small resistors if at all possible. You should especially avoid such components if non-grounded.
- Try setting the PIVREL option to 0.999.
- If all else fails you can try relaxing some of the tolerances. If your circuit does not have any small (sub-μA) currents then set ABSTOL to 1e-9 or 1e-6. You can also increase VNTOL (default 1e-6) to say 1e-3 if your circuit only has large voltages. Increasing RELTOL is the very last thing you should try. In our experience, increasing RELTOL beyond its default value (0.001) is rarely a reliable solution and can make matters worse.
- Contact technical support. We don't officially offer a "convergence fixing" service and reserve the right to decline help. However, we are always interested in non-converging circuits and usually we will look at your circuit to see if we can identify the problem.

DC Sweep

DC sweep is basically a repeated DC operating point and so the issues relating to that mode also apply to DC sweep. However, if you are sweeping a voltage or current source, then an altogether better way of dealing with DC sweep problems is to simulate the DC sweep using transient analysis with a slow ramp.

Using transient analysis to perform DC sweep also resolves problems that can occur with circuits that have regions where there is more than one stable state e.g. bistable or Schmitt triggers. Consider sweeping the input voltage of a Schmitt trigger circuit. When the input voltage is between the lower and upper thresholds, the circuit has two stable states and the DC algorithm could find either of them. As each step in a DC analysis is initialised with the previous step, it will usually find the "correct" solution *but this is not guaranteed*. This means that the output could change state even though the input has not passed either threshold. This problem doesn't occur in transient analysis as in this mode the circuit is running as it would in real life.

DC Operating Point Algorithms

Pulsonix Spice uses five alternative strategies to resolve the DC operating point. These are:

1. Junction initialised iteration. This is our name for the standard algorithm sometimes simply known as "DC Iteration".
2. Source stepping.
3. Diag GMIN stepping.
4. GMIN stepping
5. Junction GMIN stepping
6. Pseudo transient analysis.

These are described in the following sections.

Junction Initialised Iteration

This is the standard algorithm and is sometimes simply known as "DC iteration". Each semiconductor junction is initialised with a small voltage and iteration then proceeds until convergence (or otherwise). This method often succeeds and is usually the quickest. However, the starting point is only a bit better than an educated guess and can be so far removed from the real solution that it never has a chance of succeeding. ("Junction initialised iteration" is a name we have coined and you may see it referred to as "JI2" elsewhere in this manual and also in messages output by Pulsonix Spice)

Source Stepping

Source stepping. This method - as with all the remaining methods to be described - belong to a class of convergence strategies known as "continuation methods". These all work by repeating the iterative process while gradually varying some circuit parameter. The circuit parameter is chosen so that at its start value the solution is known or trivial and at its final value the solution is the operating point that is required. In source stepping, all the circuit's power sources are gradually ramped up from zero to their final value. While at zero, the circuit's solution is trivial; all the voltages and currents are zero. At the first step, the supplies might be ramped up to 10% of their maximum and the solution iterates to convergence. Then the supplies are increased and the process is repeated. At each step the solution is initialised with the previous solution which, if the steps are small, will be close to the new solution that is required and convergence will therefore be relative easy to achieve.

This method is quite effective and is included in all SPICE based simulators including those derived from SPICE2. However the SPICE versions use a fixed step size, whereas in Pulsonix Spice, the step size is variable so if a step fails, the step size is reduced and it tries again.

However, even with an arbitrarily small step size, this method can fail if the circuit contains some kind of regenerative action. As the supplies are ramped it is possible for the circuit to abruptly switch from one state to another as in a Schmitt trigger. Although circuits such as Schmitt triggers do give difficulty, even circuits that do not have such elements can also give trouble.

Diagonal GMIN stepping

In this method, a large conductance term is added to every diagonal entry of the solution matrix and gradually reduced. This is similar to placing a low value resistor from every node of the circuit to ground but is by no means equivalent. The high conductance term (=low resistance) in the matrix effectively swamps non-linearities and as a result the solution is easy to find. The term is gradually reduced until it is zero.

This method is also effective and sometimes works for circuits for which source stepping fails. It is included with all SPICE3 derived simulators but, as with source stepping, the SPICE variants use a fixed step while Pulsonix Spice uses a variable step.

GMIN stepping suffers from the same problems as source stepping but not always with the same circuits so it always worth trying both approaches.

The received wisdom has always been that GMIN stepping is more effective than source stepping. This has not however been borne out by our own research which has shown the source stepping converges more often and more quickly. For this reason, Pulsonix Spice attempts source stepping before GMIN stepping. This is the reverse of SPICE3 and its derivatives.

Junction GMIN Stepping

The junction GMIN stepping method incrementally steps the conductance across semiconductor junctions. This in effect sweeps the GMIN option parameter.

This method is effective for CMOS IC designs as long as GMIN is implemented as a conductance between drain and source. This is not the default configuration for LEVEL 1 to 3 MOSFETs in which GMIN is implemented as two conductance between the drain and bulk and source and bulk. For other MOSFET models such as BSIM3 and EKV, the default GMIN is now between source and drain. For designs containing these devices, Junction GMIN Stepping is the first method attempted after JI2. For circuits that do not contain such devices, this method is not attempted at all.

Pseudo Transient Analysis

This method finds the solution using transient analysis. In Pulsonix Spice, a transient analysis is conducted while ramping up all power sources, in effect simulating the action of switching on the power supplies. This is not the same as source stepping as the latter is a pure DC method with all reactive components set to zero. Because reactive components - i.e. capacitors and inductors - are included in transient analysis, effects such as abrupt changes are damped and occur gradually over a finite time. This eliminates the problem - described above - that the DC continuation methods suffer from.

The above assumes, however, that the circuit is well modelled with all reactive elements correctly specified. With integrated circuit design this is usually the case, but for discrete circuits frequently is not. Opamp macro models, for example, consist of many idealised elements that are not always damped by reactive elements. Without such damping, pseudo transient analysis can fail for the same reason as source and GMIN stepping. So, Pulsonix Spice automatically adds additional capacitance to the circuit to prevent this situation from arising.

The end result is a convergence strategy that *nearly always succeeds*. However, it is generally the slowest method so in Pulsonix Spice it is, by default, attempted last.

Although pseudo transient analysis is very powerful it is not completely infallible. Its *Achilles Heel* is oscillation. Because a transient analysis is being performed it is possible for the circuit to oscillate. If this happens, pseudo transient analysis can end up going on forever without ever finding a stable solution. In our experience, however, this is actually rare. A number of steps are taken to damp oscillators so that even circuits that are designed to oscillate still succeed with pseudo transient analysis.

Pulsonix Spice provides a number of facilities to inhibit circuit oscillation during pseudo transient analysis. These are described in *DC Operating Point – Pseudo Transient Analysis*.

Controlling DC Method Sequence

You may have a circuit that only succeeds with - say - pseudo transient analysis and so attempting the other methods just wastes time. In this situation, you can force the simulator to attempt this method first, or even exclusively. To do this you need to set the two simulator options noOpiter and dcopSequence. noOpiter inhibits the first method (junction initialised iteration) while dcopSequence controls which and what order the remaining methods are attempted. The value of dcopSequence consists of any combination of "source", "gmin" and "pta" separated by the pipe symbol: '|'. "source", "gmin" and "pta" refer respectively to "source stepping", "GMIN stepping" and "pseudo transient analysis". The order in which these words appear in the value of dcopSequence, determines the order in which the corresponding methods will be attempted. So for example:

```
.options noOpiter dcopSequence=pta|gmin
```

will force pseudo transient analysis to be attempted first followed by GMIN stepping. Junction initialised iteration and source stepping won't be attempted at all.

Singular Matrix Errors

A singular matrix error occurs when the circuit does not have a unique and finite solution. For example, a circuit containing a floating capacitor does not have a unique DC solution as the capacitor can be at any voltage. Also circuits with shorted inductors, voltage sources or a combination of both will fail with this error.

If you get this error, you must first check your circuit. The simulator will tell you where the problem is either as a node name or a device name.

If you think you circuit is OK then it is possible that the error is occurring because during the course of iterating to a solution, some node voltages or device currents reached very high values and the limited accuracy of the machine made it seem that the matrix was singular. This can happen with junction initialised iteration. If this is the case, try setting the option:

```
.options noOpiter
```

This will inhibit this mode and the simulator will start with "source stepping". This method, and the others that follow, don't generally suffer from this problem.

Note that the simulation will abort if a singular matrix is detected in junction initialised iteration. It will not automatically attempt the other methods. This is because, by far the most common reason for singular matrices is circuit error.

Transient Analysis - "Time step too small" Error

The message:

```
Timestep too small
```

is not actually due to non-convergence. It means that, because of the nature of your circuit, to achieve the required accuracy, a time step smaller than the minimum permissible was needed. This can happen if you perform a very long transient analysis on a circuit with relatively short time constants. If you get this message, you can try reducing the minimum time step with the MinTimeStep simulator option. The default value for MinTimeStep is 1e-9*max time step and the max time step defaults to (Tstop-Tstart)/50 where Tstop and Tstart are respectively the stop and start times of the transient analysis.

Accuracy and Integration Methods

A Simple Approach

The accuracy of the simulation can be a complicated subject. So we will start with simple advice. If you wish to increase the accuracy of a simulation, reduce the value of the relative tolerance RELTOL. This defaults to 0.001 so to reduce it to say 1e-5 add the following line to the netlist:

```
.OPTIONS RELTOL=1e-5
```

(The setting of RELTOL is supported by the front end. See Simulation Parameters for details.)

The simulation will run slower. It might be a lot slower it might be only slightly slower. In very unfortunate circumstances it might not simulate at all and fail with a convergence error.

Conversely, you can speed up the simulation by increasing RELTOL, but we don't recommend it. Increasing RELTOL beyond its default value often degrades accuracy to an unacceptable level.

To increase speed with a reasonably controlled loss of precision, increase POINTTOL to 0.1 or even 1.0 but no higher.

Iteration Accuracy

For DC and transient modes, the simulator essentially makes an approximation to the true answer. For DC analysis an iterative method is used to solve the non-linear equations which can only find the exact answer if the circuit is linear. The accuracy of the result for non-linear circuits is determined by the number of iterations; accuracy is improved by performing more iterations but obviously this takes longer. In order to control the number of iterations that are performed an estimate is made of the error by comparing two successive iterations. When this error falls below a predetermined tolerance, the iteration is deemed to have converged and the simulator moves on the next step or completes the run. Most SPICE simulators use something similar to the

following equations to calculate the tolerance:

For voltages:

TOL = RELTOL * instantaneous_value + VNTOL

For currents:

TOL = RELTOL * instantaneous_value + ABSTOL

"instantaneous_value" is the larger of the current and previous iterations. VNTOL has a default value of 1µV so for voltages above 1mV, RELTOL dominates. ABSTOL has a default of 1pA so for currents above 1nA, RELTOL dominates. The above method of calculating tolerance works fine for many circuits using the default values of VNTOL and ABSTOL. However, SPICE was originally designed for integrated circuit design where voltages and currents are always small, so the default values of ABSTOL and VNTOL may not be appropriate for - say - a 100V 20A power supply. Suppose, that such a PSU has a current that rises to 20A at some point in the simulation, but falls away to zero. When at 20A it has a tolerance of 20mA but when it falls to zero the tolerance drops to ABSTOL which is 1pA. In most situations the 1pA tolerance would be absurdly tight and would slow down the simulation. Most other SPICE products recommend increasing ABSTOL and VNTOL for PSU circuits and indeed this is perfectly sound advice. However, In Pulsonix Spice the tolerance equation has been modified to make this unnecessary in most cases. Here is the modified equation:

For voltages:

TOL = RELTOL * MAX(peak_value * POINTTOL, instantaneous_value) + VNTOL

For currents:

TOL = RELTOL * MAX(peak_value * POINTTOL, instantaneous_value) + ABSTOL

peak_value is the magnitude of the largest voltage or current encountered so far for the signal under test. POINTTOL is a new tolerance parameter and has a default value of 0.001. So for the example we gave above, peak_value would be 20 and when instantaneous_value falls to zero the tolerance would be:

0.001 * MAX(20 * 0.001, 0) + 1p = approx. 20µA

20µA is a much more reasonable tolerance for a signal that reaches 20A. The above method has the advantage that it loosens the tolerance only for signals that actually reach large values. Parts of a circuit that only see small voltages or currents - such as the error amplifier of a servo-controlled power supply - would still be simulated with appropriate precision.

POINTTOL can be increased to improve simulation speed. It is a more controlled method than increasing RELTOL. POINTTOL can be raised to 0.1 or even 1.0 but definitely no higher than 1.0.

Time Step Control

The tolerance options mentioned above also affect the time step control algorithm used in transient analysis.

In Pulsonix Spice, there are three mechanisms that control the time step, one of which has to be explicitly enabled. These are:

1. Iteration time step control

2. LTE time step control

3. Voltage delta limit

Item 3 above is inactive unless explicitly enabled using the MAXVDELTAREL option setting.

Iteration Time Step Control

Iteration control reduces the time step by a factor of 8 if convergence to the specified accuracy cannot be achieved after 10 iterations. (10 by default but can be changed with ITL4 option). If convergence is successful, the time step is doubled. As this mechanism is controlled by the success or otherwise of the iteration it is also affected by the same tolerance options described in the above section.

LTE Time Step Control

The theory behind this method is beyond the scope of this manual but essentially it controls the accuracy of the numerical integration method used to model reactive devices such as inductors and capacitors. These devices are governed by a differential equation. It is not possible in a non-linear circuit to solve these differential equations exactly so a numerical method is used and this - like the iterative methods used for non-linear devices - is approximate.

In the case of numerical integration, the accuracy is determined by the time step. The smaller the time step the greater the accuracy but also the longer the simulation time.

The accuracy to which capacitors are simulated is controlled by RELTOL, POINTTOL and two other options namely TRTOL and CHGTOL. The latter is a charge tolerance and has a similar effect to VNTOL and ABSTOL but instead represents the charge in the capacitor. It's default value is 1e-14 which, like ABSTOL and VNTOL is appropriate for integrated circuits but may be too low for PSU circuits with large capacitors. However, the peak detection mechanism controlled by POINTTOL described in the above section also works for the LTE time step control algorithm and it is therefore rarely necessary to alter CHGTOL.

TRTOL is a dimensionless value and has a default value of 7. It affects the overall accuracy of the numerical integration without affecting the precision of the iteration. So reducing TRTOL will increase the accuracy with which capacitors and inductors are simulated without affecting the accuracy of the iterative method used to simulate nonlinear elements. However, in order for the simulation of reactive devices to be accurate, the non-linear iteration must also be accurate. So, reducing TRTOL much below unity will result in a longer simulation time but no improvement in precision. Increasing TRTOL to a large value, however, may be appropriate in some circumstances where the accuracy to which reactive devices are simulated is not that important. This may be the case in a circuit where there is an interest in the steady state but not in how it was reached.

Inductors are controlled by the same tolerances except CHGTOL is replaced by FLUXTOL. This defaults to 1e-11.

The default LTE time step algorithm used in Pulsonix Spice is slightly different to that used by standard SPICE. The standard SPICE variant is also affected by ABSTOL and VNTOL. The Pulsonix Spice algorithm controls the time step more accurately and as a result offers better speed-accuracy performance.

Voltage Delta Limit

This places a limit on the amount of change allowed in a single timestep for each node. This limit is governed by the option setting MAXVDELTAREL and MAXVDELTAABS and is included to overcome a problem that can cause false clocking of flip-flops. The limit can be calculated from:

MAXVDELTAABS + MAXVDELTAREL*(node_voltage)

where node_voltage is the larger of the node voltage at current time step and the node voltage at the previous time step. The above is calculated for all voltage nodes. If the change in voltage exceeds this limit, the time step is cut back.

The above mechanism is not enabled if MAXVDELTAREL is zero or less and MAXVDELTAREL is zero by default.

Setting MAXVDELTAREL to a value of about 0.4 will usually fix problems of false clocking in flip-flops. However, this will slow down the simulation slightly and it is not recommended that this setting is used in circuits that do not contain flip-flops.

Accuracy of AC analyses

The small-signal analysis modes .AC, .TF and .NOISE do not use approximate methods and their accuracy is limited only by the precision of the processor's floating point unit. Of course the DC operating point that always precedes these analysis modes is subject to the limitations described above. Also, the device models used for nonlinear devices are also in themselves approximations. So these modes should not be seen as exact but they are not affected by any of the tolerance option settings.

Summary of Tolerance Options

RELTOL

Default = 0.001. This affects all DC and Transient simulation modes and specifies the relative accuracy. Reduce this value to improve precision at the expense of simulation speed. We do not recommend increasing this value except perhaps to run a quick test. In any case, you should never use a value larger than 0.01.

POINTTOL

Proprietary to Pulsonix Spice. Default = 0.001. Can increase to a maximum of 1.0 to improve speed with loss of precision. Reduce to 0 for maximum accuracy but note this may just slow down the simulation without really improving precision where it is needed.

ABSTOL

Default = 1pA. This is an absolute tolerance for currents and therefore has units of Amps. This basically affects the tolerance for very low values of current. Sometimes worth increasing to resolve convergence problems or improve speed for power circuits.

VNTOL

Default = 1µV. Same as ABSTOL but for voltages.

TRTOL

Default = 7. This is a relative value and affects how accurately charge storage elements are simulated. Reduce it to increase accuracy of reactive elements but there is no benefit reducing below about 1.0. In circuits where there is more interest in the steady state rather than how to get there, simulation speed can be improved by increasing this value.

CHGTOL

Default = 1e-14. Minimum tolerance of capacitor charge. Some convergence and speed improvement may be gained by increasing this for circuits with large capacitors. Generally recommended to leave it alone.

FLUXTOL

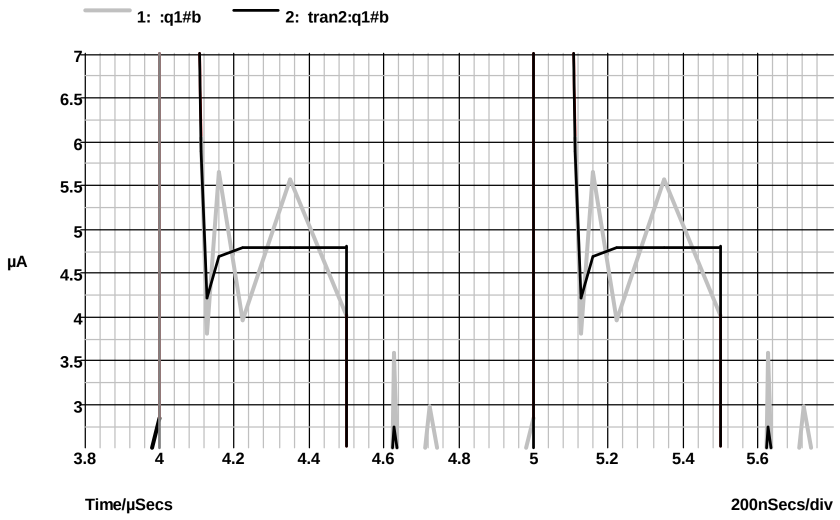
Default = 1e-11. Same as CHGTOL except applied to inductors.

Integration Methods - METHOD option

Pulsonix Spice, along with most other SPICE products use three different numerical integration methods to model reactive elements such as capacitors and inductors.

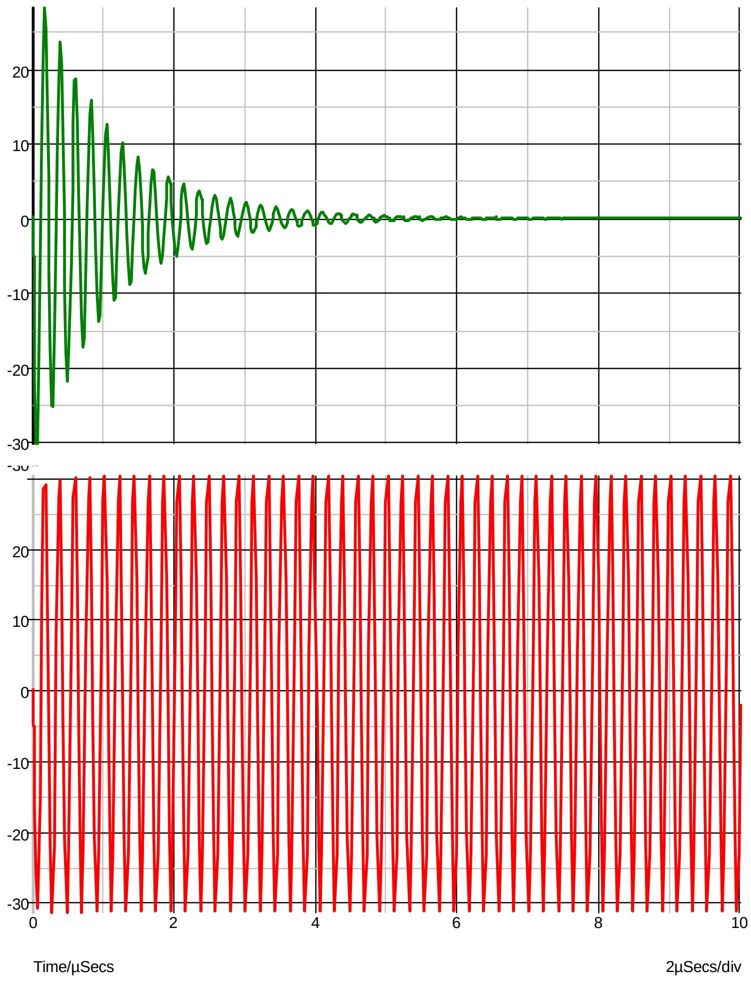
These are Backward Euler, Trapezoidal Rule and Gear. Backward Euler is used unconditionally at various times during the course of a simulation but at all other times the method used is controlled by the METHOD option (as long as ORDER is set to 2 or higher - see below).

The METHOD option can be set to TRAP (trapezoidal - the default) or GEAR. Gear integration can solve a common problem whereby the solution seems to oscillate slightly. An example is shown below.



The grey curve was simulated with the default trapezoidal integration method whereas the black used Gear integration. Note that gear integration introduces a slight overshoot.. This is a common characteristic. To find out whether such overshoots are a consequence of the integration or are in fact a real circuit characteristic, you should simulate the circuit with much smaller values of RELTOL (see above). It is also suggested that you switch back to trapezoid integration when using tight tolerances; the oscillating effect shown above will vanish if the tolerance is tight enough..

Note, you should not use Gear integration if you are simulating strongly resonant circuits such as oscillators. Gear integration introduces a numerical damping effect which will cause resonant circuits to decay more rapidly than they should. For example:



The above curves are the result of simulating a simple LC circuit that is completely undamped. The top trace was the result of Gear integration and the bottom, trapezoidal. The bottom curve is correct and agrees with theory. The top curve is inaccurate. If the analysis was done with Gear integration but with a smaller value of RELTOL, the damping effect would be less, so for all methods the result is ultimately accurate if the tolerance is made small enough. But trapezoidal gives accurate results without tight values of RELTOL.

ORDER option

This defaults to 2 and in general we recommend that it stays that way. Setting it to 1 will force Backward Euler to be used throughout which will degrade precision without any speed improvement. It can be increased up to a value of 6 if METHOD=GEAR but we have not found any circuits where this offers any improvement in either speed or precision.

Summary of Simulator Devices

The following information is needed to define schematic symbols for the various devices supported by the simulator.

In order to be able to plot component pin currents the pin names for the schematic symbol must match up with those used by the simulator. So for a BJT (bipolar junction transistor) the simulator refers to the four pins as "b", "c", "e" and "s" for base, collector, emitter and substrate. The same letters must also be used for the pin names for any schematic BJT symbol. The simulator device pin names are listed below.

The **Model** property is the schematic symbol property which describes what type of device the symbol refers to. SPICE uses the first letter of the component reference to identify the type of device. The Pulsonix Spice netlister prepends the model property (and a '$' symbol) to the component reference to comply with this. This makes it possible to use any component reference on the schematic.

Device	**Model property**	**Pin no.**	**Pin names**	**Pin function**
XSPICE device	A			See documentation for particular device in the XSPICE Device Overview section
Arbitrary Sources	B	1	p	
		2	n	
Bipolar junction transistors	Q	1	c	Collector
		2	b	Base
		3	e	Emitter
		4	s	Substrate
Capacitor	C	1	p	
		2	n	
Current Controlled Current Source (2 terminal)	F	1	p	
		2	n	
Current Controlled Current Source (4 terminal)	F	1	p	+ output
		2	n	- output
		3	any name	+ control current

		4	any name	- control current
Current Controlled Voltage Source (2 terminal)	H	1	p	
		2	n	
Current Controlled Voltage Source (4 terminal)	H	1	p	+ output
		2	n	- output
		3	any name	+ control current
		4	any name	- control current
Current Source	I	1	p	+
		2	n	-
Diode	D	1	p	Anode
		2	n	Cathode
GaAsFets	Z	1	d	Drain
		2	g	Gate
		3	s	Source
Inductor	L	1	p	
		2	n	
Junction FET	J	1	d	Drain
		2	g	Gate
		3	s	Source
MOSFET	M	1	d	Drain
		2	g	Gate
		3	s	Source
		4	b	Bulk (substrate)
Resistors	R	1	p	
		2	n	

Subcircuits	X	Pins can be given any name. Numbering must be in the order that pins appear in the .subckt control which defines the subcircuit. Pulsonix Spice uses a special extension of the netlist format to tell the simulator what the pin names are.		
Transmission Lines	T (lossless)	1	p1	Port 1 Terminal 1
	O (lossy)			
		2	n1	Port 1 Terminal 2
		3	p2	Port 2 Terminal 1
		4	n2	Port 2 Terminal 2
Voltage Controlled Current Source	G	1	p	+ output
		2	n	- output
		3	cp	Non-inverting control input
		4	cn	Inverting control input
Voltage Controlled Switch	S	1	p	Switch terminal 1
		2	n	Switch terminal 2
		3	cp	Non-inverting control input
		4	cn	Inverting control input
Voltage Controlled Voltage Source	E	1	p	+ output
		2	n	- output
		3	cp	Non-inverting control input
		4	cn	Inverting control input
Voltage Source	V	1	p	+ output
		2	n	- output

Monte Carlo Analysis

Overview

Monte Carlo analysis is a procedure to assess manufacturing yields by repeating simulation runs with varying applied random variations to component parameters. The technique is very powerful and usually gives a more realistic result than worst-case analysis which varies component values to their extremes in a manner which produces the worst possible result.

The implementation of Monte Carlo analysis in Pulsonix Spice has been designed to be quick to set up for simple cases while still providing the required flexibility for more advanced requirements as might be required for integrated circuit design.

Pulsonix Spice offers a level of flexibility for tolerance specification that cannot be found in other products including some high priced Spice applications. It is possible, for example, for different model parameters to be dependent on a single random variable. This makes it possible to model the fact that a number of model parameters might be dependent on a single physical characteristic, for example, the base width of a bipolar transistor. Of course, *lot* tolerances are also implemented accounting for the matching of devices in integrated circuits and other multiple components built onto a common substrate. However, in many products, lot tolerances can only be applied to the same type of device. In Pulsonix Spice it is possible to model parametric relationships between different types of device which occur in integrated circuits but which are rarely taken into account.

As well as conventional multiple step Monte Carlo analysis, single step Monte Carlo sweeps may also be performed. These are available for the four swept modes, .AC, .DC, .NOISE and .TF. For example, a Monte Carlo analysis of the DC offset voltage of an amplifier can be performed using a single run of .DC using a special sweep mode. This is dramatically faster than the alternative of repeated .OP runs. This type of analysis can also be used to analyse the gain of an amplifier at a single frequency using .AC or .TF or even the noise, again at a single frequency, using .NOISE.

Specifying a Monte Carlo Run

Monte Carlo runs are invoked in the same way as multi-step analyses (see “General Sweep Specification”). The basic syntax is:

.analysi_name analysi_parameters SWEEP MONTE *num_runs*

Where

.analysis_name	Dot control for analysis. Either .TRAN, .AC, .DC, .NOISE, .TF
analysis_parameters	Specific parameters for that analysis
num_runs	Number of runs

Examples

Run 10 Monte Carlo runs for 1mS transient analysis

```
.TRAN 1m SWEEP MONTE 10
```

100 Runs of a DC Sweep

```
    .DC V1 0 5 0.01 SWEEP MONTE 100
```

AC sweep of voltage source V5 from -300mV to 300mV. Repeat 50 times

```
    .AC DEVICE=V5 LIN 100 -300m 300m F=100000 SWEEP MONTE 50
```

Specifying a Single Step Monte Carlo Sweep

Monte Carlo sweep is one of the six modes available to the swept analysis modes, .AC, .DC .NOISE and .TF. The other modes are explained in “General Sweep Specification”. The general syntax is:

.analysis_name MONTE *num_points analysis_parameters*

Where:

.analysis_name	Dot control for analysis. Either .AC, .DC, .NOISE, .TF
analysis_parameters	Specific parameters for that analysis
num_points	Number of points in sweep

Examples

1000 point Monte Carlo sweep.

```
    .DC MONTE 1000
```

AC Monte Carlo sweep 100 steps. Frequency = 10K.
This is useful if - say - you are interested in the gain of an amplifier at one frequency and it needs to lie within a defined tolerance. Previously you would need to repeat an AC sweep at a single frequency to achieve this which could take a long time especially if the circuit has a difficult to find operating point. The analysis defined by the following line will take very little time even for a large circuit.

```
    .AC MONTE 100 F=10K
```

Log File

Unless explicitly disabled with the NOMCLOG option, a log file will always be generated for Monte Carlo analyses. It has the default name of MCLOG.TXT but this can be changed with the MCLOGFILE option. Here is an example of an actual output

```
Run 1: Seed=1226978751
Run 2: Seed=1521158126


Device        Nom.     Value        (Dev.)        Value         (Dev.)

Q10.D1:bv     5.9      5.9185638    (0.314641%)   5.8463766    (-0.90887%)
Q10.Q1:bf     220      283.10907    (28.68594%)   130.81497    (-40.5386%)
Q10.Q1:is     380a     368.7899a        (-2.95004%)   219.7988a     (-42.1582%)
Q11.D1:bv     5.9      5.9425623    (0.721395%)   5.8262401    (-1.25017%)
Q11.Q1:bf     220      285.27225    (29.66921%)   129.91303    (-40.9486%)
Q11.Q1:is     380a     354.1045a        (-6.8146% )   220.1177a     (-42.0743%)
Q12.D1:bv      5.9     5.8957932    (-713ppm )    5.787891     (-1.90015%)
Q12.Q1:bf     220      280.37304    (27.44229%)   130.28208    (-40.7809%)
```

```
Q12.Q1:is    380a   359.6985a   (-5.34249%)  225.8706a   (-40.5604%)
Q13.D1:bv    5.9    5.9020281   (343.74ppm)  5.8132485   (-1.47036%)
Q13.Q1:bf    220    280.04731   (27.29423%)  129.20488   (-41.2705%)
Q13.Q1:is    380a   367.7199a   (-3.2316% )  222.1358a   (-41.5432%)
Q14.D1:bv    5.9    5.9178142   (0.301936%)  5.8096709   (-1.531% )
Q14.Q1:bf    220    276.57192   (25.71451%)  129.93424   (-40.939% )
Q14.Q1:is    380a   364.6015a   (-4.05223%)  222.7107a   (-41.3919%)
Q4.D1:bv     5.9    5.9398543   (0.675496%)  5.8354342   (-1.09434%)
Q4.Q1:bf     220    277.08078   (25.94581%)  127.82878   (-41.896% )
Q4.Q1:is     380a   362.7751a   (-4.53287%)  225.9888a   (-40.5293%)
Q7.D1:bv     5.9    5.9281884   (0.47777% )  5.8421649   (-0.98026%)
Q7.Q1:bf     220    276.66227   (25.75558%)  129.29449   (-41.2298%)
Q7.Q1:is     380a   360.4184a   (-5.15304%)  227.0065a   (-40.2614%)
Q8.D1:bv     5.9    5.8811702   (-0.31915%)  5.8260238   (-1.25383%)
Q8.Q1:bf     220    280.33672   (27.42578%)  131.98533   (-40.0067%)
Q8.Q1:is     380a   361.0834a   (-4.97804%)  218.837a    (-42.4113%)
Q9.D1:bv     5.9    5.9001842   (31.215ppm)  5.8517296   (-0.81814%)
Q9.Q1:bf     220    281.41183   (27.91447%)  128.02565   (-41.8065%)
Q9.Q1:is     380a   358.8014a   (-5.57857%)  221.6128a   (-41.6809%)
```

The ‘Device’ column provides the name of the device and its model or instance parameter that is being reported. Q10.D1 is a diode ref D1 inside subcircuit Q1, BV is the model parameter.

The ‘Nom’ column displays the nominal value for that parameter.

Two columns are listed for each run. ‘Value’ is the actual value of the parameter and ‘(Dev.)’ is the deviation from the nominal.

The ‘Seed’ values displayed for each run at the top are the values used to seed the random number generator. These can be used to set the SEED option in order to repeat a particular random set. See below for more details.

Seeding the Random Number Generator

The random variations are created using a pseudo random number sequence. The sequence can be seeded such that it always produces the same sequence of numbers for a given seed. In Monte Carlo analysis, the random number generator is seeded with a new value at the start of each run and this seed value is displayed in the log file (see above). It is also possible to fix the first seed that is used using the SEED option. This makes it possible to repeat a run. To do this, note the seed value of the run of interest then add the line:

.OPTIONS SEED=*seed*_value

For example if you wanted to repeat run 2 in the above example you would add this line:

.OPTIONS SEED=1521158126

The first run of each Monte Carlo analysis will use the same random values as run 2 above. Note this assumes that only changes in values are made to the circuit. Any topology change will upset the sequence.

Specifying Tolerances

Overview

Tolerances for Monte Carlo analysis may be specified by one of the following methods:

1. Using a distribution function in an expression.

2. Using the device parameters TOL, MATCH and LOT

3. Using a tolerance model

1. above is new to release 2 and is the most general and flexible. 2 and 3 are provided primarily for backward compatibility but may also be more convenient in many circuits.

Distribution Functions

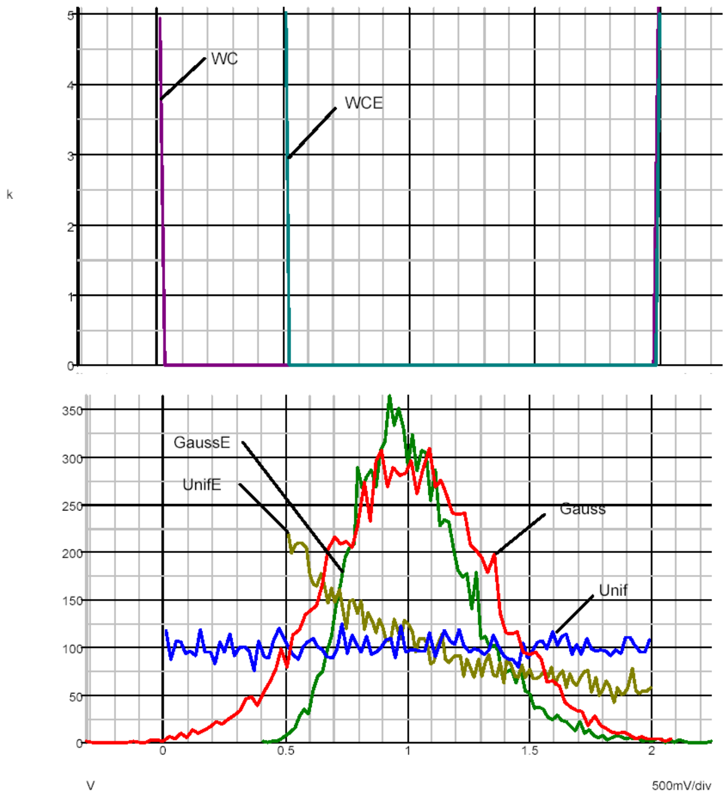
To specify Monte Carlo tolerance for a model or device parameter, define the

parameter using an expression (see “Using Expressions”) containing one of the following 12 functions:

Name	Distribution	Lot?
GAUSS	Gaussian (3-sigma)	No
GAUSSL	Gaussian (3-sigma)	Yes
UNIF	Uniform	No
UNIFL	Uniform	Yes
WC	Worst case	No
WCL	Worst case	Yes
GAUSSE	Gaussian logarithmic (3-sigma)	No
GAUSSEL	Gaussian logarithmic (3-sigma)	Yes
UNIFE	Uniform logarithmic	No
UNIFEL	Uniform logarithmic	Yes
WCE	Worst case logarithmic	No
WCEL	Worst case logarithmic	Yes

The logarithmic versions are included for compatibility with Version 1 and earlier but are nevertheless useful for some parameters which are logarithmic in nature such as the IS parameter for PN junctions.

The graphs below show the characteristics of the various distributions. The curves were plotted by performing an actual Monte Carlo run with 10000 steps.



Examples

Apply 50% tolerance to BF parameter of BJT with gaussian distribution.

```
    .MODEL NPN1 NPN IS=1.5e-15 BF={180*GAUSS(0.5)}
```

Lot Tolerances

The *lot* versions of the functions specify a distribution to be applied to devices whose tolerances track. These functions will return the same random value for all devices that reference the same model.

Alternatively, a device can be given a lot value as was required with earlier versions of Pulsonix Spice. Devices must have the same lot value, and also reference the same model, in order to track. This allows, for example, two or more chips using the same process to be simulated together without having to rename the models.

Examples

Specify 50% uniform lot tolerance and 5% gaussian device tolerance for BF parameter

.MODEL NPN1 NPN IS=1.5E-15 BF={180*GAUSS(0.05)*UNIFL(0.5)}

Here is an abbreviated log file for a run of a circuit using 2 devices referring to the above model:

```
                   Run 1                       Run 2

Device   Nom.      Value        (Dev.)         Value         (Dev.)
Q1:bf    180       93.308486    (-48.162% )    241.3287      (34.0715% )
Q2:bf    180       91.173893    (-49.3478%)    245.09026     (36.16126%)

                   Run 3                       Run 4

Device   Nom.      Value        (Dev.)         Value       (Dev.)
Q1:bf    180       185.95824    (3.310133%)    210.46439   (16.92466%)
Q2:bf    180       190.8509     (6.02828% )    207.04202   (15.02335%)
```

For the four runs BF varies from 91 to 245 but the two devices never deviate from each other by more than about 2.7%.

Notes

The tracking behaviour may not be as expected if the model definition resides within a subcircuit. When a model is defined in a subcircuit, a copy of that model is created for each device that calls the subcircuit. Here is an example:

```
XQ100 VCC INN Q100_E 0 NPN1
XQ101 VCC INP Q101_E 0 NPN1
.SUBCKT NPN1 1 2 3 SUB
Q1 1 2 3 SUB N1
Q2 SUB 1 2 SUB P1
.MODEL N1 NPN IS=1.5E-15 BF={180*GAUSS(0.05)*UNIFL(0.5)}
.ENDS
```

In the above, XQ100 and XQ101 *will not track*. Two devices referring to N1 *inside* the subcircuit definition would track each other but different instances of the subcircuit will not. To make XQ100 and XQ101 track, the definition of N1 should be placed outside the subcircuit. E.g.

```
XQ100 VCC INN Q100_E 0 NPN1
XQ101 VCC INP Q101_E 0 NPN1
.SUBCKT NPN1 1 2 3 SUB
Q1 1 2 3 SUB N1
Q2 SUB 1 2 SUB P1
.ENDS
.MODEL N1 NPN IS=1.5E-15 BF={180*GAUSS(0.05)*UNIFL(0.5)}
```

Arguments to Distribution Functions - the 'key' Value

Each of the distribution functions takes 1 or 2 arguments. The first argument is the tolerance while the second is an optional key value. The key is an arbitrary number - preferably an integer - which, in effect, names a random variable for which the results of that distribution function will be based. Another call to the same distribution function *in the same model* and with the same key value, will also be based on the same random variable and return the same value for each Monte Carlo step. The key make it possible to accommodate parameters that tend to track each other possibly because they depend on the same physical characteristic of the device.

Example

Suppose the BF and TF parameters of a BJJT tend to track each other. That is a 50% increase in BF tends to be accompanied by a 50% increase in TF (there is no physical basis for this; it's just an example). The following model definition would implement this:

```
.MODEL NPN1 NPN BF={UNIF(0.5,1)*180} TF={1e-11*UNIF(0.5,1)}
```

For all devices using that model, BF and TF will always have a fixed relationship to each other even though each parameter can vary by +/-50% from one device to the next.

Here is the log of a run carried out on a circuit with two of the above devices:

		Run 1		Run 2	
Device	Nom.	Value	(Dev.)	Value	(Dev.)
Q1:bf	180	226.52869	(25.84927%)	117.2733	(-34.8482%)
Q1:tf	10p	12.58493p	(25.84927%)	6.515184p	(-34.8482%)
Q2:bf	180	179.58993	(-0.22782%)	164.21785	(-8.76786%)
Q2:tf	10p	9.977218p	(-0.22782%)	9.123214p	(-8.76786%)

Notice that the BF and TF parameters always deviate by exactly the same amount for each device. However, the two devices do not track each other. If this were needed, the lot versions of the functions could be used instead. E.g.

```
.MODEL NPN1 NPN BF={UNIFL(0.5,1)*180} TF={1e-11*UNIFL(0.5,1)}
```

This is the log for such an example:

		Run 1		Run 2	
Device	Nom.	Value	(Dev.)	Value	(Dev.)
Q1:bf	180	104.57858	(-41.9008%)	93.855425	(-47.8581%)
Q1:tf	10p	5.809921p	(-41.9008%)	5.21419p	(-47.8581%)
Q2:bf	180	104.57858	(-41.9008%)	93.855425	(-47.8581%)
Q2:tf	10p	5.809921p	(-41.9008%)	5.21419p	(-47.8581%)

Distribution Functions and .PARAM

The key mechanism described above only works for parameters within the same model. If you wish to define a fixed relationship between parameters of different models then you can define a random variable using .PARAM.

Using a distribution function in a .PARAM expression in effect creates a global random variable. .PARAM expressions are only evaluated once for each Monte Carlo step so the parameter it defines will be the same value wherever it is used.

Note that .PARAM values used for this purpose should be defined at the top level i.e. not in a sub-circuit. If defined in a sub-circuit they will be local to that sub-circuit so each instance of the sub-circuit will use its own random variable.

TOL, MATCH and LOT Device Parameters

These parameters may be used as a simple method of applying tolerances to simple devices such as resistors. The TOL parameter specifies the tolerance of the device's value. E.g.

```
R1 1 2 1K TOL=0.05
```

The above resistor will have a tolerance of 5% with a gaussian distribution by default. This can be changed to a uniform distribution by setting including the line:

```
.OPTIONS MC_ABSOLUTE_RECT
```

in the netlist.

Multiple devices can be made to track by specifying a LOT parameter. Devices with the same LOT name will track. E.g.

```
R1 1 2 1K TOL=0.05 LOT=RES1
R2 3 4 1k TOL=0.05 LOT=RES1
```

R1 and R2 in the above will always have the same value. Deviation between tracking devices can be implemented using the MATCH parameter. E.g.

```
R1 1 2 1K TOL=0.05 LOT=RES1 MATCH=0.001
R2 3 4 1k TOL=0.05 LOT=RES1 MATCH=0.001
```

R1 and R2 will have a tolerance of 5% but will always match each other to 0.1%. MATCH tolerances are gaussian by default but can be changed to uniform by specifying

```
.OPTIONS MC_MATCH_RECT
```

Distributions are always the logarithmic versions as described in “Distribution Functions” earlier in this section.

If using device tolerance parameters, note that any absolute tolerance specified must be the same for all devices within the same lot. Any devices with the same lot name but different absolute tolerance will be treated as belonging to a different lot. For example if a circuit has four resistors all with lot name RN1 but two of them have an absolute tolerance of 1% and the other two have an absolute tolerance of 2%, the 1% devices won't be matched to the 2% devices. The 1% devices will however be matched to each other as will the 2% devices. This does not apply to match tolerances. It's perfectly OK to have devices with different match tolerances within the same lot.

Tolerance Models

Overview

Tolerance models are an alternative method of applying tolerances to device models to the distribution function method described in an earlier section. The distribution function method is in general more flexible and is recommended for most applications. However, the tolerance model method has some advantages as follows:

- It is compatible with earlier Pulsonix Spice versions from 1.0.
- It allows tolerances to be applied to devices without modifying the main model.

Definition

The format for a tolerance model:

.MODEL *modelname modeltype*.tol *parameter_list*

modelname must be the same name as the normal model for the device while *modeltype* must be the same type. So for example a tolerance model for a Q2N2222 transistor might be:

.MODEL Q2N2222 npn.tol BF=0.5

This will vary the BF parameter over a +/- 50% range for all BJTs referring to the Q2N2222 model. The above model only specifies one parameter but you can place any parameter specified for that device in a tolerance model. For MOSFETs the level number *must* be included with the tolerance model otherwise the model will be ignored.

Important note

Note that tolerances will only be applied to parameters explicitly specified in the base model for the device. Tolerances will not be applied to default values. If the base model for the Q2N2222 device in the above example is:

```
.MODEL Q2N2222 npn ( IS=2.48E-13 VAF=73.9 NE=1.2069
+ TF=4.00E-10)
```

The BF parameter in the tolerance model would *not* be used as it is not specified in the base model. If the base model was modified to:

```
.MODEL Q2N2222 npn ( IS=2.48E-13 VAF=73.9 NE=1.2069
+ TF=4.00E-10) BF=400
```

Then the BF tolerance would be applied.

Matching Devices Using Tolerance Models

To match devices with tolerances defined using a tolerance model, specify the LOT parameter on the device line. E.g.

```
Q1 1 2 3 0 Q2N2222 LOT=lot1
Q2 4 5 6 0 Q2N2222 LOT=lot1
.MODEL Q2N2222 npn.tol BF=0.5
```

In the above example the BF parameter for Q1 and Q2 will always be the same. To specify a deviation for matched devices requires a match tolerance model definition. This is of the form:

.MODEL *modelname modeltype*.match *parameter_list*

modelname must be the same name as the base model for the device while *modeltype* must be the same type. So for example a matching tolerance model for a Q2N2222 transistor might be:

```
.MODEL Q2N2222 npn.match BF=0.5
```

Note that the components will only be matched if they all refer to the same model. Any components with the same lot name but referring to a different model treated as if they belong to a different lot.

Schematic Tolerance Specification

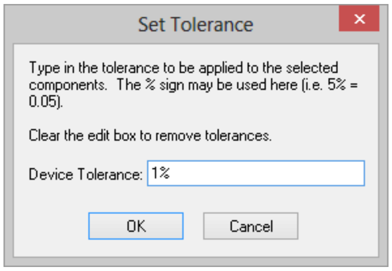
Component device tolerances can be specified at the schematic level.

Device tolerances are applied for each component and are specified as a <Spice Tolerance> Attribute on a Schematic Component using the **Set Tolerance** option, or as "tol=*tolerance*" appended to the device line in a netlist.

Setting Device Tolerances

To select individual device tolerances proceed as follows:

1. Select the component or components whose tolerances you wish to be the same. (You can individually select components by holding the Control key down and left clicking on each in turn).
2. Select the **Monte Carlo** option, **Set Tolerance,** and enter the tolerance in the dialog.



Type in the tolerance to be applied to the selected components. The % sign may be used here (i.e. 5% = 0.05).

Clear the edit box to remove tolerances.

Device Tolerance: 1%

3. You may use the '%' symbol here if you wish, so 5% and 0.05 have the same effect. (Note: this is the only place that '%' is recognised - you can't use it netlists or models).
4. If the all resistors or all capacitors in a circuit are to have the same tolerance, use either **Monte Carlo** option **Select All Capacitors** or **Select All Resistors** and then **Set Tolerance**.

   Device tolerances can be applied to the following components:

   Capacitors

   Resistors

   Inductors

   Fixed voltage sources

   Fixed current sources

   Voltage controlled voltage sources

   Voltage controlled current sources

   Current controlled voltage sources

   Current controlled current sources

   Lossless transmission lines (applied to Z0 parameter)

   Device tolerance will be ignored for other devices.

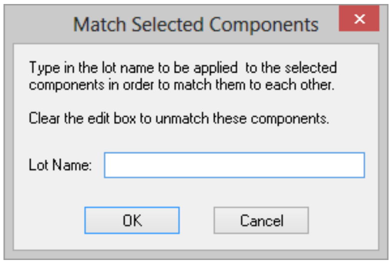
Matching Devices.

Some devices such as resistor networks are constructed in a manner that their tolerances track. Such devices often have two specifications one is an absolute tolerance and the other a matching tolerance. A thin film resistor network might have an absolute tolerance of 1% but a matching tolerance of 0.05%. This means that the resistors will vary over a +/-1% range but will always be within +/-0.05% of each other.

To specify matched devices for Monte Carlo analysis two pieces of information are required. Firstly, the components that are matched to each other must be identified and secondly their matching tolerances need to be specified.

To Identify Matched Devices

1. Select the components you wish to match to each other. (Use Control key to select multiple components.)
2. Select the **Monte Carlo** option **Match Components**. The following dialog will be displayed:



Match Selected Components

Type in the lot name to be applied to the selected components in order to match them to each other.

Clear the edit box to unmatch these components.

Lot Name:

3. You must now supply a ***lot*** name which must be unique. You can use any alphanumeric name.

Matching Tolerances

As for absolute tolerances, matching tolerances can be specified for a device or for a model. To specify device match tolerances, proceed as follows:

1. Select the components you wish to match to each other. (Use control key to select multiple components.)
2. Select the **Monte Carlo** option **Set Match Tolerances.**
3. Enter the desired tolerance.

If using device tolerance parameters, note that any absolute tolerance specified must be the same for all devices within the same lot. Any devices with the same lot name but different absolute tolerance will be treated as belonging to a different lot. For example if a circuit has four resistors all with lot name "RN1" but two of them have an absolute tolerance of 1% and the other two have an absolute tolerance of 2%, the 1%'s won't be matched to the 2%'s. The 1%'s will however be matched to each other as will the 2%'s. This does not apply to *match* tolerances. It's perfectly Okay to have devices with different match tolerances within the same lot.

The format for a matching tolerance models is defined previously in this section.

Random Distribution

The default distribution is Gaussian with the tolerance representing a 3σ spread. This can be changed to rectangular using two simulator options. These are:

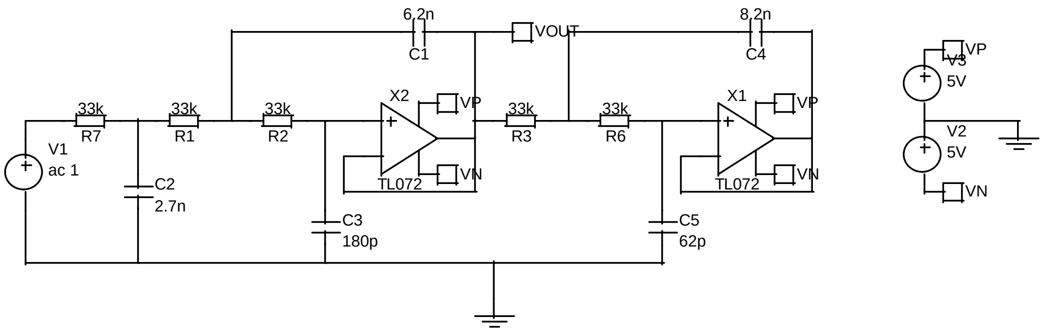
MC_ABSOLUTE_RECT	If set absolute tolerances will have a rectangular distribution

MC_MATCH_RECT	If set matching tolerances will have a rectangular distribution.

Distributions can be specified on a per component basis or even on a per parameter basis by using distribution functions in an expression as mentioned earlier in this section.

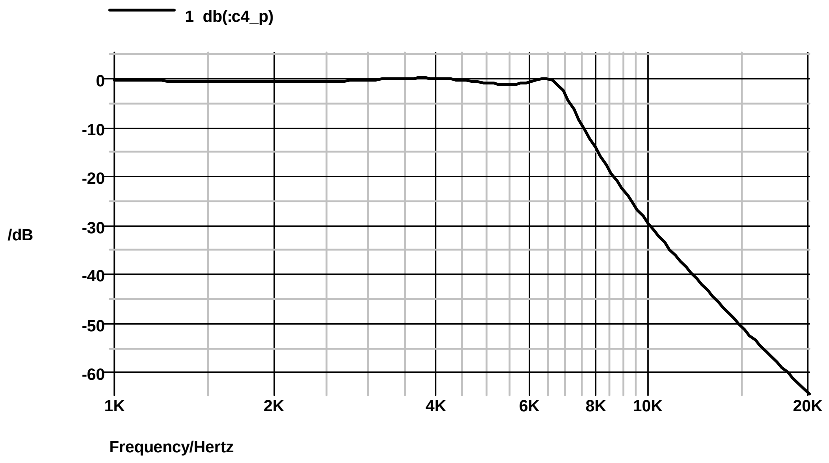
An Example

Consider the following active filter circuit.



This circuit can be found in Pulsonix\Examples\Spice\MonteCarlo\cheb.sch

The circuit is a 5th order low-pass 7kHz Chebyshev filter with a 1dB passband ripple specification. Its nominal response is:

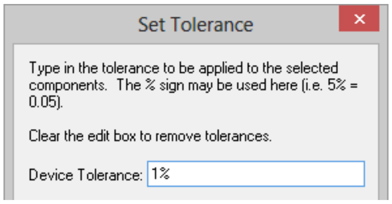


This circuit is to be used in an application that requires the gain of the amplifier to remain within 2dB of the dc value from 0 to 6kHz. A 1dB ripple specification therefore seems a reasonable choice. Clearly though the tolerance of the capacitors and resistors may upset this. To investigate, a Monte Carlo analysis is required. The standard component tolerances are 10% for capacitors and 1% for resistors. To apply these tolerances, proceed as follows.

Note, all of the Monte-Carlo options are to be found on the **Monte Carlo** sub-menu, which is on the **Simulation** menu.

1. Use **Select All Resistors** option.

2. Select **Set Tolerance** option to set the tolerance of the selected resistors. The following dialog is displayed:



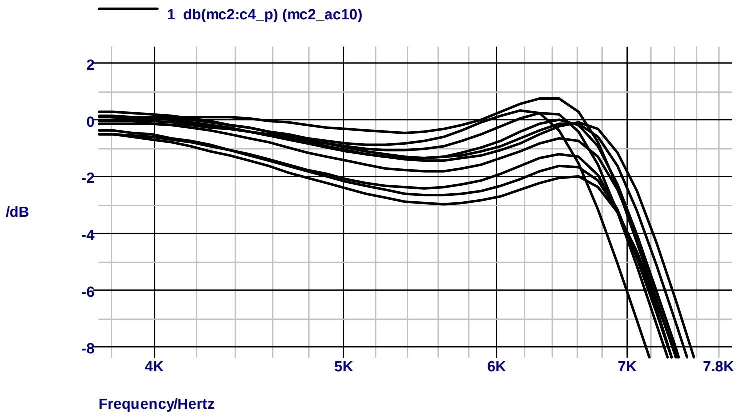
Type in the tolerance to be applied to the selected components. The % sign may be used here (i.e. 5% = 0.05).

Clear the edit box to remove tolerances.

Device Tolerance: 1%

3. Enter **1%.** (The % is recognised)
4. Now use **Select All Capacitors.**
5. Again use **Set Tolerance** to set all capacitor tolerances to 10%.
6. Use **Simulation Parameters** option. On AC page enable Multi-step.
7. Press the **Define Mode** button and set up Monte Carlo with 100 steps.
8. Use **Simulate Design** to perform the analysis.

The analysis will be repeated 10 times. Now plot the result in the usual way. (Use the **db Voltage** Random Probe). The result is the following:



As can be seen, the specification is not met for some runs. Note that - as a rule of thumb - the spread for 10 runs will be about half of a realistic maximum spread (3σ).

If you wish to use the cursors to make measurements off of the graph, first switch cursors on (graph popup menu). Next place each of the cursors where you wish along the x-axis. To move cursors between curves, use the TAB key for the main cursor and Ctrl-TAB for the reference cursor. (The popups **Step Main Cursor**, and **Step Reference Cursor**, do the same thing.)

The Pulsonix Spice Monte Carlo analysis implementation has many more features which are described in the following section. These are:

1. Random variation of device model parameters.
2. Support for matched devices.

3. Log file creation.
4. Seed selection to allow repeated runs with same randomly applied values.

Running Monte Carlo

Overview

There are actually two types of Monte Carlo analyses. These are:

1. Single step Monte Carlo sweep
2. Multi step Monte Carlo run

1. above is applicable to AC, DC, Noise and Transfer Function analyses. 2. can be applied to the same analyses in addition to transient analysis.

An example of 1. can be seen in the AC section of the Analysis Modes chapter. This was a run where the gain at a single frequency was calculated 1000 times with the Monte Carlo tolerances applied. This used AC analysis with the Monte Carlo sweep mode – one of the six modes available. Only a single curve is created hence the name *single step*

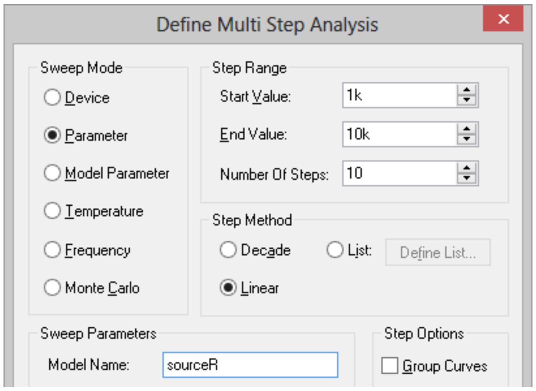
An example of 2 is the example at the beginning of this chapter. Here a complete frequency sweep from 1kHz to 100kHz was repeated 100 times creating 100 curves.

Setting up a Single Step Monte Carlo Sweep

1. Select schematic menu **Simulation | Simulation Parameters…** Select the **AC, DC, Noise** or **TF** tab as required.
2. In the Sweep Parameters section, press the **Define Mode**… button.
3. In the Sweep Mode section select **Monte Carlo**.
4. In the Parameters section enter the required value for the Number of points.
5. For AC, Noise and TF, you must also supply a value for Frequency.

Setting up a Multi Step Monte Carlo Run

1. Select schematic menu **Simulation | Simulation Parameters…** Select the **AC, DC, Noise, Transient** or **TF** tab as required.
2. Define the analysis as required.
3. In the **Monte Carlo** and **Multi-step Analysis** section, check the **Enable multi-step** box and press the **Define Multi-step** button. This will open:



Sweep Mode: [✓ Parameter]

Start Value: 1k

End Value: 10k

Number Of Steps: 10

Step Method: [✓ Linear]

Sweep Parameters

Model Name: sourceR

4. In the **Sweep mode** section, select **Monte Carlo**.
5. In the **Step Parameters** section, enter the number of steps required.

Running a Monte Carlo Analysis

Monte Carlo analyses are run in exactly the same way as other analyses. Press F9 or use Simulate Design from the simulation menu.

Setting the Seed Value

The random variations are created using a pseudo random number sequence. The sequence can be seeded such that it always produces the same sequence of numbers for a given seed. In Monte Carlo analysis, the random number generator is seeded with a new value at the start of each run and this seed value is displayed in the log file. It is also possible to fix the first seed that is used using the SEED option. This makes it possible to repeat a run. To do this, note the seed value of the run of interest from the log file then set the seed as follows:

1. Select the schematic menu **Simulation | Simulation Parameters**
2. Select **Options** tab and enter the seed value in the **Monte Carlo** Section

The first run of each Monte Carlo analysis will use the same random values as the run from which you obtained the seed value in the log file. Note this assumes that only changes in values are made to the circuit. Any topology change will upset the sequence.

This technique is a convenient way of investigating a particular run that perhaps produced unexpected results. Obtain the seed used for that run, then repeat with the seed value but doing just a single run. You will then be able to probe around the circuit and plot the results for just that run.

Analysing Monte-Carlo Results

Plots

Plots of Monte Carlo analyses are performed in exactly the same manner as for normal runs. When you probe a circuit point curves for each run in the MC analysis will be created. You will notice, however, that only one label for each *set* of curves will be displayed. Operations on curves such as deleting and moving will be performed on the complete set.

Identifying Curves

Sometimes it is useful to know exactly which run a particular curve is associated with. To do this proceed as follows:

1. Switch on graph cursors. (Cursors | Toggle On/Off menu)
2. Pick up the main cursor (the one with the short dashes) and place it on the curve of interest. (To pick up a cursor, place mouse cursor at intersection, press left key and drag).
3. Select **Show Curve Info** menu. Information about the curve will be displayed in the command shell.

This is an example of what will be displayed.

```
Source group: ac1
Curve id: 4
Run number: 49
```

The information of interest here is the *Run number*. With this you can look up in the log file details of the run i.e. what values were used for each component and parameter. You can also obtain the seed value used so that the run can be repeated. See “Setting the Seed Value” on the previous page.

Plotting a single Curve

If you wish to plot a single curve in a Monte Carlo set, you must obtain the run number then use the **Random Probe | Add Curve**… option to plot an indexed expression. We use an example to explain the process.

Using the Chebyshev filter example, let’s suppose that we wish to plot the curve of the filter output created by run 49 alone without the remaining curves. Proceed as follows.

1. Run the Chebyshev filter example as explained at the beginning of this chapter.
2. Select **Random Probe** option and use **Add Curve** from the shortcut menu.
3. Click on the output of the filter. You should see Yout entered in the Y expression box.
4. You must now modify the expression you have entered to give it an index value. For the simple case of a single voltage or current just append it with

   ```
   [index]
   ```

   where *index* is the run number less 1. in this example the run number is 49 so we enter 48 for the index. You should now have:

```
vout_P[48]
```

displayed in the Y expression box.

5. Close box. You should see a single curve plotted.

An alternative method of plotting single curves is given in “Setting the Seed Value”

Creating Histograms

See “Performance Analysis and Histograms” on in a previous section.

Index