Qucs

A Tutorial

DC Analysis, Parameter Sweep and Device Models

Stefan Jahn
Chris Pitcher

DC Static Circuits

A favourite question in electronics courses used to be:

> *You have twelve one ohm resistors; you connect them together so that each resistor lies along the edge of a cube. What is the resistance between opposite corners of the cube?*

The intention may have been to teach soldering, as more than one student solved it by making just such a cube! These days we can do that without touching the soldering iron; we simulate the circuit.

Here is my attempt to make a cube in Qucs; anyone is welcome to try and improve it.

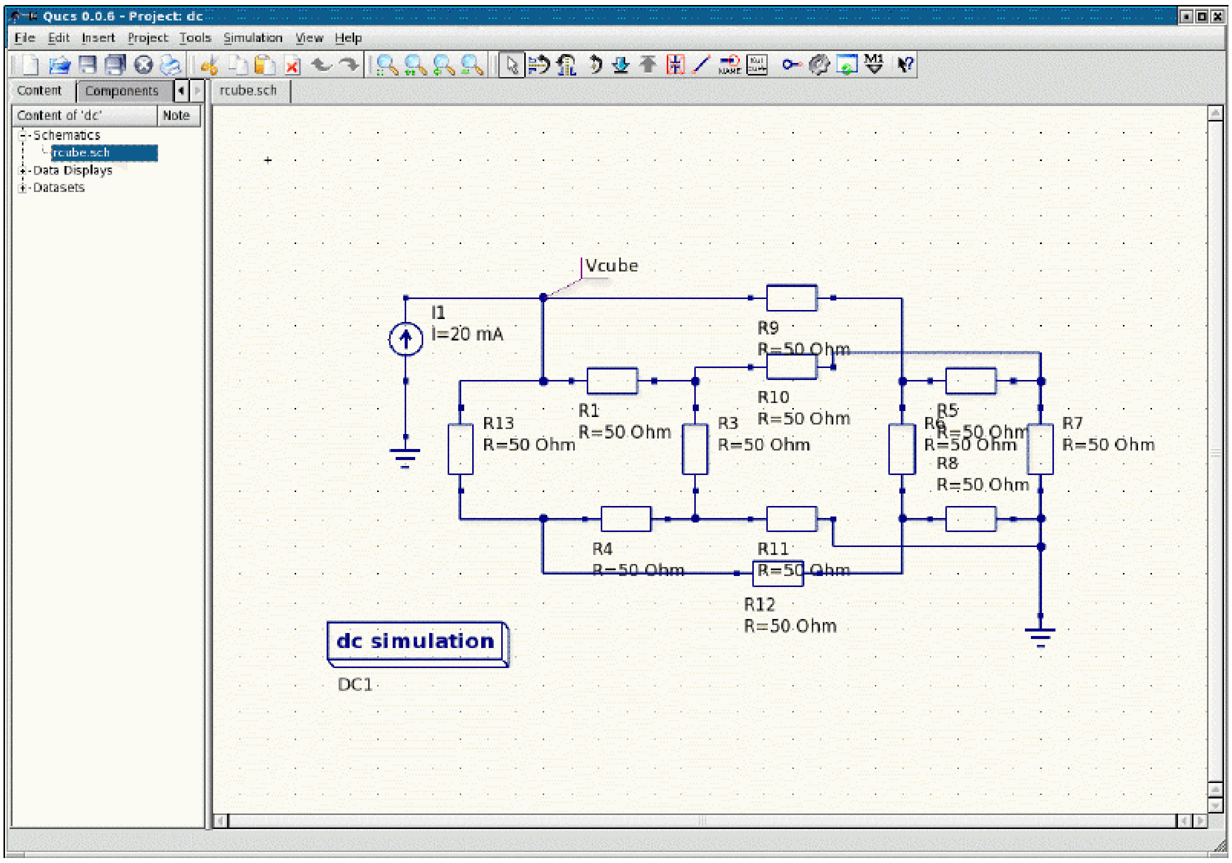


Figure 1: resistor cube schematic

All I did was select resistance in the left hand component window and paste them down, rotating as necessary, until I had twelve on the schematic. Then I wired two sets of four into squares, then connected the remaining four between the corners of the squares. Which I'm sure is topologically the same as a cube.

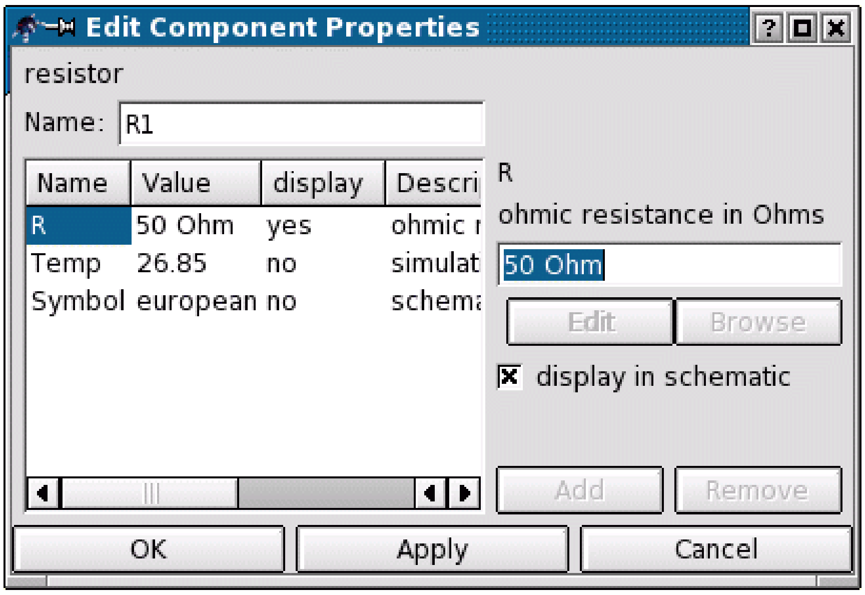
Which all might seem trivial, but is a good reminder right at the beginning that we are creating a virtual representation of a physical circuit. Sometimes we have to bend and squeeze things to get it into a format that our simulator will accept, which leaves us wondering whether we are working with an accurate representation.

The Rule is: if we can correlate the junctions of our components with those of the real circuit, we are accurately representing the physical circuit. And, I might add, it is ALWAYS worth checking that we have done it right; simulate the wrong circuit and it will tell you lies.

With my cube of resistors accurately drawn, I only have to hit the simulation button and the tabulated results will show me the voltage at the corner node. As I am forcing a constant current through the cube from one corner to another, Ohm's Law tells me that the voltage between those corners will give me the resistance. If I use a current of one amp, the output voltage will be equal to the resistance in ohms.[1]

Those with good attention to detail will be complaining about now that I haven't really solved the problem, as the question mentioned one ohm resistors while I have used fifty ohms. Well, yes, I cheated. Which I often do in simulations.

To set all the resistances to the correct value I would have had to open the Properties Editor window twelve times; here is how it looks...



resistor

Name: R1

Name	Value	display	Descri
R	50 Ohm	yes	ohmic r
Temp	26.85	no	simulat
Symbol	european	no	schema

R

ohmic resistance in Ohms

50 Ohm

[✓] display in schematic

Figure 2: component property dialog

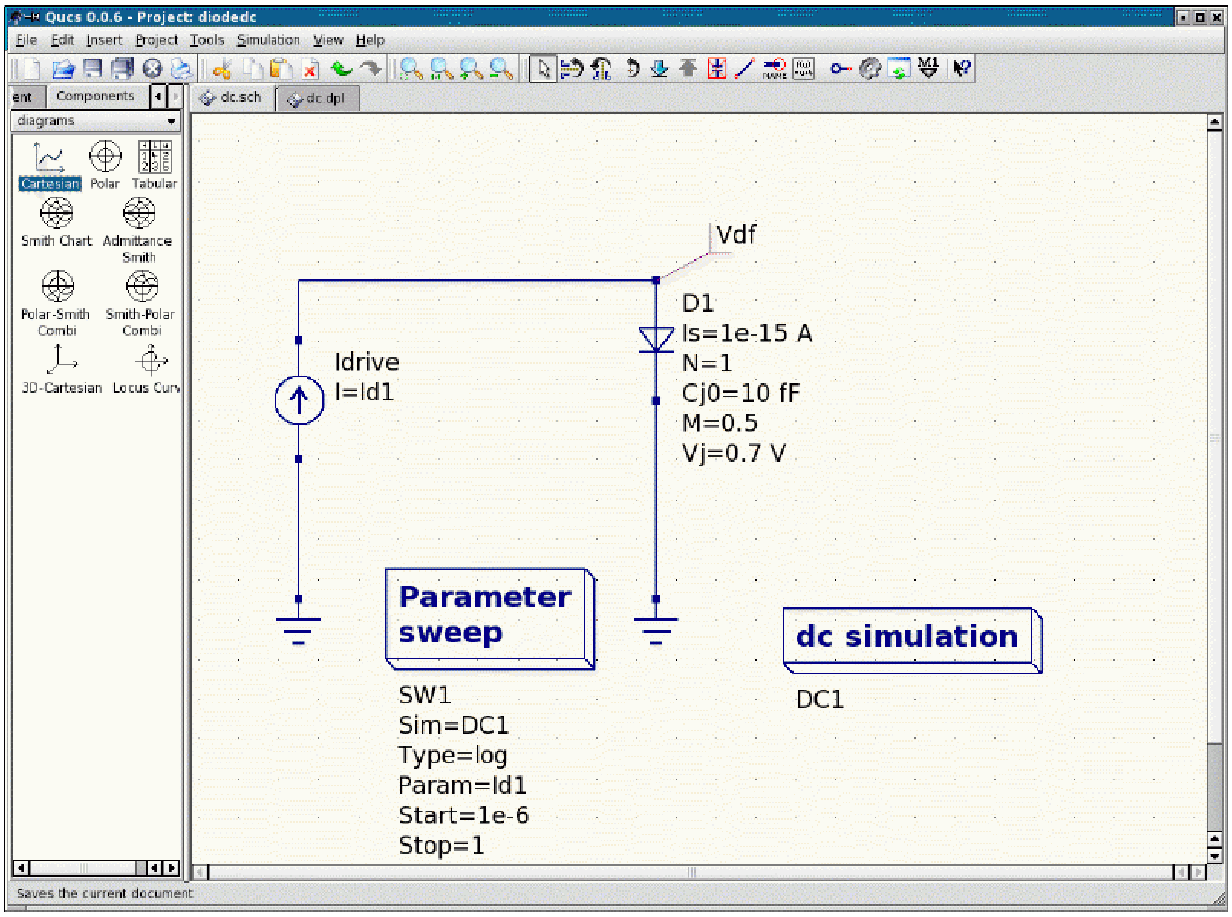
[1] I could tell you the value my simulation gave, but why should I spoil your fun.... go ahead and run it yourself. If you really want to be thorough you could then also build the circuit and measure the result.....

and the highlighted value is inviting me to type in an alternative. I could have done this, but natural laziness got the better of me. I reasoned that fifty ohms is fifty times too high, but if I reduced the current source from one amp to twenty milliamps, the output voltage would be the same. You will find such laziness (or acute perception, depending on is telling the story!) can save much time and effort.

When Things Vary

All of which is interesting, but not nearly as interesting as when we start changing things like the supply voltage and see the effects. For linear devices with a DC supply, the answer would be: not much. It's when we introduce non-linear elements that things start to happen.

The simplest non-linear element is the diode, and the question we ask most often about a diode is: how does the diode forward voltage vary with current? So back to Qucs and draw this circuit...



Idrive
I=Id1

D1
Is=1e-15 A
N=1
Cj0=10 fF
M=0.5
Vj=0.7 V

SW1
Sim=DC1
Type=log
Param=Id1
Start=1e-6
Stop=1

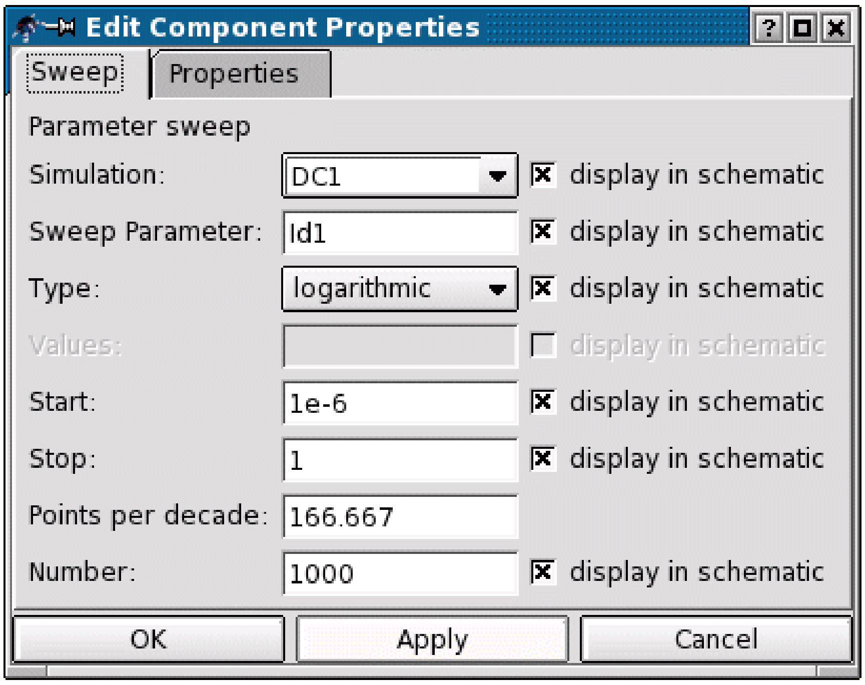
DC1

This circuit looks deceptively simple, but it introduces a few more features of Qucs, so let's go through them in order.

The components were again selected from the left hand window and wired together. Then the two boxes were selected from the **simulations** window.

The DC simulation box can be pretty much left as is for now, but take note of the name of the simulation: **DC1**.

The **Parameter sweep** box properties dialog looks like this when opened...



Parameter sweep

Simulation:	DC1	[✓] display in schematic
Sweep Parameter:	Id1	[✓] display in schematic
Type:	logarithmic	[✓] display in schematic
Values:		[ ] display in schematic
Start:	1e-6	[✓] display in schematic
Stop:	1	[✓] display in schematic
Points per decade:	166.667	
Number:	1000	[✓] display in schematic

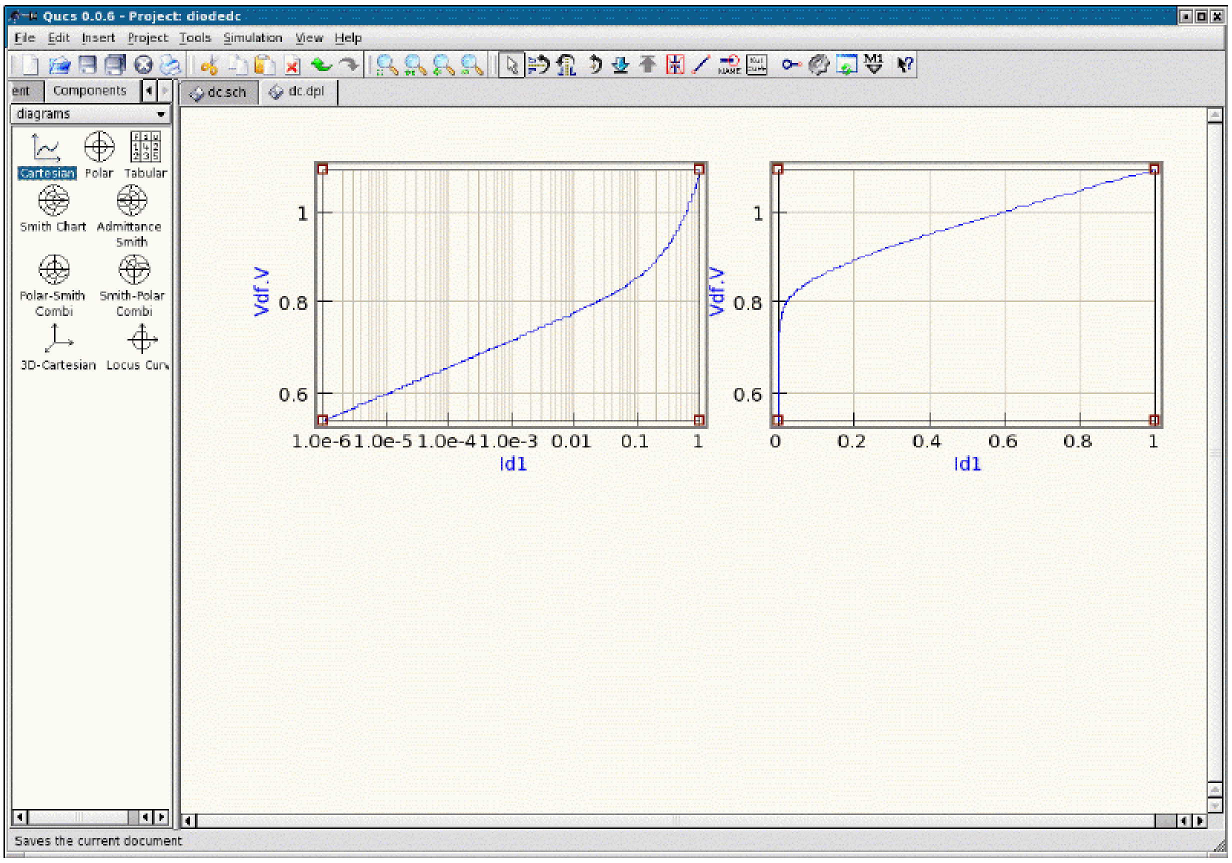
The first two items to take note of are the **Simulation** entry (here **DC1**, corresponding to the name of the simulation box) and the **Sweep Parameter** entry, here entered as **Id1**. If you look at the current source driving our diode you will see that it just happens to be labeled **Idrive**. So the result of all this is that the component property value **Id1** of the current source's property **I** will be swept through a range of values as determined by our parameter sweep function named **SW1**.[2]

The rest of the entries set the type of sweep (here logarithmic) and the range of values over which to sweep. You can try different values in any of these to see the effect; one of the advantages of a simulator over a physical prototype is that you can't blow up your diode by feeding too much current through it!

So I hit the simulation button and it passed me over the results page, and I created a

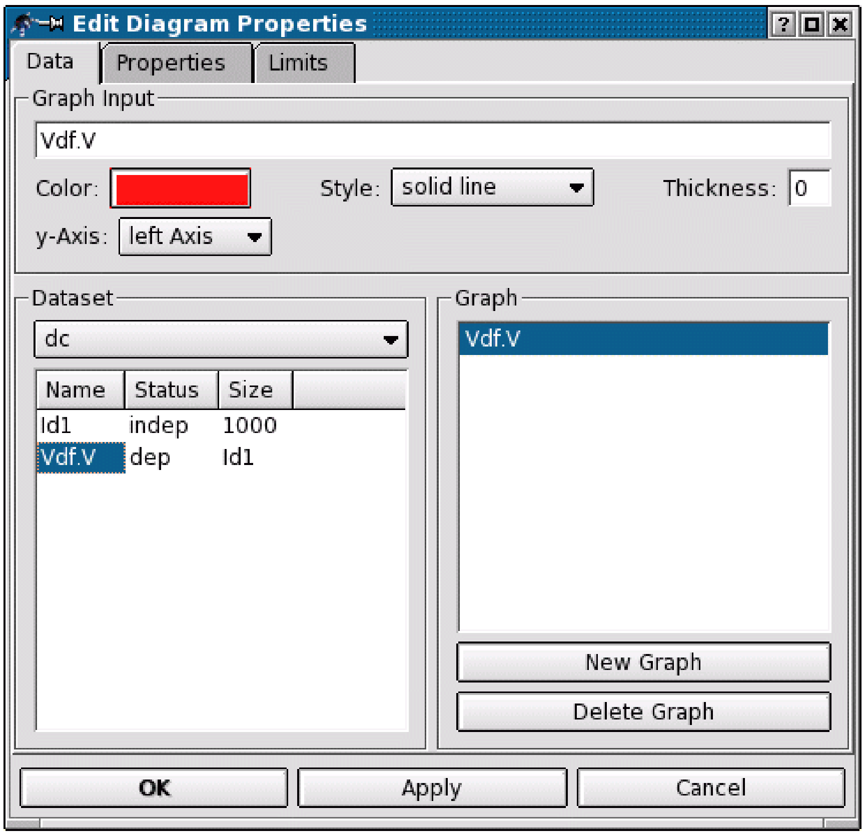
[2]You can change this name if you wish, in the Properties menu of the Edit properties window.

couple of graphs of the output. This is how my screen looked...



In each case I have a plot of diode forward voltage (Y-axis) against forward current (X-axis). The left hand graph has a logarithmic scale for forward current, while the right hand graph uses a linear current scale. How did I do that? Well, you should know by now that all things are easy with Qucs!

When you select a graph type from the left hand window and drag it into the viewing area, it creates a graph and opens a dialog which looks like this



Graph Input: Vdf.V

Color: 🟥 Style: solid line Thickness: 0

y-Axis: left Axis

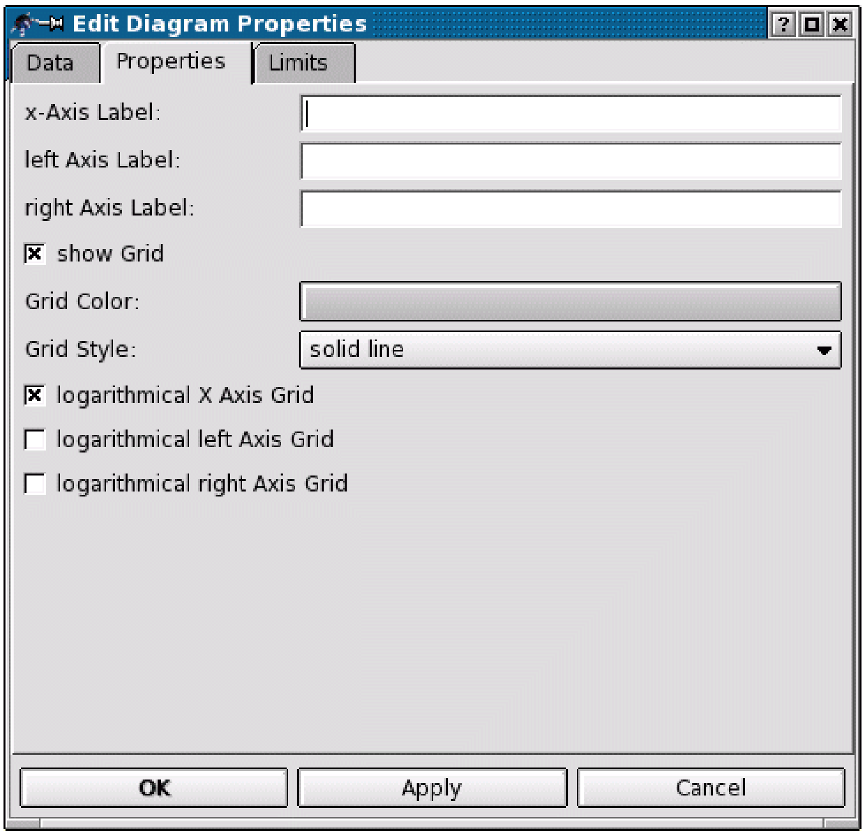
Dataset: dc

Name	Status	Size
Id1	indep	1000
Vdf.V	dep	Id1

Graph: Vdf.V

The left hand window shows the available variables and whether they are dependent or independent. Here the current Id1 is the independent variable, and the forward voltage Vdf.V is the dependent. Double-click on the entry for Vdf.V and it is transferred to the right hand side; hit OK and the graph will be drawn.

That should give you something like the right hand graph in my screenshot above. Do it all again, but this time before clicking OK open the Properties window, which looks like this.


Edit Diagram Properties

Data | Properties | Limits

x-Axis Label:

left Axis Label:

right Axis Label:

[✓] show Grid

Grid Color:

Grid Style: solid line

[✓] logarithmical X Axis Grid

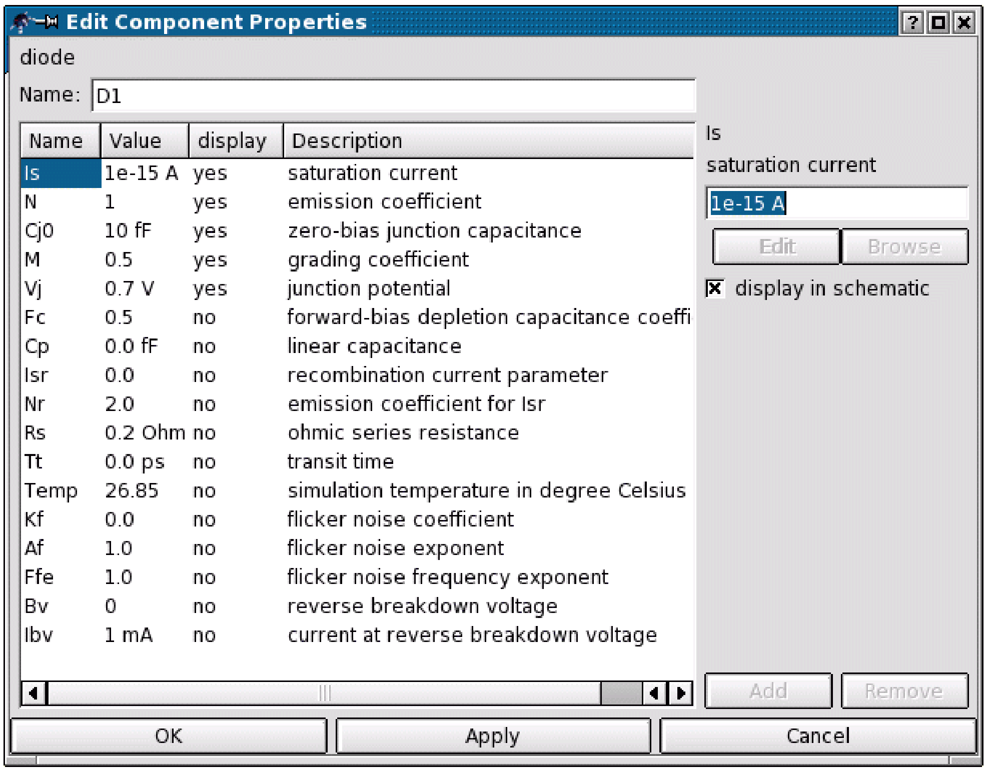
[ ] logarithmical left Axis Grid

[ ] logarithmical right Axis Grid

Here I've selected a logarithmic X Axis, which gave me the graph on the left hand side. I've also moved them around and re-sized them to pretty them up; you can do all kinds of fancy things if you want.

Now I've sneaked in another test to see if you are really following this. Those of you who did run this simulation are probably wondering about now why your graphs look rather different to mine. In particular, at high currents on the logarithmic scale your curve is a straight line, while mine curves upwards alarmingly. What is happening ?

What I did was open the Properties dialog for the diode and set some parameters. This is what the dialog box looks like...



Name: D1

Name	Value	display	Description
Is	1e-15 A	yes	saturation current
N	1	yes	emission coefficient
Cj0	10 fF	yes	zero-bias junction capacitance
M	0.5	yes	grading coefficient
Vj	0.7 V	yes	junction potential
Fc	0.5	no	forward-bias depletion capacitance coeffi
Cp	0.0 fF	no	linear capacitance
Isr	0.0	no	recombination current parameter
Nr	2.0	no	emission coefficient for Isr
Rs	0.2 Ohm	no	ohmic series resistance
Tt	0.0 ps	no	transit time
Temp	26.85	no	simulation temperature in degree Celsius
Kf	0.0	no	flicker noise coefficient
Af	1.0	no	flicker noise exponent
Ffe	1.0	no	flicker noise frequency exponent
Bv	0	no	reverse breakdown voltage
Ibv	1 mA	no	current at reverse breakdown voltage

Is
saturation current
1e-15 A
☒ display in schematic

and each of these entries sets one parameter of the virtual component we are using to model the diode.

So, what are these parameters? Time to explore one of the delights of computer circuit simulation, device modeling...

Models and Parameters

When the computer creates that small piece of virtual reality which represents your physical circuit, it uses sets of equations which describe the operation of each device you insert. The equation which relates the diode DC forward voltage as a function of current is

$$I_d = I_s \cdot \left(e^{\frac{V_d}{n \cdot V_t}} - 1 \right)$$

where V_t is the forward voltage drop at 25 degrees C of an ideal junction, also given by

$$V_t = \frac{k_B \cdot T}{q}$$

where

$$\begin{aligned} k_B &= \text{Boltzmann's constant} \\ T &= \text{temperature in degrees Kelvin} \\ q &= \text{charge of the electron} \end{aligned}$$

most of these are constants that the program already knows about. The ones we need to supply are the ones listed in the properties editor window. For the DC characteristics, most of the time, the only ones we need to worry about are Is, the saturation current, and T, the temperature. If we are going to push relatively high currents through the diode we can also include an estimate for the series resistance Rs; if we are worried about low current behaviour then we need to add the reverse current parameter Isr.

How do we know what values to insert? Much could be written about device modeling; much indeed has been written about device modeling. As always, we really have two choices: use a value from someone else, or find our own values, usually by trial and error.

There are a great many models available for various simulation programs. Probably the most freely available are those for spice, many of which can be downloaded from the semiconductor companies. Here, for example, is a typical spice model for a 1N4148 diode:[3]

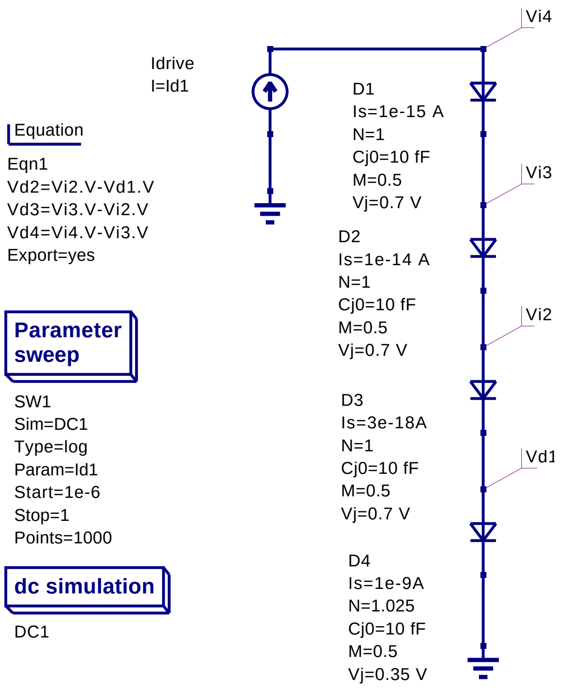
```
.model 1N4148  D(Is=0.1p Rs=16 CJO=2p Tt=12n Bv=100 Ibv=0.1p)
85-??-??   Original library
```

Any values not supplied are assumed to be the defaults.

The other way is to create your own device parameters, which is a bit like catching worms before you can go fishing. Insert values, plot the resulting characteristics, see how they compare with the published data sheet values, go back and adjust the values; continue until satisfied or exhausted.

Here, for example, is a circuit for quickly comparing the forward characteristics of diodes with different parameter values.

[3]I don't know where this came from, so I can't acknowledge the author. Most libraries are copyright, even if freely available.



And here is the plotted output...

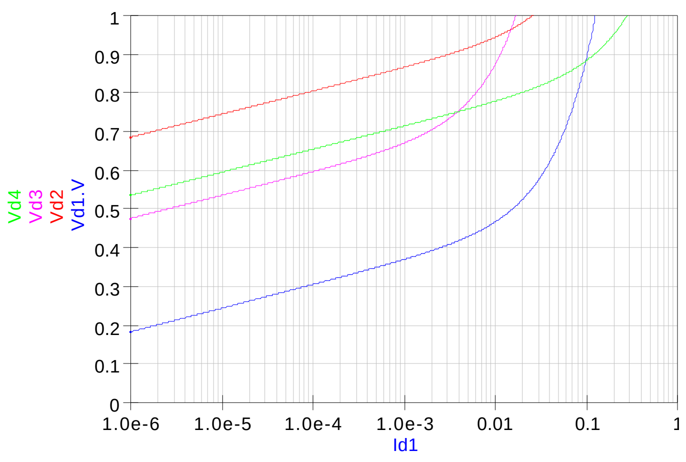


Figure 3: Diode Forward Voltage

The green and purple curves are typical of 1N4148 and 1N4448 devices; the others are medium and low-barrier Schottky devices. I have done a first pass compare with the data sheets, but I can't guarantee that these curves are any more than my best estimates.[4]

If you want to know more details of what each parameter does, there has been a great deal written over the years, particularly for spice, on the subject; a google search will quickly

[4]I'm assuming you are sick of screenshots by now, so I've just printed the schematic and display files from Qucs; you'll find the print item in the file menu, and if you ask it nicely it will print a postscript file.

reveal most of it. Qucs comes with a document which lists the details of its models, and, being open source, there is always the code itself.

Most of us end up taking a great deal on trust, and matching curves to data sheets as best we can. This is yet another instance of one of the fundamentals of engineering, the Duck Principle[5]: If you can't detect any difference between the behaviour of your model and the physical device, then they are, for engineering purposes, the same. Put it another way, when the difference between the model and the real device drops below the usual level of measurement uncertainty, it does matter any more.

In any case, component spreads in the real world tend to make the fine details of model inaccuracies somewhat academic, as we shall see when we model more complex devices.

[5]Usually expressed as: If it looks like a duck, walks like a duck, quacks like a duck and tastes like a duck, then, for all practical purposes, it is a duck.