Lab 1 SPICE Tutorial

Chris Winstead

August 29, 2014

Getting Started

SPICE is designed to run as a classic console tool, aka a terminal command. If you are unfamiliar with the Linux terminal, you should spend some time to get acquainted with basic terminal commands, and how to organize and navigate directory structures (a directory is often called a "folder"). I prepared a quick-start terminal tutorial that you can review here: `https://electronics.wiki.usu.edu/Linux_Tutorial`

If you plan to use NGSpice in the lab (which I recommend), then you may want to check out our NGSpice wiki page: `https://electronics.wiki.usu.edu/NGSpice` You can also find NGSpice information and the full manual here: `http://ngspice.sourceforge.net/`

You will also need to choose a text editor for preparing your SPICE files. I like to use Emacs (it has an optional SPICE mode that is pretty handy). Most students prefer to use GEdit. You can launch these editors from the terminal.

Basic SPICE Structure

- The first line is the circuit name
- Comments begin with *
- The file terminates with a .end
- External files are included with .include
- Parameters are declared with .param

A stupid example

This example creates a 10V DC voltage source. It includes a title line, a commented file description, an include statement, a parameter statement, and an analysis statement (.op).

```
Stupid Example Circuit
******************************
** Created by Chris Winstead
** for no specific purpose
******************************
.include my_models.sp
.param  VDD=10

V1 1 0 DC VDD

.op

.end
```

The source is named "V1", it connects between nodes '1' and '0' (node '0' is always ground). The voltage is set by a parameter.

Constructing Your Circuit

In the following slides you will assemble the pieces needed to model the circuit from Lab 1. You should use the same basic structure from the previous slide.

For reference, please refer to the excellent guide provided by Jan Van der Spiegel at the University of Pennsylvania:
`http://www.seas.upenn.edu/~jan/spice/spice.overview.html`

There is also some information on SPICE simulations in your textbook.

Basic Components

This fragment describes the resistors, capacitors and voltage sources used in our lab assignment:

```
* Main power supplies:
VDD nvdd 0 DC 15V
VSS nvss 0 DC -15V

* Linear passive components:
C1 nvsig nvx 100uF
C2 nvy   nvz 1uF

R1  nvdd  nvx 1k
R2  nvdd  nvy 100k
R3  nvz   0   1Meg
R4a nvg   0   50k
R4b nvout nvg 50k
```

Notice that my node names all start with the letter 'n' so that they are not confused with parameter or component names. Also notice that Meg has to be spelled out because SPICE is not case sensitive, so it will think 'M' is "milli." Also, I've split potentiometer R4 into two resistors, R4a and R4b. Their center-tap is connected to the op amp's virtual ground node, which I've named "nvg".

Signal Sources

Signal sources can simultaneously specify behaviors for DC, AC and transient simulations:

```
* Signal source:
Vsig nvsig 0 DC 0V AC 1V SIN(0V 100mV 50e3)
```

This says that our source is 0V for DC analysis, 1V amplitude for AC analysis, and for transient analysis it will have 0V offset, 100mV amplitude and a 50kHz frequency.

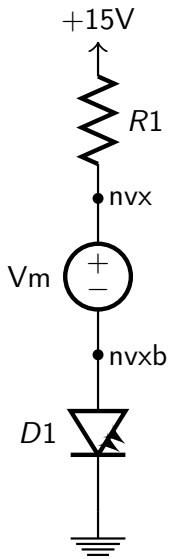
It's important to specify all these behaviors because SPICE will likely be used to perform several different types of analysis.

Meter Sources

Our circuit contains coupled devices (the LED and photodiode), in which one device's current is proportional to the other. To model this, we need to sense the current flowing through the first device:

```
* Place a "meter" source to monitor the LED forward current
Vm nvx   nvxb DC 0
```

Here I've broken node "nvx" into two nodes: "nvx" (connected to C1 and R1) and "nvxb" (which will connect to D1). The meter source Vm bridges between these nodes. Since Vm is a 0V source, it has no electrical effect on the circuit.



Diodes

Now let's place our diodes:

```
* LED and Photodiode:
D1 nvxb  0    SFH486
D2 0     nvy  SFH235
```

The diode statement lists its two terminals (anode then cathode) followed by a model name. Note that D2 is reverse biased, so its nodes are listed in reverse order.

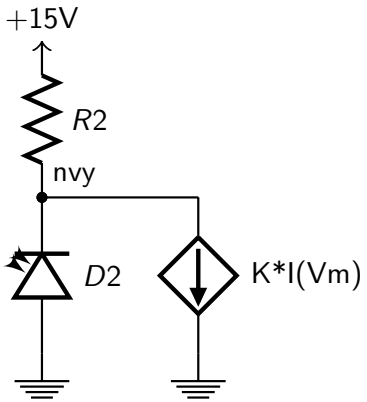
Since diodes are complex devices, SPICE needs to be given a model containing all their physical characteristics. For our class, these models are placed in a file called lab_parts.md. You should place this model file in your project directory and load it with an include statement near the top of your SPICE file:

```
.include lab_parts.md
```

Current Controlled Current Source

To model the photocurrent in D2, we insert a CCCS in parallel with D2. This is done using an 'F' component in SPICE:

```
* Place a current-controlled current source to model the photo-induced current in D2:
F1 nvy   0   Vm K
```



You will also want to declare the Coupling Coefficient K as a parameter near the top of your SPICE file, e.g.:

```
.param K=0.001
```

Modeling the Op Amp

Now we need to place a model of the 741 op amp:

```
* Place a 741 op amp model as the output amplifier:
X1 nvz  nvg  nvdd nvss nvout uA741
```

The 'X' indicates a subcircuit model; the last entry on the line is the model name, uA741.
This model is a complex circuit defined in lab_parts.md. Here is the top portion of the uA741 model, which explains its usage:

```
* connections:   non-inverting input
*                |  inverting input
*                |  |  positive power supply
*                |  |  |  negative power supply
*                |  |  |  |  output
*                |  |  |  |  |
.subckt uA741    1  2  3  4  5
*....[complex circuit description here]....
.ends
```

All subcircuit models end with a .ends statement.

Analysis Commands

SPICE won't do anything unless you give it an analysis command. Here we will do AC simulation and TRANsient simulation:

```
* Simulation commands:
.tran 100ns 600us
.ac dec 10 100 10e6
```

First, we specify transient simulation with a 100ns time step, ending at 600µs.

Second, we specify an AC simulation, sweeping the frequency with 10 points per decade, from 10Hz up to 10MHz.

Control Commands

NGSpice allows control scripts that are placed inside a .control block. These commands allow you to do more complex things:

```
 .control
ac dec 10 100 10e6
plot xlog db(v(nvx))
plot xlog db(v(nvy))
plot xlog db(v(nvz))
plot xlog db(v(nvout))

tran 100ns 600us
plot xlimit 0 60u v(nvz) v(nvout) v(nvsig) v(nvy)

fft v(nvout)
plot xlog xlimit 10e3 1e6 db(v(nvout))
.endc
```

This example performs AC and transient analyses and auto-generates desired plots. Then it computes an FFT from the transient result and displays the result in a log-scale plot. The control block must be terminated with a .endc statement.

Making Hardcopy Plots

You can export your plots to a printable image by using the hardcopy command, which has the same syntax as the plot command (you just specify the filename first):

```
.control
ac dec 10 100 10e6
hardcopy dbx.ps xlog db(v(nvx))
hardcopy dby.ps xlog db(v(nvy))
hardcopy dbz.ps xlog db(v(nvz))
hardcopy dbout.ps xlog db(v(nvout))

tran 100ns 600us
hardcopy tran.ps xlimit 0 200u v(nvz) v(nvout) v(nvsig) v(nvy)

fft v(nvout)
hardcopy fft.ps xlog xlimit 10e3 1e6 db(v(nvout))
.endc
```

This command generates PS files. You can convert them to PDF using the "pstopdf" command. You can view the PS or PDF file using the "evince" command. You can have both "hardcopy" and "plot" commands together in a control block.

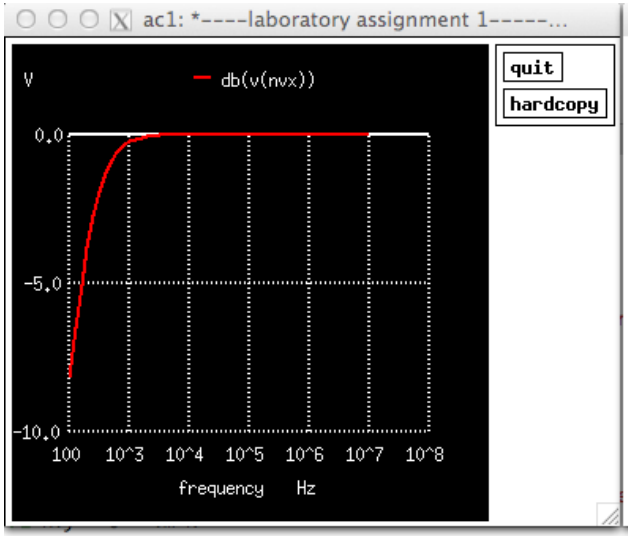
Running the Simulation

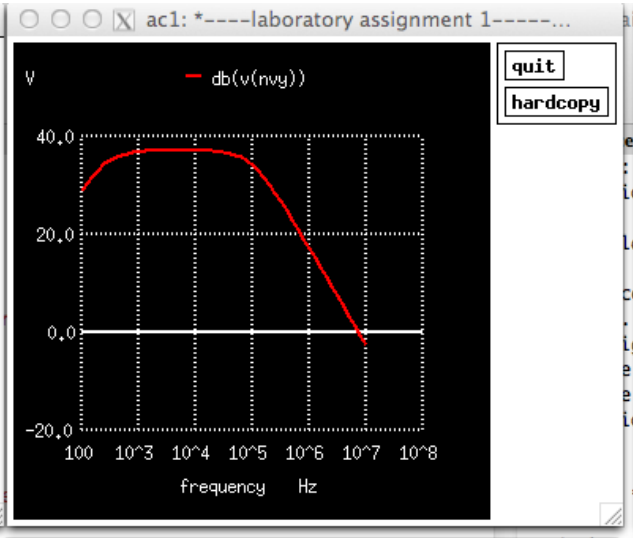
If you correctly prepared your SPICE file and saved it as "lab1.sp", and if your directory is correctly organized with the lab_parts.md file, you can run the simulation using this terminal command:

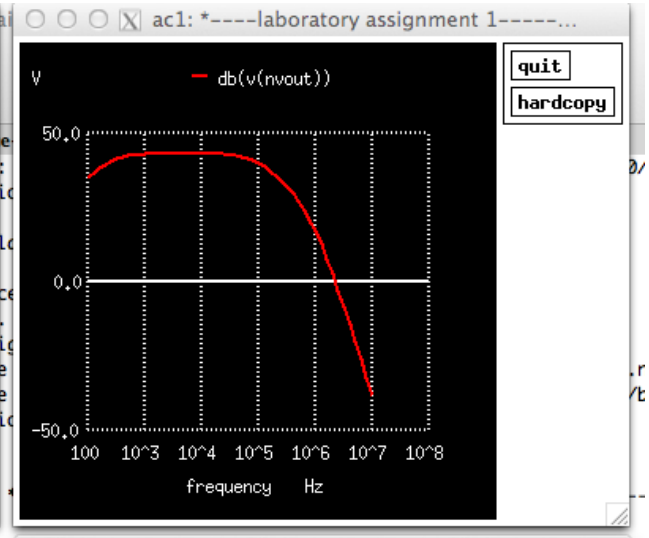
```
ngspice lab1.sp
```

If you used the control blocks, then the simulations should run automatically and the plots should appear.

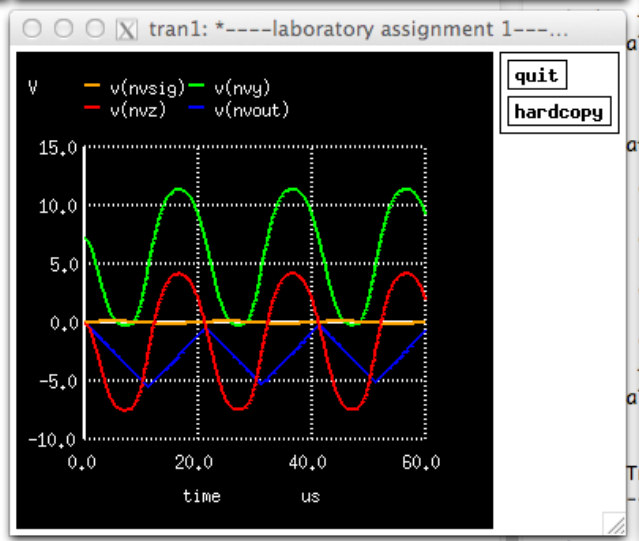
What You Should See

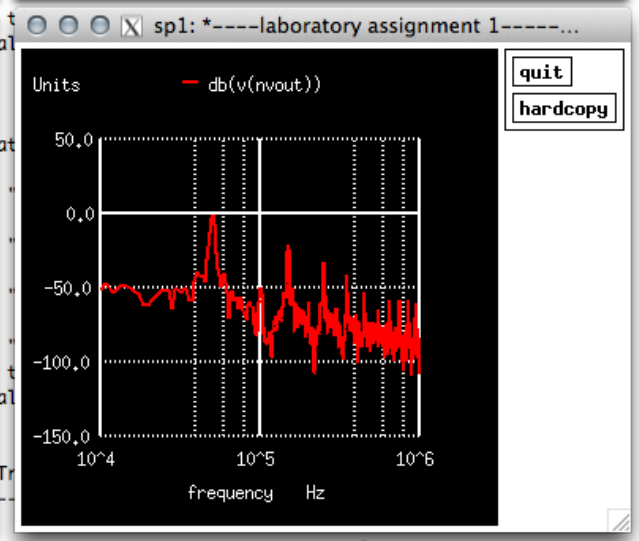







```
* Place a 741 op amp model as the output amplifier:
X1 nvz  nvg  nvdd nvss nvout uA741

* Simulation commands:
.tran 10ns 2ms
.ac dec 10 100 10e6

* Control Script:
 .control
 ac dec 10 100 10e6
hardcopy dbx.ps xlog db(v(nvx))
hardcopy dby.ps xlog db(v(nvy))
hardcopy dbz.ps xlog db(v(nvz))
hardcopy dbout.ps xlog db(v(nvout))
plot xlog db(v(nvx))
plot xlog db(v(nvy))
plot xlog db(v(nvz))
plot xlog db(v(nvout))

tran 100ns 600us
hardcopy tran.ps xlimit 0 200u v(nvout) v(nvsig)
plot xlimit 0 60u v(nvz) v(nvout) v(nvsig) v(nvy)

fft v(nvout)
plot xlog xlimit 10e3 1e6 db(v(nvout))
.endc

.end
-:---  lab1.sp      Bot (66,0)     (Hspice/Eldo-RF-VIA/FastHenry|Spice Compiling)
```





Node	Voltage
nvdd	14
nvss	-15
nvsig	0
nvx	1.32088
nvy	7.15327
nvz	-0.0799072

```
-:%*-  *compilation*   Top (1,0)     (Compilation:run Compiling)
```

Finishing Up

When you're done analyzing the results, type quit to exit NGSpice. If you want to change your SPICE file and re-run the simulation, the easiest way is to quit and restart ngspice.

In many cases, you can use NGSpice to do all your analyses. Sometimes NGSpice doesn't do exactly what you need, and in that case you can import the simulation data into Matlab. To do this, you need to run NGSpice with raw output:

```
ngspice lab1.sp -r lab1.raw
```

Then, if you have downloaded the ReadNGSpice script, you can launch Matlab and load the results using the command "ReadNGSpice(lab1.raw)".

That's All!