SPICE 1: Tutorial

Chris Winstead

January 6, 2015

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.

Creating a Project

First, you'll want to open a terminal window and create a directory tree for your work this semester. You could setup your directory tree using these commands:

```
cd
mkdir 3410
cd 3410
mkdir spice
cd spice
mkdir lab1
cd lab1
```

Here the `cd` command is used to change directories, and the `mkdir` command is used to create a directory.

Create a new SPICE file

SPICE files (often called “decks” for historical reasons) are plain text files. In this course we will use the `.sp` file extension (there is no standard file extension for SPICE; some people use `.cir` or `.ckt`).

To make a SPICE file, use the text editor of your choice. Most students use gedit, which is a basic text editor. From the terminal, type

```
gedit circuit1.sp &
```

The ‘&’ symbol is important, since it tells gedit to run in the background.

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

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.

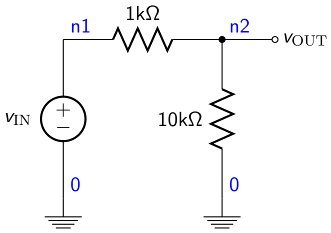
For basic language reference, please refer to the excellent guide provided by Jan Van der Spiegel at the University of Pennsylvania:
`http://www.seas.upenn.edu/~jan/spice/spice.overview.html`

A comprehensive NGSpice Manual is available here:
`http://ngspice.sourceforge.net/docs/ngspice-manual.pdf`
There is also some information on SPICE simulations in your textbook.

First example

This example describes Fig. 1 from Pre-Lab 1, Exercise 2.

```
Lab 1, Circuit 1
******************************
** Created by Chris Winstead
******************************

Vin n1 0 DC 5V

R1  n1 n2 1k
R2  n2 0  10k

.control
dc Vin 0V 5V 0.1V
plot v(n2)
hardcopy circuit1_dc.ps v(n2)
meas dc y FIND v(n2) AT=1
echo Solution: vout = vin*$&y
.endc

.end
```



The circuit's node names are labeled in blue. In the following slides, we will decompose each line of this circuit description.

Voltage Sources

Circuit 1 includes a simple DC voltage source defined by the syntax:

```
V<name> nplus nminus DC <value>V
```

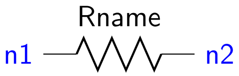
A voltage source line always starts with ‘V’. It is followed by two node names. The source’s positive terminal is named first, followed by the negative terminal. Note that a ground node must always be specified, and is always named 0 (this is important for SPICE’s internal numerical calculations).

This source could be converted into a current source by changing the leading ‘V’ to an ‘I’. There are several other types of sources that we’ll study in future examples.

Resistors, Capacitors, Inductors

Our circuit includes two resistors, specified by the ‘R’ device type:

```
R<name> n1 n2 <value><unit>
```



The same procedure can be used to create a capacitor or inductor by changing the leading ‘R’ to a ‘C’ or ‘L’, respectively.

Units

The units (kilo, milli, micro, etc) can be specified using mostly normal symbols:

Symbol	Unit	Sumbol	Unit
m	milli	k	kilo
u	micro	Meg	mega
n	nano	G	giga
p	pico		

Since SPICE is not necessarily case-sensitive, you have to actually spell out "Meg" to indicate "Mega". A capital 'M' will be interpreted as "milli".

Control Blocks

Many versions of SPICE allow control blocks, which can be used to automate simulations, plots and measurements. A control block begins with `.control` and ends with `.endc`. In this example, we perform a DC simulation:

```
dc <component name> <start> <stop> <step>
```

This line requests a DC Sweep simulation of the named voltage source. The source will be varied from `start` to `stop` in increments of `step`.

The component name doesn't have to be a voltage source. It could be a resistor or other component. For example, we could sweep resistor R1 from 1 kΩ to 10 kΩ in steps of 100 Ω:

```
dc R1 1k 10k 100
```

Plot Commands

Within a control block, we can execute various types of plot commands. The simplest plots are node voltages, like this:

```
plot v(<node name>)
```

Plotting a current is not as simple. Because of SPICE's numerical design, you can only plot currents through voltage sources, like this:

```
plot -i(Vin)
```

The above command plots the current passing through our example circuit.

The Hardcopy Command

The hardcopy command produces a graphics file containing a desired plot. It can be used within a control block, and its syntax is similar to the plot command:

```
hardcopy <filename> <plot_command>
```

In our example, we have:

```
hardcopy circuit1_dc.ps v(n2)
```

This command will produce a PostScript file called `circuit1_dc.ps` containing a plot of v_{OUT}. To view this file, `quit` NGSpice and use the `evince` terminal command:

```
evince circuit1_dc.ps &
```

You can also convert the file to PDF format by using the `ps2pdf` command.

Meas and Echo commands

The meas command can be used to take precise measurements from a SPICE simulation. In future exercises, we will see several examples of how this command can be used. For this example, we use the command to simply measure v_{OUT} at the point where $v_{\mathrm{IN}} = 1\,\mathrm{V}$:

```
meas dc y FIND v(n2) AT=1
```

In this statement, the measurement result is stored in a "vector" named y. The raw result is printed directly, but additional context can be provided by using an "echo" command:

```
echo Solution: vout = vin*$&y
```

In the above command, everything is just text except for "$&y", which references the value of the measurement result in y.

Running the Simulation

After entering the circuit description, save the file as `3410/spice/lab1/circuit1.sp`. Then, from the terminal, run the command:

```
ngspice circuit1.sp
```

You should see a straight-line plot showing v_{OUT} vs v_{IN}. In the console, you should see a line saying:

```
Transfer Characteristic:  vout = vin*0.909091
```

When you are finished, type `quit` in the terminal to exit the simulation.

Logging Your Results

When running the simulations in this lab, paste your results into a text file named "log.txt" located in your project directory. In this file, copy the results of any meas or echo commands produced by NGSpice. It should look like this:

```
*** RESULTS FOR CIRCUIT 1:
y                    =  9.090909e-01
Transfer Characteristic: vout = vin*0.909091
```

Add to this file as you complete the different portions of this tutorial.

Second Example: Transient Simulation

We now modify the example to perform a time-domain transient simulation using a sinusoidal input signal.

```
Lab 1, Circuit 1
*****************************
** Created by Chris Winstead
*****************************

Vin n1 0 SIN(1V 2V 10k)

R1  n1 n2 1k
R2  n2 0  10k

.control
tran 10us 1ms
plot v(n2)
hardcopy circuit1_tran.ps v(n2)
.endc

.end
```

Modify your `circuit1.sp` file and save it as “`circuit1_tran.sp`”.

Transient Simulation Command

A transient simulation shows the circuit's time-domain behavior, and is specified in the control block using this command:

```
tran <tstep> <tstop>
```

This will execute a simulation from time 0 up to time `tstop` with an initial time-step `tstep` (note that SPICE will vary the actual time-step dynamically as simulation progresses).

Sinusoidal Sources

For time-domain (i.e. transient) simulations, we declare a sinusoidal source:

```
V<name> <nplus> <nminus> SIN (<offset> <amplitude> <frequency>)
```

In our example, we gave this declaration:

```
Vin n1 0 SIN(1V 2V 10k)
```

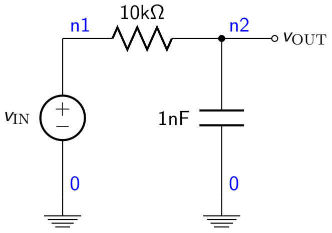
This asks for a sinusoidal voltage source with a frequency of 10kHz, an offset of 1V and an amplitude of 2V.

Third Example

This example describes Fig. 2 from Pre-Lab 1, Exercise 3.

```
Lab 1, Circuit 2
** Created by Chris Winstead

Vin n1 0 DC 0V AC 1 SIN(1V 2V 10k)

R1  n1 n2 10k
C1  n2 0  1n

.control
**********************
* TRANSIENT SIMULATION
**********************
tran 1u 1m
plot v(n2)
hardcopy tran_circuit2.ps v(n2)

**********************
* AC SIMULATION
**********************
ac dec 10 100 10e6

set units=degrees
plot vdb(n2)
plot vp(n2)
hardcopy magnitude_circuit2.ps vdb(n2)
hardcopy phase_circuit2.ps vp(n2)

*... continues on right column ...
```

```
*... continuation:

meas ac y1 FIND vdb(n2) AT=1k
meas ac y2 FIND vdb(n2) AT=10k
meas ac y3 FIND vdb(n2) AT=50k
meas ac f3dB WHEN vdb(n2)=-3
meas ac p3dB FIND vp(n2) AT=$&f3dB
echo The -3dB frequency is $&f3dB Hz
echo with Phase phi=$&p3dB degrees
.endc

.end
```



AC and Transient Sources

In this example, our voltage source has been extended to include AC and Transient behavior. The AC portion declares that source will be used as a signal source in AC simulations:

```
V<name> <nplus> <nminus>  AC <amplitude>
```

In most cases, you should set the AC amplitude to 1.

It's important to distinguish SIN from AC — these refer to different types of simulation modes and have to be declared separately.

Multi-Mode Sources

We can combine all behaviors (DC, AC and transient) in a single statement:

```
Vin n1 0 DC 0V AC 1 SIN(1V 2V 10k)
```

This statement declares a voltage source with the following characteristics:

- For DC simulations: 0V
- For AC simulations: 1V amplitude
- For transient simulations: a 10kHz sinusoidal signal with 1V offset and 2V amplitude.

AC Simulation Command

AC simulations are used to obtain a circuit's frequency response and Bode plot. An AC simulation is requested as follows:

```
ac dec <numpoints> <fstart> <fstop>
```

This command requests a frequency sweep from `fstart` up to `fstop` with `numpoints` samples per decade. In our example, we have:

```
ac dec 10 100 10e6
```

This command requests a sweep from 100Hz up to 10MHz. It will calculate the transfer response at 10 points spaced logarithmically between 10Hz and 100Hz, then another 10 points spaced between 100Hz and 1kHz, and so on.

Obtaining a Bode Plot

The `plot` and `hardcopy` commands have many flexible options for plotting different types of data. In our example, we generate a Bode magnitude plot using the expression `plot vdb(n2)`. This plots the mangitude of v_{OUT} in dB. Since we declared v_{IN} as an AC input with magnitude 1, the transfer function is

$$|H(\omega)| = \left|\frac{v_{\mathrm{OUT}}(\omega)}{v_{\mathrm{IN}}(\omega)}\right| = |v_{\mathrm{OUT}}(\omega)|\,.$$

We similarly obtain the phase plot by calling `plot vp(n2)`.

Note that SPICE reports phase in radians by default, so to get output in degrees we used the `set units=degrees` option.

Meas... When Command

In this example, we took advantage of the meas command to find the -3dB frequency and the phase shift at that frequency. To obtain the -3dB frequency, we used this command:

```
meas ac f3dB WHEN vdb(n2)=-3
```

This command uses the WHEN option to locate the frequency *when* vdb(n2)=-3. The result is stored in a variable called f3dB. To get the phase shift, we use the FIND...AT syntax to get the phase shift at the frequency equal to f3dB:

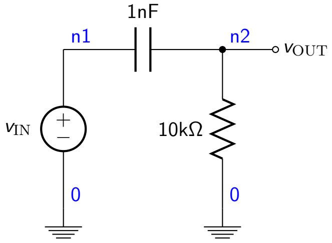
```
meas ac p3dB FIND vp(n2) AT=$&f3dB
```

The phase is calculated at the specified frequency and stored in a variable called p3dB. These results are then printed using an echo statement.

Exercise

In this exercise, you will compose a SPICE description for the circuit in Fig. 3, Pre-Lab 1, Exercise 4. Save the file as `circuit3.sp`. Use appropriate control commands to perform the required measurements specified in the exercise.

Use the `hardcopy` command to produce plots of your simulation result. Save the plots with names like `circuit3_tran.ps`, `circuit3_magnitude.ps`, and `circuit3_phase.ps`.



What to Turn In

After completing all the assigned simulations, create a ZIP file containing all of your .sp and .ps files, along with your results.txt file. In Linux, you can produce a ZIP file using this command:

```
zip spice1.zip *.sp *.ps *.txt
```

After creating your ZIP file, verify it using the command

```
unzip -l spice1.zip
```

If all the files appear in the list, then open a web browser and submit the ZIP file in Canvas.

That's All!