ECE206s LTSpice Reference

University of Illinois at Urbana-Champaign

March 20, 2020

1 Introduction

In this class, we'll be using `SPICE` to simulate our circuit designs. `SPICE` stands for *Simulation Program with Integrated Circuit Emphasis*, and is an essential part of modern circuit design. `SPICE` comes in many "flavors" like `HSPICE`, `PSPICE`, `ngspice`, and many others, all of which perform the same front-end function of circuit simluation, but with different "behind-the-scenes" optimizations. `LTSpice` is a free-to-use `SPICE` simulator created by Linear Technologies (now part of Analog Devices), and we'll be using it for this class. This document is a reference manual for the various functions and tools in `LTSpice`. As a general rule, **bold text** means a keyboard shortcut, whereas `monospace text` is a menu option.

2 Keyboard Shortcuts

While you can click through all the menus you like, sometimes knowing the keyboard shortcuts are faster. These are all for the *schematic editor*:

F1 - Open the Help menu

F5 - Delete an item

F6 - Copy an item

W - Select the wiring mode

E - Add a net name

R - Add a resistor

T - Add plain text

S - Add `SPICE` text

D - Add a diode

F - Find/add a part

G - Add a ground port

X - Add an inductor

C - Add a capacitor

Space - Zoom full

Ctrl+W - Move

Ctrl+E - Mirror

Ctrl+R - Rotate

Ctrl+T - Toggle pins

Ctrl+Y - Redo

Ctrl+S - Save File

Ctrl+D - Drag

Ctrl+F - Find text

Ctrl+G - Toggle grid

Ctrl+H - Halt simulation

Ctrl+Z - Undo

Ctrl+X - Delete

Ctrl+C - Copy

Ctrl+V - Paste

Ctrl+B - Run simulation

Ctrl+N - New schematic

3 Getting Started

Open `LTSpice` by either clicking on the desktop shortcut, or going to `Start`→`Programs`→`LTSPICE XVII`. You'll see Figure 1 upon starting up:

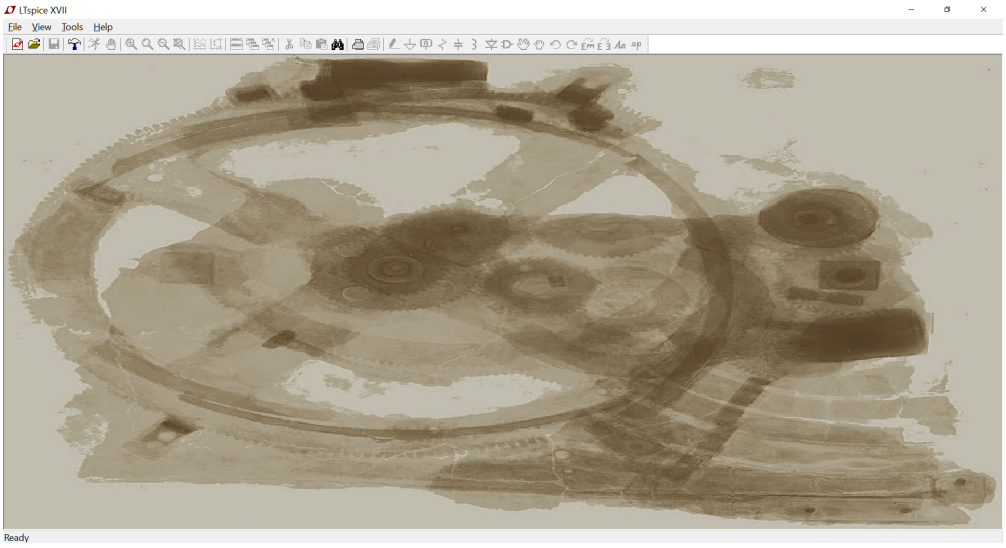

Figure 1: LTSpice startup screen. Engineering sketches made out of coffee stains!

Let's get started by creating a new schematic. Go to `File` →`New Schematic`, or hit **Ctrl+N**, and the coffee stains should disappear, leaving you with a fresh page. Let's make our first circuit, shown in Figure 2. To add the voltage source, press the AND gate symbol (or **F**) and search for the `voltage` part.

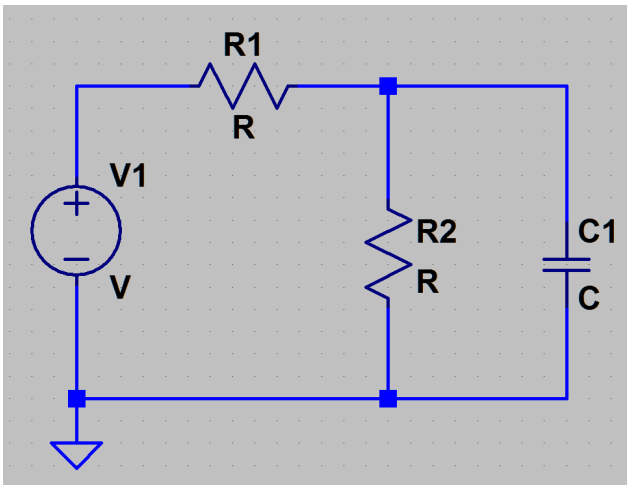

Figure 2: Basic lowpass filter schematic.

You should get the following menu:

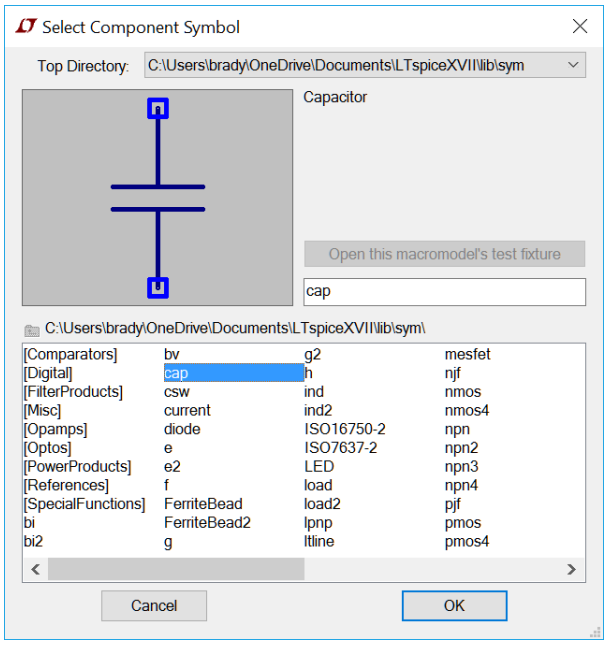

Figure 3: Adding components window.

Now we can assign values to all the parts by right-clicking on them. Note `LTSpice` has two useful features: first, it auto-assigns the units for us, so we can say the resistance is 50 or 50 Ω, it doesn't care. Second, it implements the SI prefixes, so we can say 4.7n instead of 0.0000000047. However, it is case-insensitive! 10mΩ is 10 `milli-`Ω, and 10 `Meg`Ω is 10

`Mega-Ω`. Just putting 10 `M` would be interpreted as the `milli-` prefix, so be careful! As an example, we chose R1 = 1 kΩ, R2 = 10 kΩ, and C1 = 1 μF. Note we can also choose a specific "real-life" model, which will come in handy when using active devices.

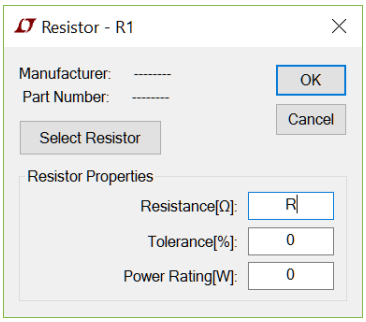

Figure 4: Editing components window

Next we'll set V1. Right-click on it and choose `advanced`. Copy the values shown in Figure 5, and be sure to also set the small signal values AC values in the right hand side column.

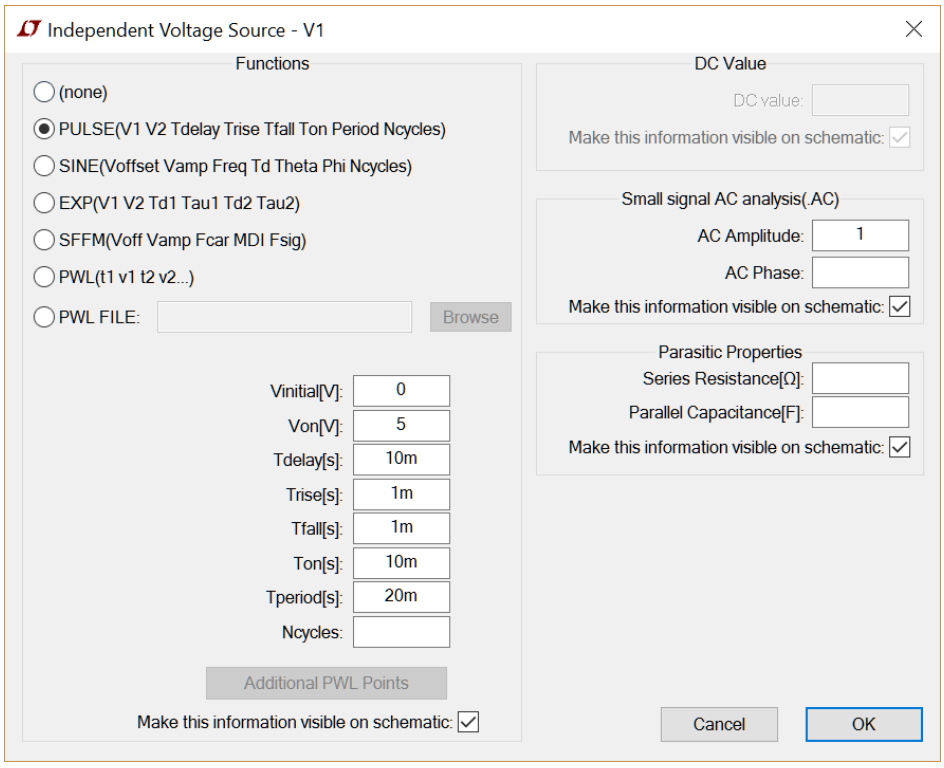

Figure 5: V1 settings.

The different function choices are:

(none) - Just a constant DC output

PULSE - Square wave output (series of pulses)

SINE - Sinusoidal wave

EXP - Exponentially decaying/rising wave

SFFM - Single frequency FM wave

PWL - Piece-wise linear wave described by a set of (x, y) points

3.1 Netlists

LTSpice doesn't work with our schematic directly, but instead compiles it down to a *netlist*. You can view the netlist by going to View →SPICE Netlist. Our example netlist is:

```
V1 vin 0 5 AC 1
R1 out vin 1k
R2 0 out 10k
C1 out 0 1?.backanno
.end
```

Each line can be read as: [Device Type][Device Name] [Connection nodes] [Parameters]. The device type is selected based on it's first letter, so all resistors must be called R<name>, similarly all capacitors are named C<name>. Based on the device type, SPICE will then read the next N "words" as connections. The remaining "words" are treated as parameters.

4 Simulations

`LTSpice` has five total simulation types: transient, AC, DC, noise, TF, and OP. We'll only elaborate on the common ones here, more information about the other two can be found online. To add a simulation type, go to `Simulate` →`Edit Simulation Command`. Once you have created one, you can right-click on the command to bring the menu back up. The simulation commands appear as text on the schematic, in the exact same way it would in a netlist. You can run a simulation by pressing the "running man" icon, or pressing **Ctrl+B**. Note only one simulation can be run at a time, but you can have several simulation commands at once. `LTSpice` will simply ask you when simulation you want to run. While a source can have multiple simulation properties (e.g., a voltage source can output a `PULSE` in transient and a sine wave in AC), only the parameters relevant to the simulation will be used.

4.1 .op

Operating point simulations are used to quickly measure DC voltages and currents. All inductors are treated as shorts, and all capacitors are treated as open. It prints out the voltage at every node, and the current through every device. For this exercise, you will use DC voltage source by selecting `(none)` in the setting and choosing DC value of 5. Our example output is shown below. Note the capacitor current is not exactly zero due to numerical errors, but can be treated as zero for all purposes.

```
       --- Operating Point ---
V(out):  4.54545  voltage
V(vin):  5  voltage
I(C1):  4.54545e-018  device_current
I(R2):  -0.000454545  device_current
I(R1):  -0.000454545  device_current
I(V1):  -0.000454545  device_current
```

4.2 .tran

In transient simulations, we measure our outputs over time. Figure 6 shows the settings menu. In this class, you will rarely have to edit anything besides the stop time. The maximum time-step can be set if your waveforms look "choppy" or piece-wise, instead of a smooth curve. Starting external supplies at 0 V helps with stability occasionally, but is generally not needed. Make sure to change the voltage source setting back to the `PULSE` mode.

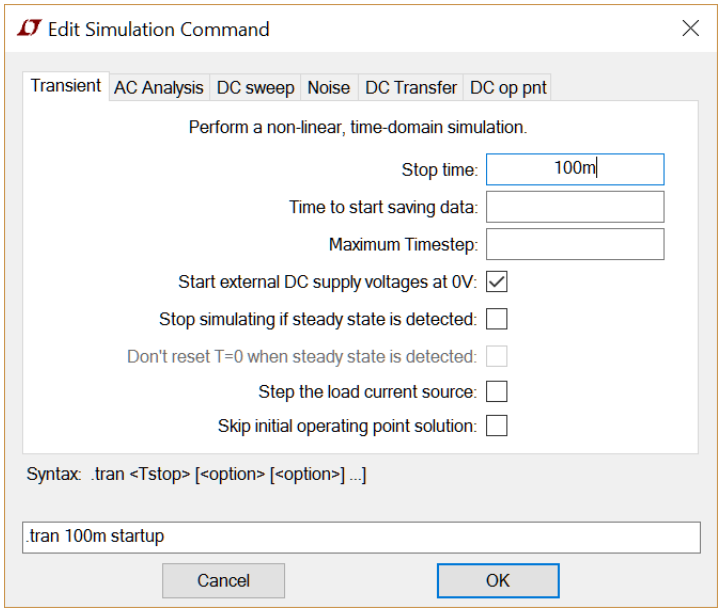

Figure 6: Transient simulation settings.

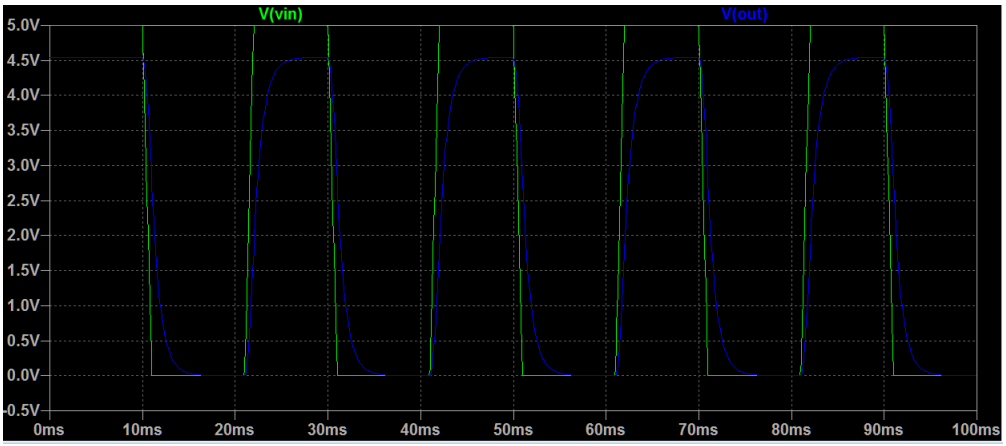

Figure 7: `LTSpice` plotting window.

4.3 .dc

In a DC simulation, we are sweeping a DC source as our x-axis, and measuring that effect on the output. Specify the voltage source by name (**V1** in our example) and then give an increment value. If your output waveform looks too choppy or piece-wise, it is recommended to decrease the increment size. We can sweep up to three voltage sources at once. We can also sweep current sources using this simulation.

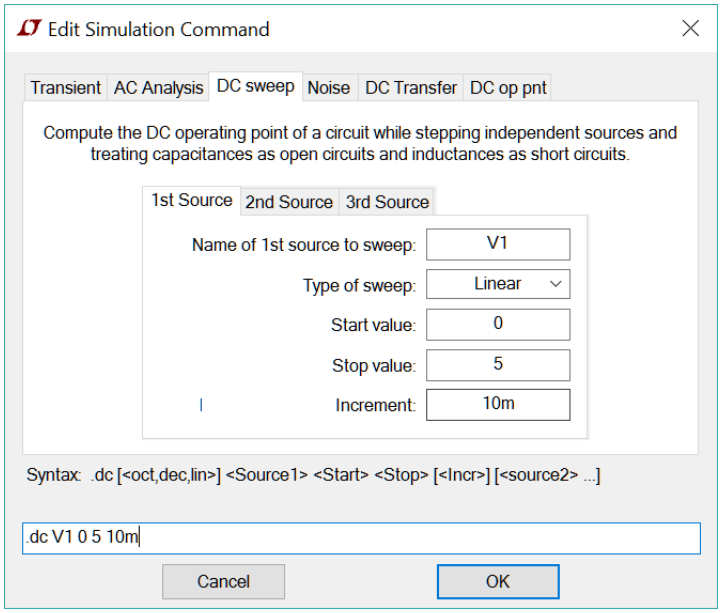

Figure 8: DC simulation settings.

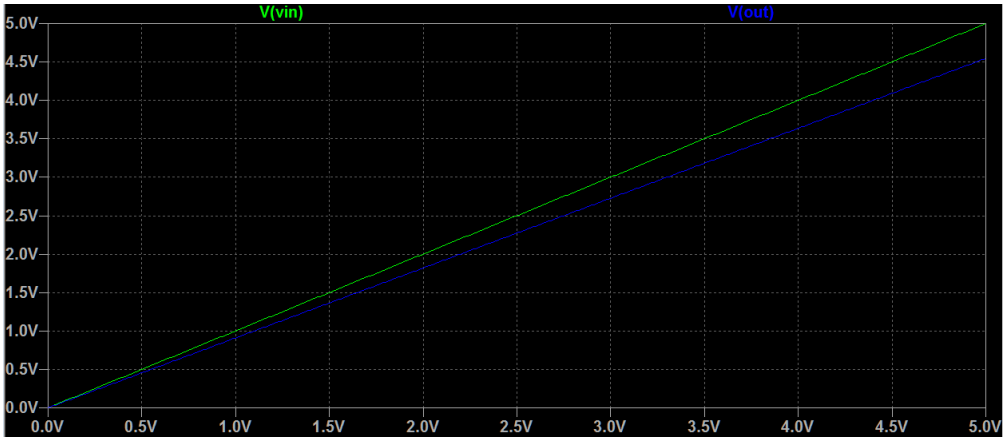

Figure 9: DC simulation example.

4.4 .ac

In an AC simulation, we do a small-signal frequency sweep around a certain DC operating point. Make sure to set the AC small signal parameters in the source, as simply setting a sine wave will not work. We recommend always using `decade` with at least 20 points per decade. By default the magnitude is plotted using a straight line with units on the left-hand axis, and the phase in a dashed line with units on the right-hand axis. You can right-click on either axis to it and hide its corresponding line.

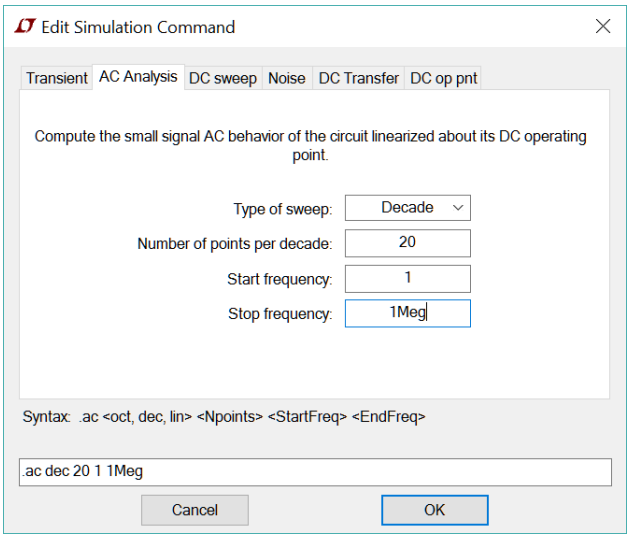

Figure 10: AC simulation settings.

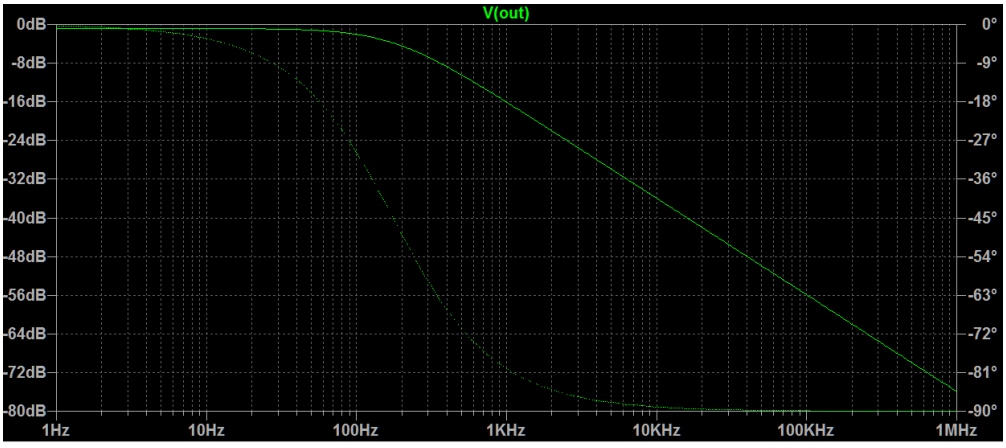

Figure 11: AC simulation example.

4.5 Cursors

The default viewer in `LTSpice` can be somewhat difficult to read, so we use **cursors** to read precise numbers. Add a cursor by left-clicking on the trace name, above the plot. You can add two cursors measure (x_1, y_1), (x_2, y_2), and $(\Delta x, \Delta y)$. To move cursors, use the left and right arrows, or click and drag.

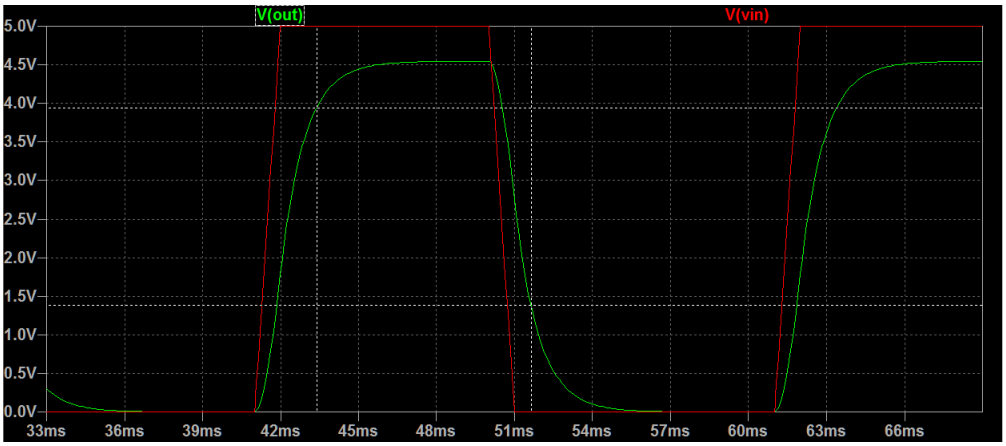

Figure 12: Cursor example in a transient sim.

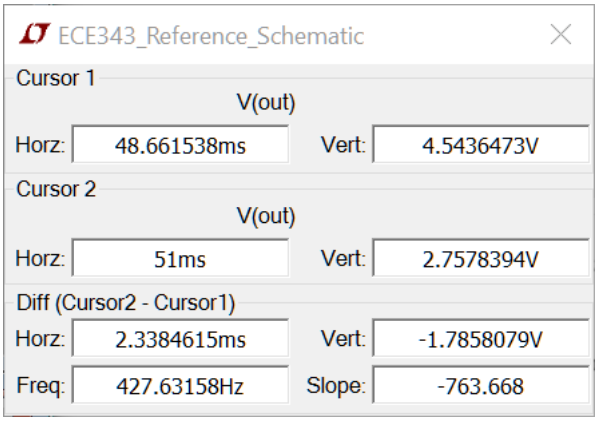

Figure 13: Cursor menu (transient output).

5 Variables + Sweeping

Sometimes we want to see the effect of sweeping a value and checking the output across a range. We do this by parameterizing a value in `SPICE Directive`. To set a parameter as a variable, put the name in curly braces. Then we add a `param` statement in the schematic, specifying any default values. The syntax is:

```
.param [Name]=[Val] [Name2]=[Val2] ...
```

We use a `.step` statement to sweep through the values.

```
.step param [Type] [Variable] [Start] [Stop] [Increment]
```

You can right-click on the `.step` line to set options through a menu as well.

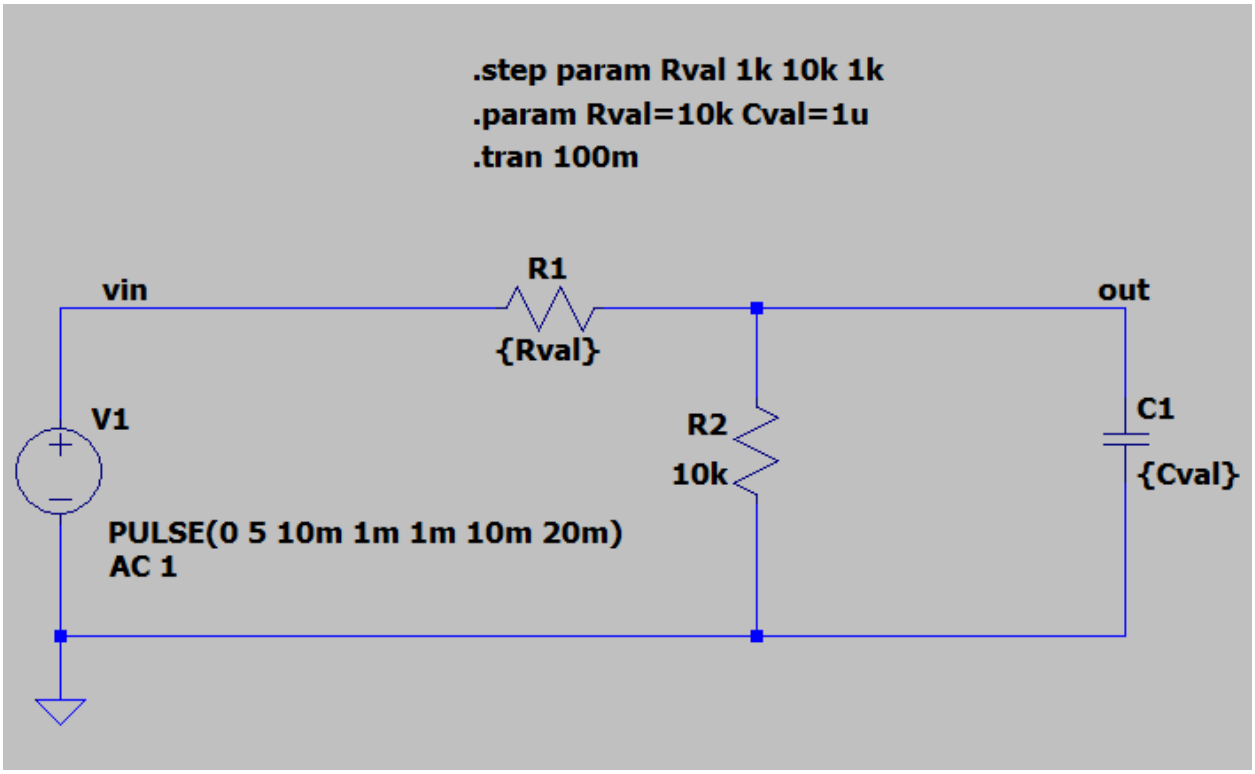

Figure 14: `.step` example.

One slightly annoying fact is the lack of a legend on the plot. In order to figure out which trace corresponds to which run, we have to add a cursor to the plot. Once you add a cursor,

right click on it to have a pop-up describing the run number and value. To switch between runs, use the up and down arrows.

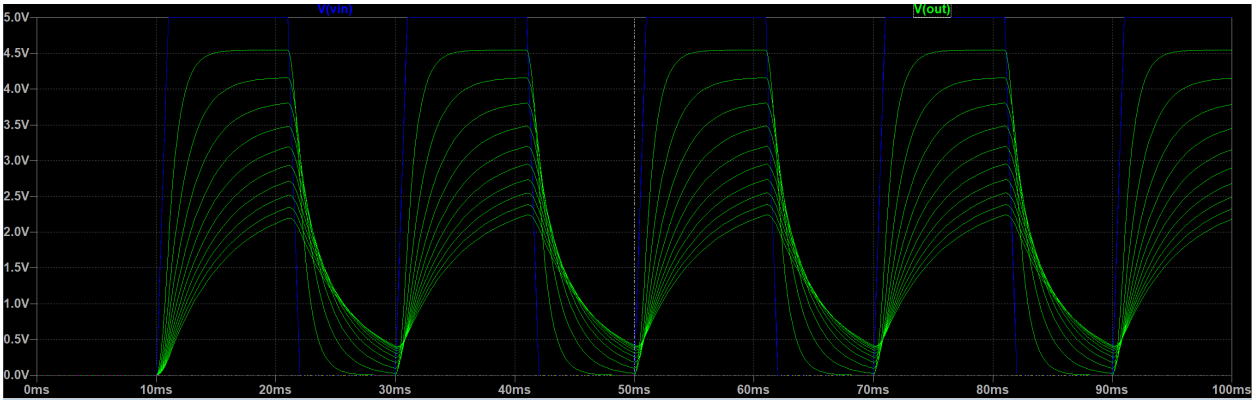

Figure 15: Plotting the `.step` results.

6 Monte Carlo Simulations

Sometimes we don't know the exact values we want to sweep, but instead just a range. It is very typical in circuit design to be given a part with only a certain accuracy, such as 10 kΩ resistor with tolerance ±20%. This type of simulation is called a **Monte Carlo** simulation, where we try to figure out our worst-case output. To do this, we set the parameter using a the `mc` function, as shown in the schematic. The syntax is `mc(mean, var)`. This function will return a random value in the range $[mean * (1 - var) : mean * (1 + var)]$ with a normal (Gaussian) distribution. To run a MC simulation, we use the `.step` command, specifically using the `run` variable. `run` is a reserved keyword for `LTSpice`, which acts as a pseudo-random number generator, so we can deterministically repeat our "random" results.

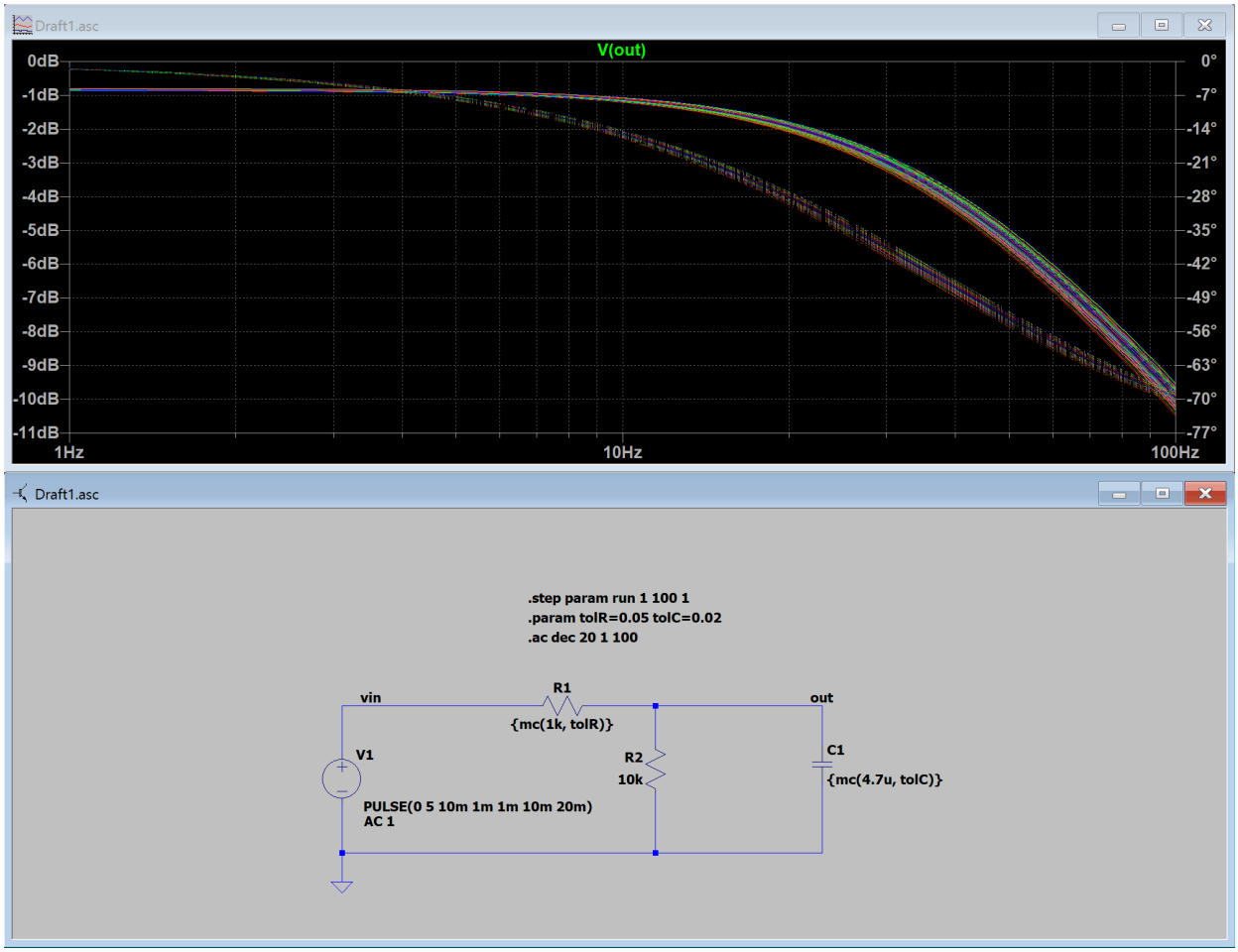

Figure 16: Monte Carlo simulation example

7 Exporting + Plotting

While the `LTSpice` plotter gets the job done, it's not the best thing to include in a report or printout. Instead, we prefer you export the data to `MATLAB` or `Python` and do the plotting there. We'll give example code here to help you do so for some simple examples. First, make your plot in `LTspice`, and go to `File`→`Export data as text`. The following menu will pop up:

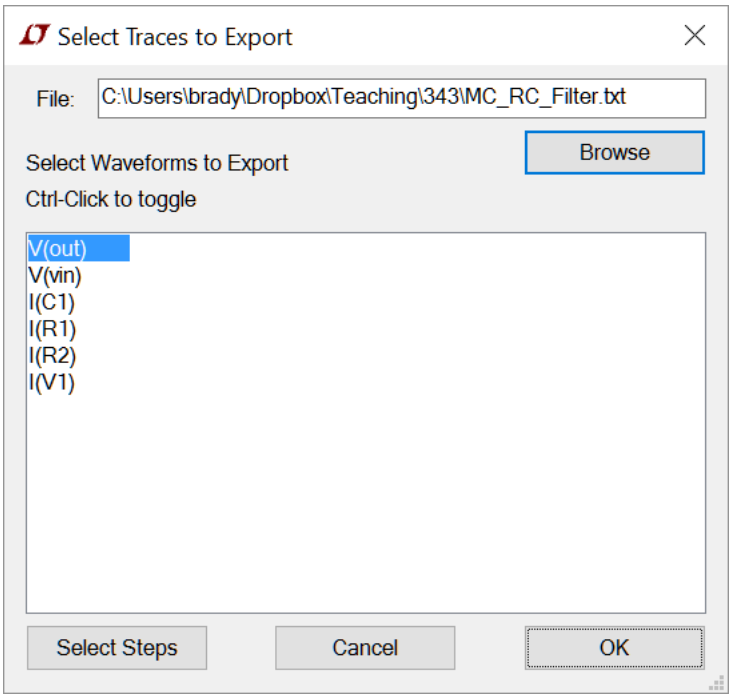

Figure 17: Trace export window.

From here you can choose the output data directory as well as which traces and specific runs you would like to save. By default, it selects whatever is currently plotted in the window.

In Python, use the following script to plot data from `MC_RC_Filter.txt` file:

```
import matplotlib.pyplot as plt

filename = 'MC_RC_Filter.txt'

with open(filename, 'rb') as f:
    labels = f.readline().decode()[:-2].split("\t")
    data = {}

    for line in f.readlines():
        if line[:4] == b'Step': # header for each run
            step_n = int(line.decode().split(' ')[2][4:])
            data[step_n] = {}
            data[step_n]['Freq'] = []
            data[step_n]['Amp'] = []
            data[step_n]['Phase'] = []

        else:
            freq, vout = line.decode('cp1252').split('\t')
            data[step_n]['Freq'] += [float(freq)]
            data[step_n]['Amp'] += [float(vout[1:22])]
            data[step_n]['Phase'] += [float(vout[25:46])]

fig = plt.figure(figsize=(10, 5), dpi=100)

# Frequency vs Amplitude
ax = fig.add_subplot(121)
for step in data.keys():
    ax.plot(data[step]['Freq'], data[step]['Amp'])
ax.set_xlabel(labels[0])
```

```
ax.set_ylabel(labels[1])
ax.set_title('Frequency Vs. Amplitude (dB)')



# Frequency vs Phase
ax = fig.add_subplot(122)
for step in data.keys():
    ax.plot(data[step]['Freq'], data[step]['Phase'])
ax.set_xlabel(labels[0])
ax.set_ylabel(labels[1])
ax.set_title('Frequency Vs. Phase (deg)')

plt.tight_layout()
```