

ECE343 LTspice Reference

University of Illinois, Urbana-Champaign

Contents

1	Introduction	2
2	Keyboard Shortcuts	3
3	Getting Started	4
3.1	Netlists	7
4	Simulations	8
4.1	.op	9
4.2	.tran	10
4.3	.dc	11
4.4	.ac	12
4.5	Cursors	13
5	Variables + Sweeping	14
6	Monte Carlo Simulations	15
7	Exporting + Plotting	16

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 simulation, 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	G - Add a ground port	Ctrl+D - Drag
F5 - Delete an item	X - Add an inductor	Ctrl+F - Find text
F6 - Copy an item	C - Add a capacitor	Ctrl+G - Toggle grid
W - Select the wiring mode	Space - Zoom full	Ctrl+H - Halt simulation
E - Add a net name	Ctrl+W - Move	Ctrl+Z - Undo
R - Add a resistor	Ctrl+E - Mirror	Ctrl+X - Delete
T - Add plain text	Ctrl+R - Rotate	Ctrl+C - Copy
S - Add SPICE text	Ctrl+T - Toggle pins	Ctrl+V - Paste
D - Add a diode	Ctrl+Y - Redo	Ctrl+B - Run simulation
F - Find/add a part	Ctrl+S - Save File	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:

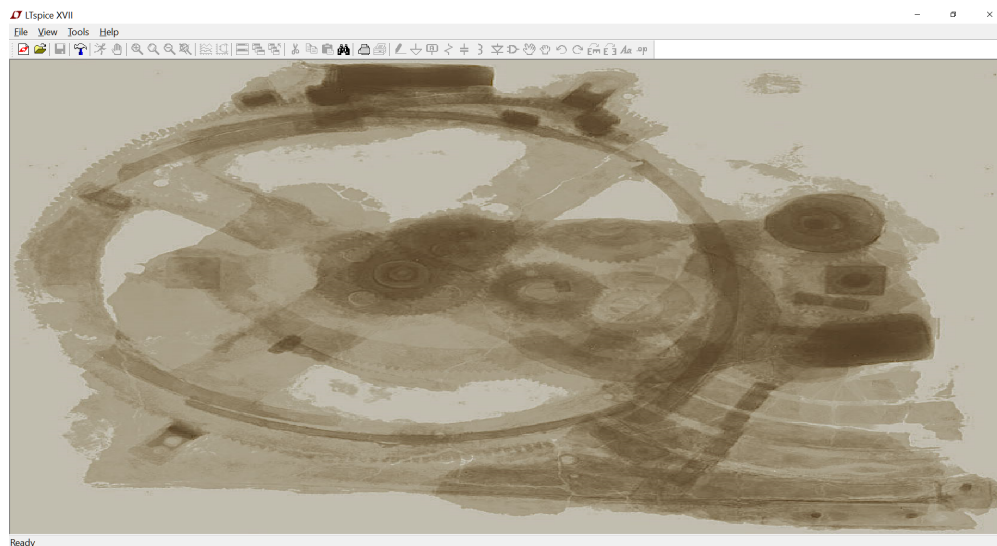


Figure 1: LTspice startup screen. Engineering sketches made out of coffee stains!

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.

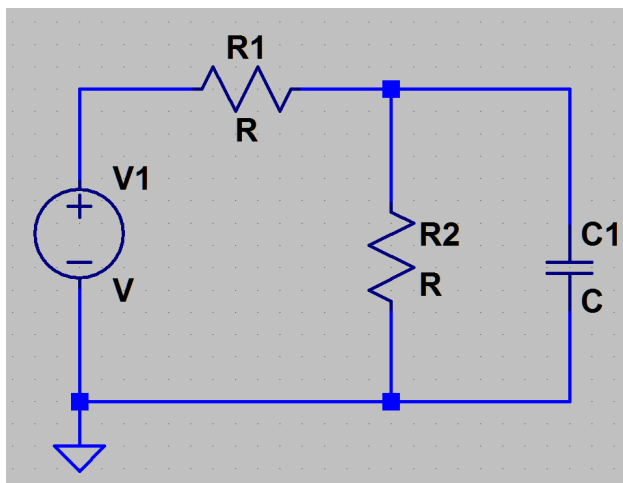


Figure 2: Basic lowpass filter schematic.

You should get the following menu:

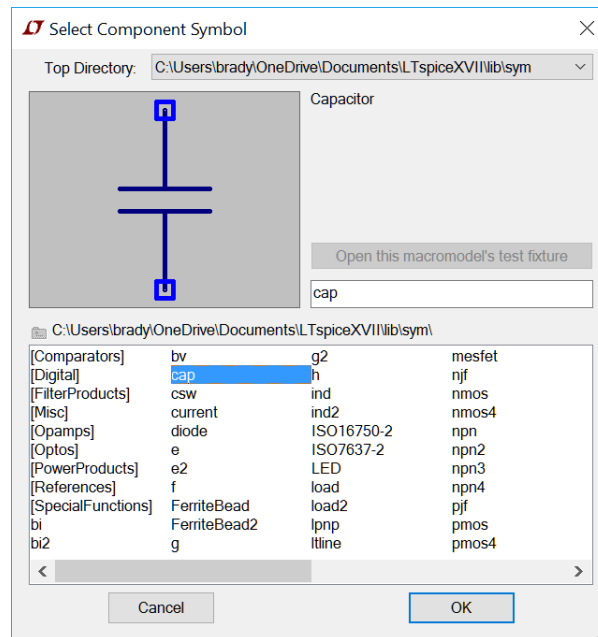


Figure 3: Adding components window.

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! 10m Ω 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 $R1 = 1 \text{ k}\Omega$, $R2 = 10 \text{ k}\Omega$, and $C1 = 47 \mu\text{F}$. Note we can also choose a specific "real-life" model, which will come in handy when using active devices.

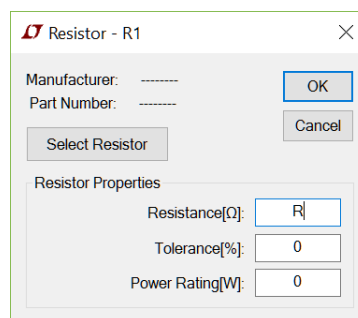


Figure 4: Editing components window

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.

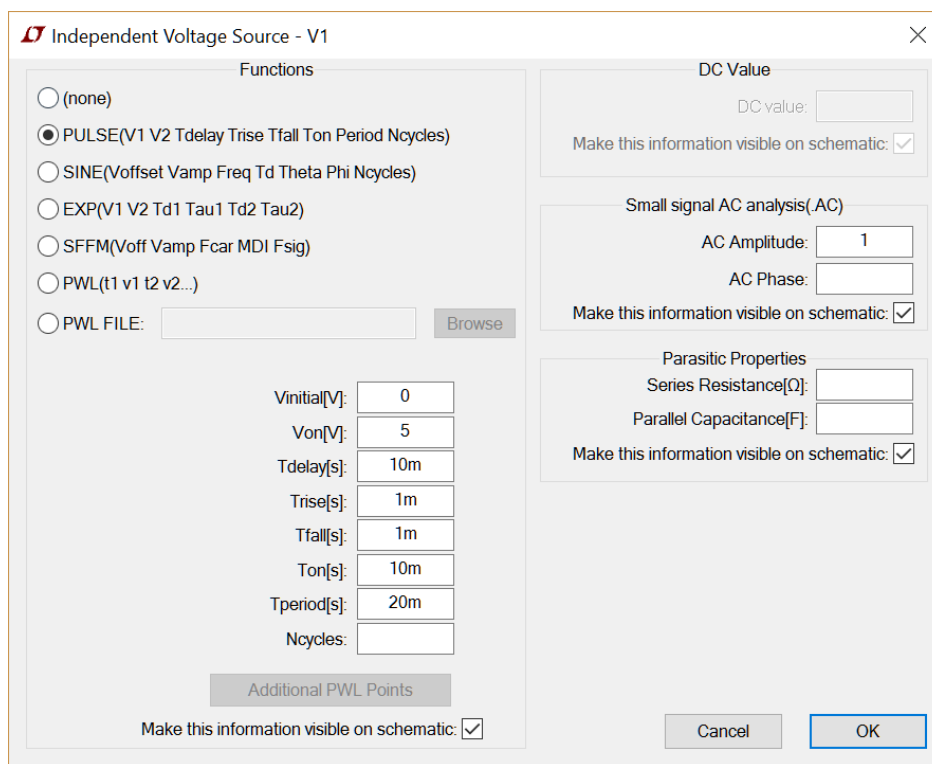


Figure 5: V1 settings.

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:

```
R1 Out N001 1k tol=0 pwr=0
R2 Out 0 10k
C1 Out 0 1u
V1 N001 0 PULSE(5 0 10m 1m 1m 10m 20m) AC 1
```

Each line can be read as: [Device Type] [Device Name] [Connection nodes] [Parameters].

The device type is selected based on its 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. 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.

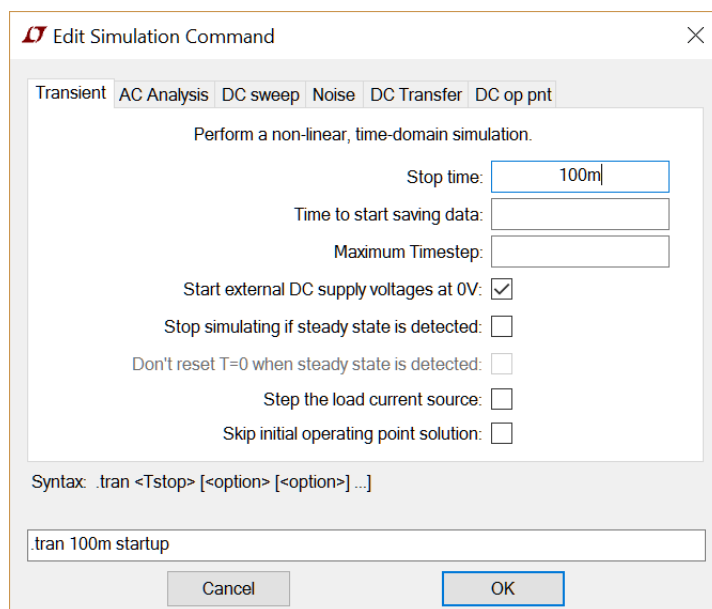


Figure 6: Transient simulation settings.

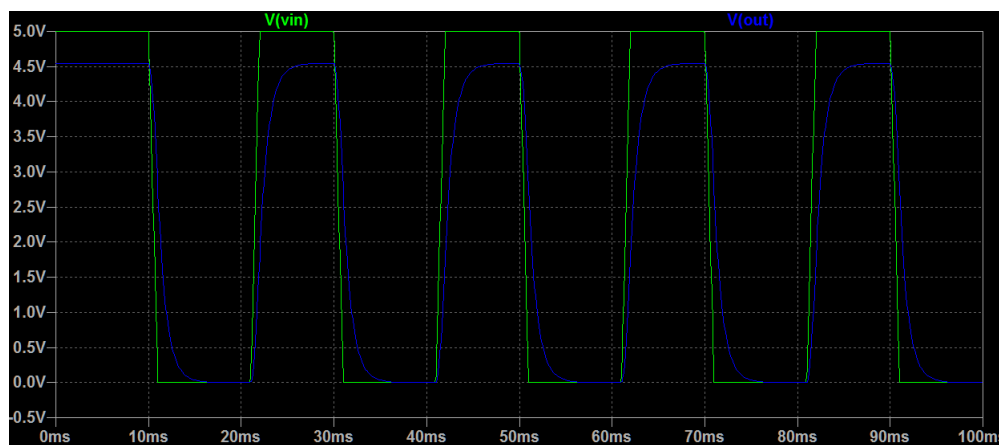


Figure 7: LTspice plotting window.

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.

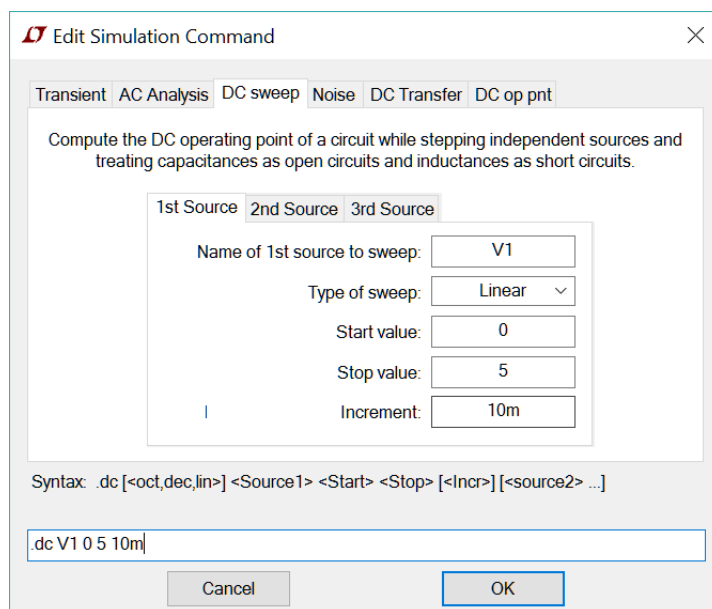


Figure 8: DC simulation settings.

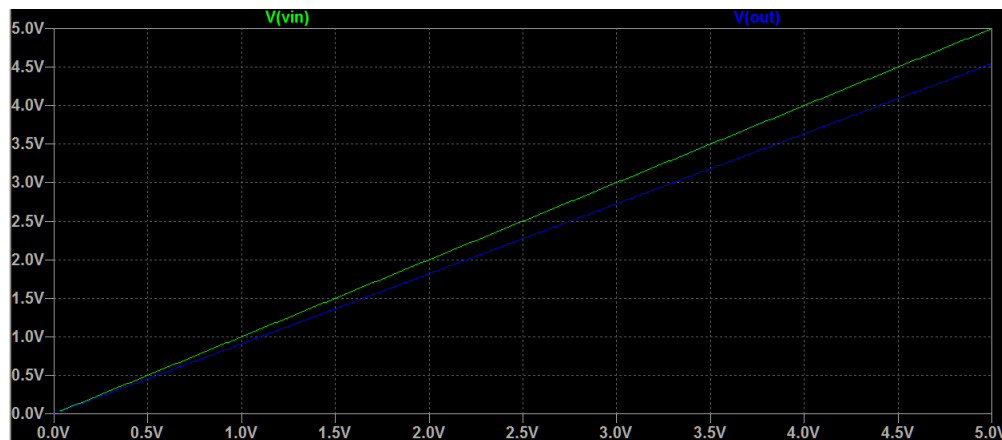


Figure 9: DC simulation example.

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 it's corresponding lines off.

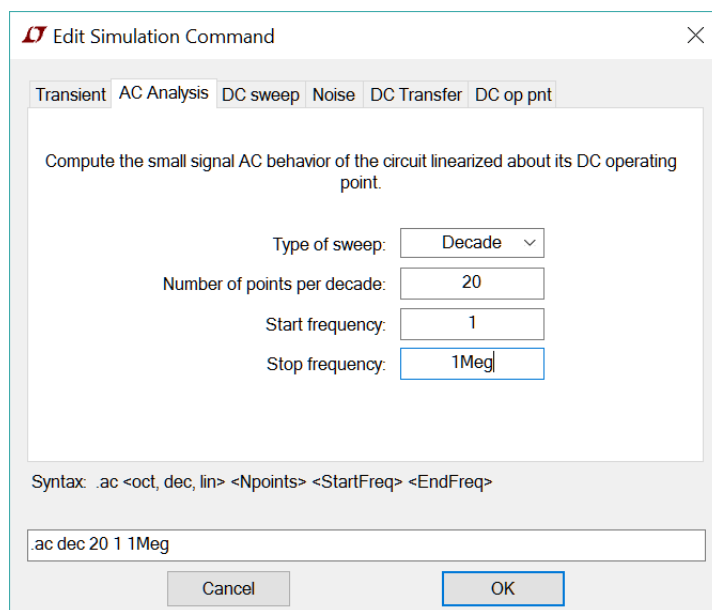


Figure 10: AC simulation settings.



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.

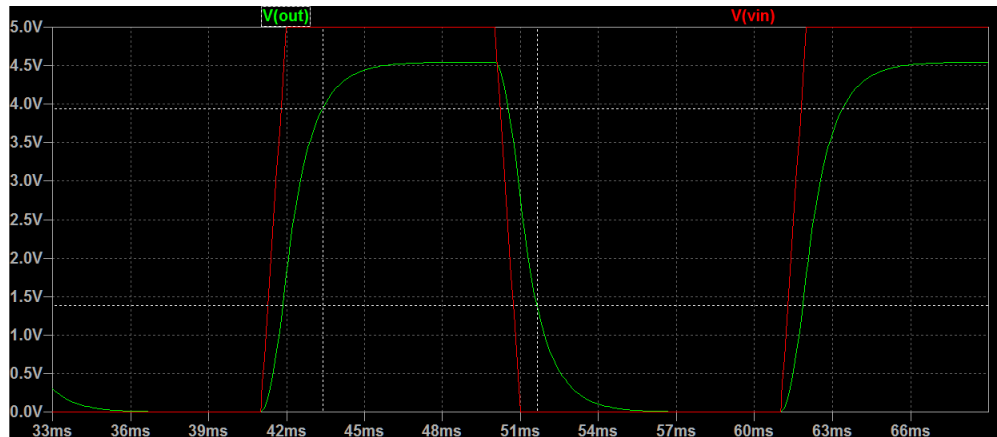


Figure 12: Cursor example in a transient sim.

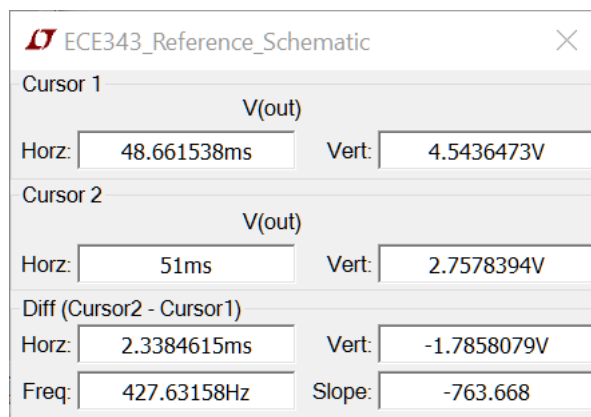


Figure 13: Cursor menu (transient output).

5 Variables + Sweeping

In order to see the effect of sweeping a value and checking the output across a range, we need to set a parameter as a variable by putting the name in curly braces. Then we add a `param` statement in the schematic, specifying any default values and use a `.step` statement to sweep through the values. The syntax is:

```
.param [Name]=[Val1] [Name2]=[Val2] ...
.step param [Type] [Variable] [Start] [Stop] [Increment]
```

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

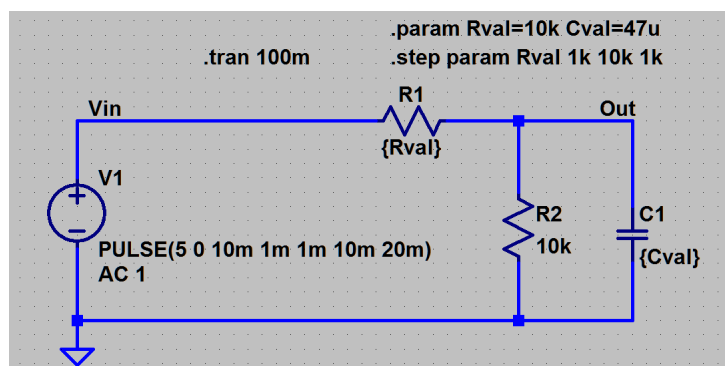


Figure 14: `.step` example.

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.

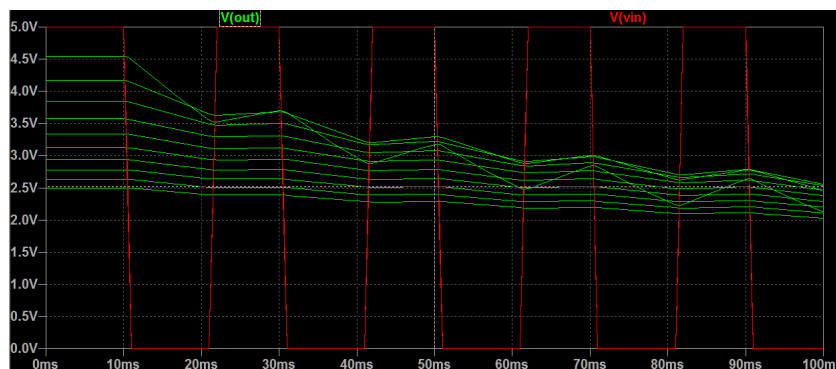


Figure 15: Plotting the `.step` results.

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 a 10 k Ω resistor with tolerance $\pm 20\%$. 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.

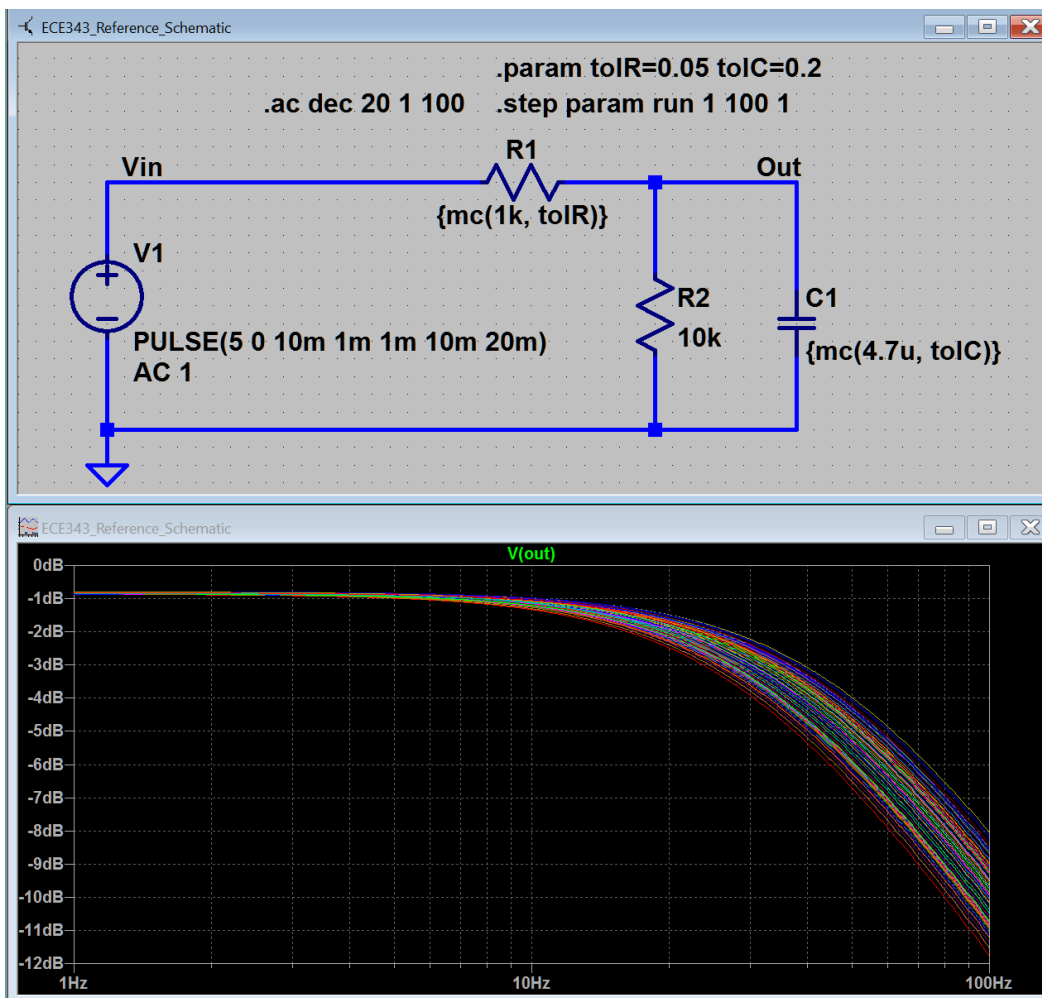


Figure 16: Monte Carlo simulation example

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, export the data to **MATLAB** or **Python** and do the plotting there is preferred. 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:

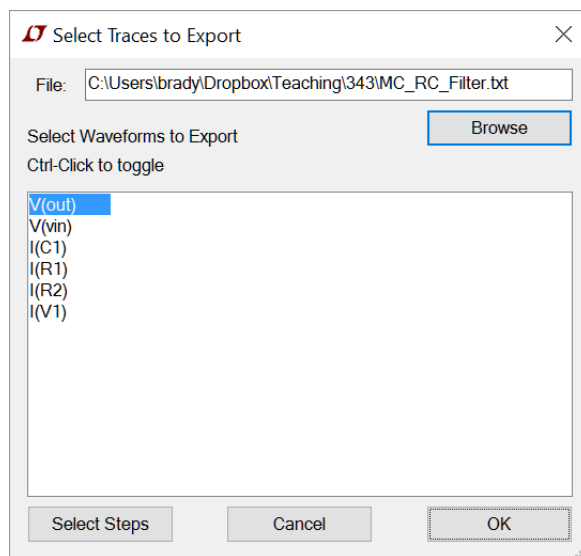
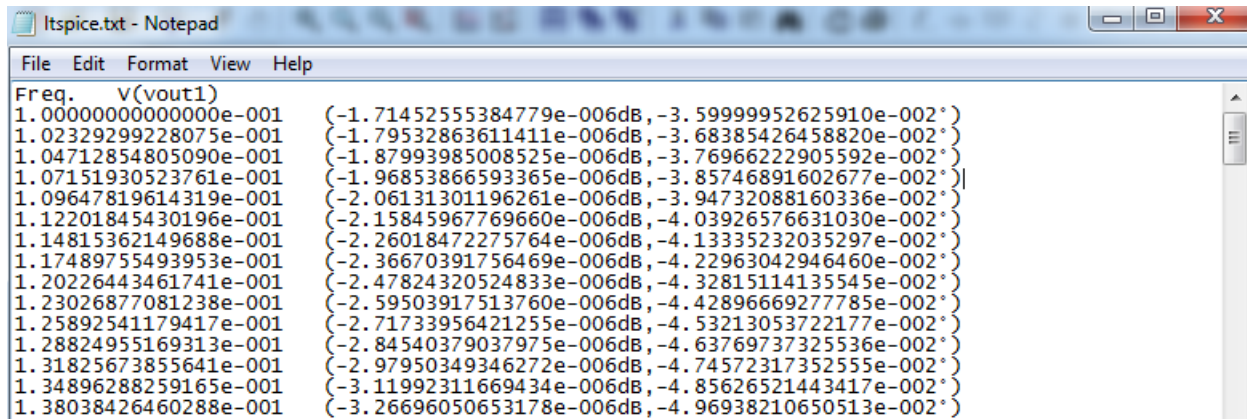


Figure 17: Trace export window.

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. It will be saved as .txt file.

Important step: Open the .txt and remove any text and symbols such as “(”, “o”, “dB”. (They can be easily removed by using the replace function under ”Edit” menu bar in Notepad.) as shown in figure 18 and figure 19 below.

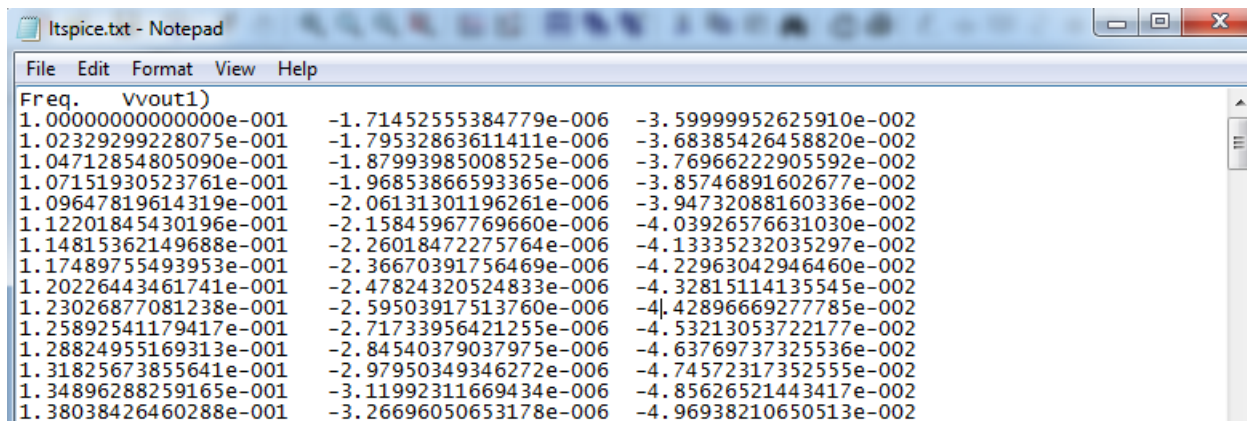


```

Itspice.txt - Notepad
File Edit Format View Help
Freq. V(vout1)
1.000000000000000e-001 (-1.71452555384779e-006dB, -3.59999952625910e-002°)
1.02329299228075e-001 (-1.79532863611411e-006dB, -3.68385426458820e-002°)
1.04712854805090e-001 (-1.87993985008525e-006dB, -3.76966222905592e-002°)
1.07151930523761e-001 (-1.96853866593365e-006dB, -3.85746891602677e-002°)
1.09647819614319e-001 (-2.06131301196261e-006dB, -3.94732088160336e-002°)
1.12201845430196e-001 (-2.15845967769660e-006dB, -4.03926576631030e-002°)
1.14815362149688e-001 (-2.26018472275764e-006dB, -4.13335232035297e-002°)
1.17489755493953e-001 (-2.36670391756469e-006dB, -4.22963042946460e-002°)
1.20226443461741e-001 (-2.47824320524833e-006dB, -4.32815114135545e-002°)
1.23026877081238e-001 (-2.59503917513760e-006dB, -4.42896669277785e-002°)
1.25892541179417e-001 (-2.71733956421255e-006dB, -4.53213053722177e-002°)
1.28824955169313e-001 (-2.84540379037975e-006dB, -4.63769737325536e-002°)
1.31825673855641e-001 (-2.97950349346272e-006dB, -4.74572317352555e-002°)
1.34896288259165e-001 (-3.11992311669434e-006dB, -4.85626521443417e-002°)
1.38038426460288e-001 (-3.26696050653178e-006dB, -4.96938210650513e-002°)

```

Figure 18: Original exported .txt file



```

Itspice.txt - Notepad
File Edit Format View Help
Freq. v(vout1)
1.000000000000000e-001 -1.71452555384779e-006 -3.59999952625910e-002
1.02329299228075e-001 -1.79532863611411e-006 -3.68385426458820e-002
1.04712854805090e-001 -1.87993985008525e-006 -3.76966222905592e-002
1.07151930523761e-001 -1.96853866593365e-006 -3.85746891602677e-002
1.09647819614319e-001 -2.06131301196261e-006 -3.94732088160336e-002
1.12201845430196e-001 -2.15845967769660e-006 -4.03926576631030e-002
1.14815362149688e-001 -2.26018472275764e-006 -4.13335232035297e-002
1.17489755493953e-001 -2.36670391756469e-006 -4.22963042946460e-002
1.20226443461741e-001 -2.47824320524833e-006 -4.32815114135545e-002
1.23026877081238e-001 -2.59503917513760e-006 -4.42896669277785e-002
1.25892541179417e-001 -2.71733956421255e-006 -4.53213053722177e-002
1.28824955169313e-001 -2.84540379037975e-006 -4.63769737325536e-002
1.31825673855641e-001 -2.97950349346272e-006 -4.74572317352555e-002
1.34896288259165e-001 -3.11992311669434e-006 -4.85626521443417e-002
1.38038426460288e-001 -3.26696050653178e-006 -4.96938210650513e-002

```

Figure 19: Modified exported .txt file

After making the modification in the textfile, the .txt file is ready to be used in MATLAB using the sample code provided in the website.

Here's the MATLAB example:

```
%% Formatting
colordef white;
% Plot with default font size 16+
% Linewidth of 2 or 3
set(0,'defaultAxesFontName', 'Arial');
set(0,'defaultTextFontName', 'Arial');
set(0,'defaultAxesFontWeight', 'Bold');
set(0,'defaultTextFontWeight', 'Bold');
set(0, 'DefaultTextFontSize', 18) ;
set(0, 'DefaultAxesFontSize', 16) ;
set(0, 'DefaultLineLineWidth', 3.5) ;
set(0, 'DefaultStairLineWidth', 3.5) ;
set(0, 'DefaultAxesLineWidth', 2.5) ;
set(0, 'DefaultPatchLineWidth', 2) ;
set(0, 'DefaultAxesXGrid', 'on') ;
set(0, 'DefaultAxesYGrid', 'on') ;
set(0, 'DefaultAxesBox', 'on') ;
set(0, 'DefaultFigurePosition', [200 175 1000 600]);

%% Plotting
filename = 'directory\ltspice.txt';
data = dlmread(filename, '', 1, 0);
freq = data(:,1);
complex = data(:,2)+i*data(:,3); %convert data into complex + j*real
magvv = abs(complex); % magnitude of AC response in V/V
magdB = 20*log10(magvv); % magnitude of AC response in dB
phase = angle(complex)*180/pi;
subplot(2,1,1)
semilogx(freq,magdB)
title('AC Response of 1st-order RC filter')
ylabel('Magnitude (dB)')
subplot(2,1,2)
semilogx(freq,phase)
xlabel('Frequency (Hz)')
ylabel('Phase (deg)')
```

In Python, there's a couple ways of doing the formatting. The simplest way is to go to your `.matplotlib` folder and [add a style](#). Save this as “`ece343.mplstyle`”.

```
# ECE@Illinois Configs
# Created on Tue Dec 6 2016
# Fonts
font.family : sans-serif
font.sans-serif : Arial
font.weight : bold
font.size : 18.0

# Axis Tick Settings
xtick.major.size : 16.0
ytick.major.size : 16.0
xtick.major.width : 4.0
ytick.major.width : 4.0
xtick.major.pad : 12.0
ytick.major.pad : 12.0
xtick.labelsize : 22.0
ytick.labelsize : 22.0
xtick.color : k
ytick.color : k

# Marker and Line Settings
legend.numpoints : 1.0
markers.fillstyle : full
lines.markeredgewidth : 2.0
lines.linewidth : 3.5
lines.markersize : 20.0
patch.linewidth : 3.5
grid.linewidth : 4.0
grid.alpha : 1.0
grid.color : #BFBFBF

axes.titlesize : 28.0
axes.titleweight: bold
```

```
axes.labelsize : 24.0
axes.labelweight : bold
axes.linewidth : 4.0
axes.grid : True
axes.edgecolor : k
axes.facecolor : w
```

```
figure.figsize : 10, 6
figure.facecolor : w
```

Then the plotting is quite similar to MATLAB: (**NOTE**: the plot settings below will not give the same plots in MATLAB version. The code provided below is purely for guidelines)

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.style.use('ece343')

filename = 'RC_Filter.txt'
data = np.loadtxt(filename, delimiter='\t', skiprows=1)

time = data[:, 0]
vin = data[:, 1]
vout= data[:, 2]

plt.plot(time, vin)
plt.plot(time, vout)
plt.legend(['Vin', 'Vout'])
plt.show()

plt.xlabel('Time [ms]')
plt.ylabel('Voltage [V]')
plt.tight_layout()
```