Getting Started with LTspice in ECE 25500

Rayane G. Chatrieux
Purdue University

Created: August 2019

Note: This tutorial is mainly geared towards Windows users and Linux users accessing LTspice through Wine. If you are a MAC user, check out the links provided on the course web-page (https://nanohub.org/groups/class_ece255) to get started. The link to the installation executable is the same for all of these OS.

Contents

1 Introduction

One of the learning objectives of ECE 25500 is that the student needs to be able to use CAD tools to analyze microelectronic circuits. There is a myriad of such tools allowing users to simulate circuit behavior. SPICE, the tool of choice for this course, is by far one of the most popular and simplest ones to use out there. Yet, you will most likely never need to go beyond its capabilities for any of your hobbyist activities (industry is another story of course). SPICE by itself is usually a simple command-line executable which reads inputs from files and completes its tasks without much interaction with the user. In ECE 25500, we will use a SPICE package called LTspice, which provides, among many other things, a graphical interface to SPICE and a waveform viewer. Throughout the course, you will learn about using SPICE through LTspice.

See section 2 for installation instructions. In section 3, I provide a little bit of background on what SPICE is and why it is such a popular CAD tool. In section 4, I show you how to build a circuit in the LTspice schematic editor. In section 5, I list the different types of simulations you'll need to know about, and how to run them. In section 6, I introduce four very important SPICE directives which you will need to successfully design your circuits in this course.

Note: (Side notes such as this one will be enclosed in a gray box) I try to improve the quality of this tutorial every semester. Part of your first LTspice assignment is to read it in its entirety and provide feedback. Please don't hesitate to be as critical as you wish.

2 Installation

LTspice is a software package owned and maintained by Linear Technology, which is now part of Analog Devices. You can get the installation executable from their website. Once the installation completes and you have made sure that the program starts correctly, take a minute to look through the installation directory. In particular, you'll notice the presence of an `examples` and a `lib` folder. In the `examples` folder, several circuit schematic examples are provided to you. In the `lib` folder, LTspice stores, among others, model parameters, sub-circuit definitions, and schematic symbols for all devices accessible to you out-of-the-box through the schematic editor. Of particular importance are the library files contained in the directory `lib/cmp`, which define the model parameters of many components. We'll have more to say about these definitions later in this tutorial.

3 What are SPICE and LTspice?

Let me just quickly give you some information about what SPICE and LTspice are, how they work, and how they are used. This information is arguably not relevant to ECE 25500. However, you will need it later on as you learn to use more and more CAD tools and you are faced with the inevitable question of which to use to solve your particular problems.

3.1 Historical Context

For several decades until the 1960s, prototyping circuits on a breadboard or a perf board was, for the most part, sufficient to thoroughly test a circuit before mass production. In the late 1950s, the **planar process** was invented, which led to a massive development of **integrated circuit (IC)** technology. In fact, the IC quickly became the main form of implementation of digital logic on a circuit. However, ICs could only be tested after they had been produced, a costly operation even at the time. As a result, in the early days of the IC industry, companies had to spend much more money on product development than they had to before. IC vendors started relying more and more on software circuit simulation rather than physical testing to both speed up, and reduce the cost of development. Unfortunately, this transition was operated for the most part internally, with each vendor developing its own proprietary simulation software tool and using it only for its own products. It is in this industrial context that **SPICE** is invented in 1973 at UC Berkeley by Laurence Nagel. SPICE, which stands for **Simulation Program with Integrated Circuit Emphasis**, is one of the first simulation tools distributed as a public domain software, meaning that its source code was made freely available to anyone. Even in its early life, SPICE had just enough analyses and models, robustness, flexibility and efficiency to become the most popular circuit simulation software in industry. Most IC vendors of the time started to use the SPICE source code as the backbone of their own customized software tool. To this day, CAD tools still use the SPICE algorithm in derivative products such as HSPICE and PSPICE. For example, Cadence Virtuoso, the leading IC design software in industry, uses a circuit simulation program called Spectre, a derivative of SPICE.

3.2 The SPICE Algorithm

The algorithm on which SPICE runs is beautiful in its simplicity. A rough outline of the works of the algorithm is shown in **Fig.1** below. Here, SPICE is actually performing what is called a transient analysis (more on this in section 5.4). This type of analysis seeks to simulate circuit behavior over a period of time (real time). Notice the presence of two loops. The outer loop simply continues until the the end of the simulation time is reached. For example, you may want to simulate a circuit over 10 ms. SPICE will compute an appropriate time step and increment the simulation time until 10 ms are reached. The inner loop does something far more interesting, and is the bread and butter of SPICE. In essence, to simulate circuit behavior over time, SPICE actually considers a sequence of snapshots of the circuit's state. The outer loop, which increments time, goes from one snapshot to the next. SPICE solves for the various circuit quantities (voltages and currents) in a snapshot by solving a set of linear nodal equations (those same ones you saw in ECE 20100) put into matrix form. In order to achieve high execution speeds, SPICE uses an iterative approximate solution method to solve these matrix equations. That is, instead of finding an exact solution to all circuit parameters, SPICE computes these in a sequence of approximate solutions in the hope that this sequence converges to the exact solution. Although this method is incredibly efficient, it is possible for a solution to not converge (usually when the circuit contains

discontinuous waveforms or high impedance nodes). However, these convergence issues will not have to be dealt with in ECE 25500.

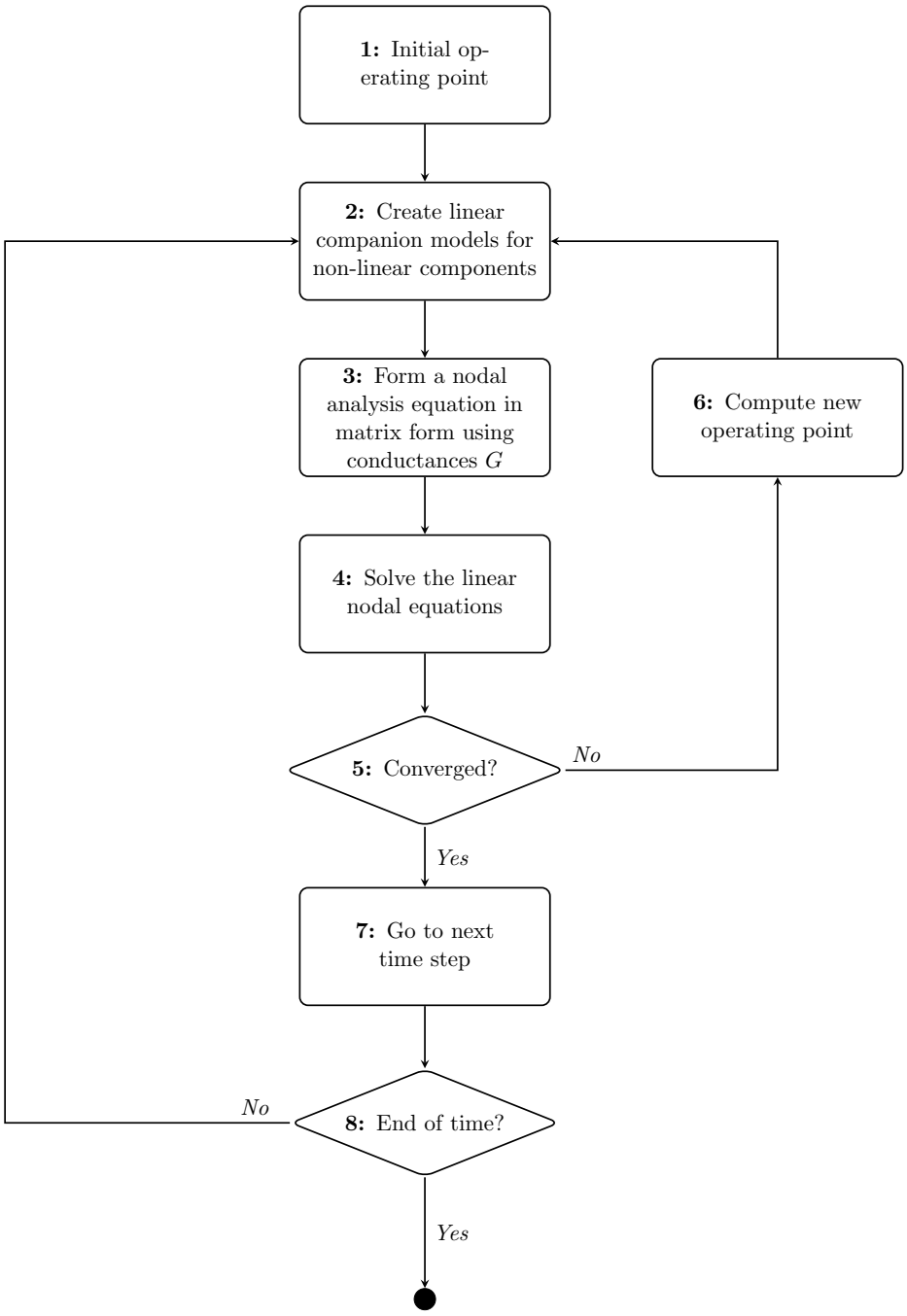

Fig.1 - The SPICE Algorithm

3.3 Device Models

SPICE is shipped with models for several common devices, such as diodes and MOSFETs. These models are actually the parametrized equations that govern the operation of the device (e.g. the i-v characteristic of a resistor). These equations are specific enough that one can quickly simulate circuits. However, they are general enough that one can customize the equations to better approximate a certain device. This customization is possible through the device parameters. For example, you do not need to tell SPICE what a resistor is, but you do need to specify its resistance. We will call the equations SPICE stores internally **Device Models**, and we will call the customizable quantities in these equations **Parameters**. In ECE 25500 (and in your entire career most likely unless you choose to go into devices), you will only ever need to specify parameters to a device.

> **Note:** It is, of course, possible to define your own device. This requires you to expand the source code of SPICE (efficiency is key!). This is not such a costly or difficult task for companies, since the entirety of the SPICE source code is available to the public. In fact, this ability to expand the source code to fit one's needs is one of the reasons behind SPICE's popularity.

3.4 Netlists

Originally, SPICE was just a command-line executable that ran on pre-defined input circuits. That is, the circuit needs to have been defined and completely specified in some way before SPICE is run. Similar to how most computer programs take their inputs, the circuit is defined in a file. That file is referred to as a circuit **netlist**, and could look something like **Fig.2**. Roughly speaking, each line in the netlist describes one component of the circuit; that is, its type (e.g. a resistor), its name, its connectivity (the name of the nodes it connects to), and its parameters (e.g. resistance). Other lines, those which start with a period, are what are called **directives** (see section 6). Their purposes are various. In general, these statements serve the purpose of controlling SPICE's behavior on the circuit before, during and after a simulation is run. In the last section of this tutorial, we'll have a look at some useful directives.

```
R1 out N001 1e3
C1 out 0 1e-6
L1 out 0 1e-3
V1 N001 0 AC 10 0
.ac dec 20 1 1e6
.backanno
.end
```

Fig.2 - An Example Netlist Obtained from LTspice

Netlists are still used to describe circuits today. However, it has become less and less necessary to have to write them yourself. Early on, companies introduced the **schematic editor**, which is a graphical interface to circuit netlist generation. With a schematic editor, one can place and connect components using a mouse, and can visualize the created circuit on the screen directly. The netlist

is then generated in the background before a simulation is run. In section 4, I show you how to use the schematic editor provided by LTspice.

3.5 LTspice

You can think of LTspice as a wrapper for SPICE. LTspice, through its schematic editor, gives you a way of creating circuits graphically. LTspice provides numerous graphical control panels to customize circuit components. LTspice provides a waveform viewer, which is certainly the most natural way of visualizing simulation results. In general, LTspice makes using SPICE a much easier and accessible task.

3.5.1 Device Parameter Models

There is one more very important thing LTspice provides us with. That is, a library of device parameter models. I mentioned in section 3.3 that SPICE stores device models internally, and all that you have to do as the user is specify the values of specific parameters. However, specifying a list of parameters for a device is actually not so easy most of the time. For example, modern MOSFET models contain about 200 customizable parameters, most of which are empirical in nature, unintuitive, and one should really not have to bother with them. Fortunately, LTspice already has a bank of pre-defined device parameter models, which are simply lists of parameter values which, together, approximate the behavior of some realistic device. In industry, it is actually the device manufacturer (e.g. TSMC) that provides these device parameter models to its clients.

4 Creating a Circuit in the Schematic Editor

To start with, let's look at how to use the schematic editor to create circuit netlists. I'll cover how to place components, create connections between components, name components, label nets, and assign parameter values to components. I will do this by creating the circuit shown in **Fig.3** as an example.

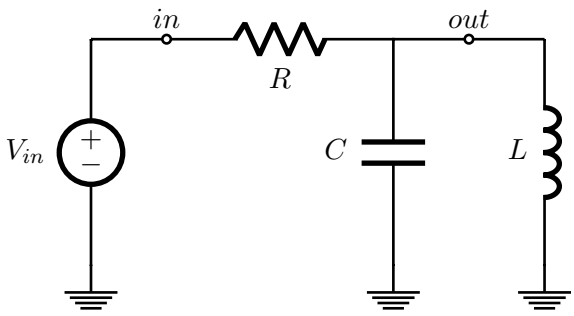

Fig.3 - Example Circuit.

4.1 The Schematic Editor

The first thing to do after launching LTspice is to start a new schematic by clicking on **New Schematic** in the toolbar. The **schematic editor** window appears (see **Fig.4**). This blank canvas is where you'll place components and directives to define a circuit. As you can see in **Fig.4**, a grid should be visible. If it is not, go to **View** and make sure **Show Grid** is checked. This grid facilitates the process of placing and connecting components together.

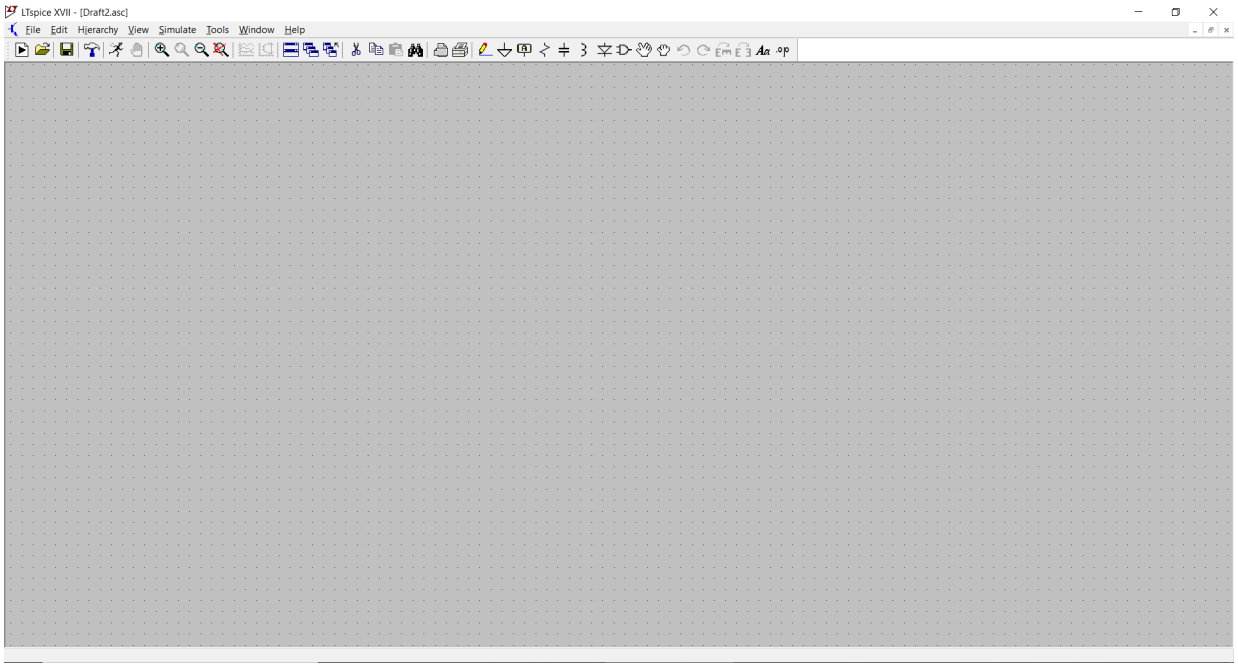

Fig.4

4.1.1 The Toolbar

When you clicked on **New Schematic** and the schematic editor appeared, you should have noticed the toolbar that is now at the top of the window. We will be using commands from this toolbar frequently.

4.1.2 Manipulating the Canvas

To move the canvas around, left click any point on the grid to grab the canvas, and drag (keeping the mouse button pressed) the cursor in any direction. The canvas will follow the cursor. You can also zoom in and out of the canvas using the mouse scroll. In the toolbar, you have some other zooming options. The **Zoom Full Extents** command recenters and resizes the canvas such that every placed component is visible. This is quite useful for large schematics.

4.2 Placing Components

We now start creating the circuit. Refer back to **Fig.3**. This process evidently begins with placing the required components.

4.2.1 Placing a Resistor, an Inductor or a Capacitor

Let's start with the capacitor. You first select the capacitor by clicking on **Capacitor** in the toolbar (alternatively, press **C** on you keyboard). Once selected, you can place instances of capacitors on the canvas (as many as you want). Now, click on **Inductor** (or press **L** on your keyboard) and place the inductor on the canvas to the right of the capacitor. Finally, click **Resistor** (or press **R** on your keyboard) to place the resistor. Before placing the resistor, press **Ctrl + R** to rotate the symbol. By now, you'll have something that looks like **Fig.5**.

> **Note:** LTspice automatically assigns names to the components you place on the canvas. Once you have placed the RLC components, have a look at the generated netlist (click on **View** → **SPICE Netlist**).

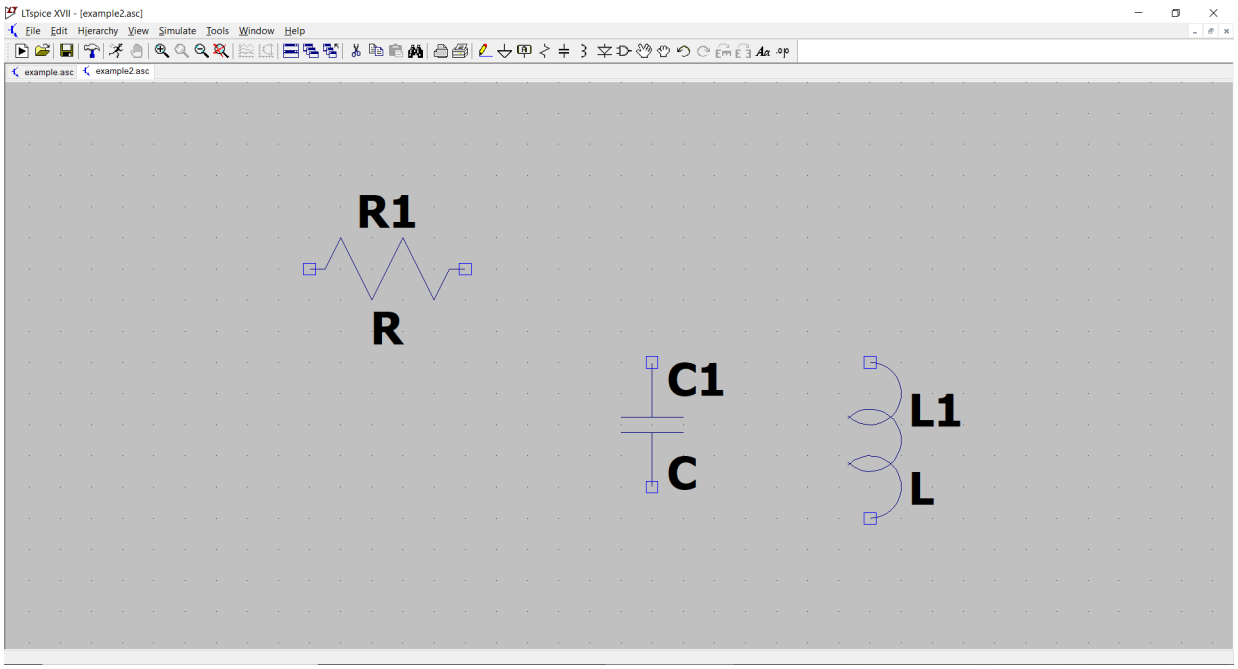

Fig.5

4.2.2 Searching for a Component

Components such as resistors, capacitors and inductors are common enough that they have their own dedicated icons in the toolbar. Most components available in LTspice however, do not. Instead we must manually search for them. Voltage sources are an example of such a component, as there is no icon dedicated to them. We must look for it in components library. Left click on **Components** (in the toolbar). In the search bar, start typing in `voltage` and click **Ok** once you have found the component. You should have the circuit shown in **Fig.6**.

> **Note:** The voltage source in LTspice represents a generic voltage source that can be set to output several types of waveforms, including DC, sinusoidal waveforms, pulse waveforms, and arbitrary waveforms (this requires a setup file though). In ECE 25500, we will be only interested in DC and sinusoidal input waveforms.

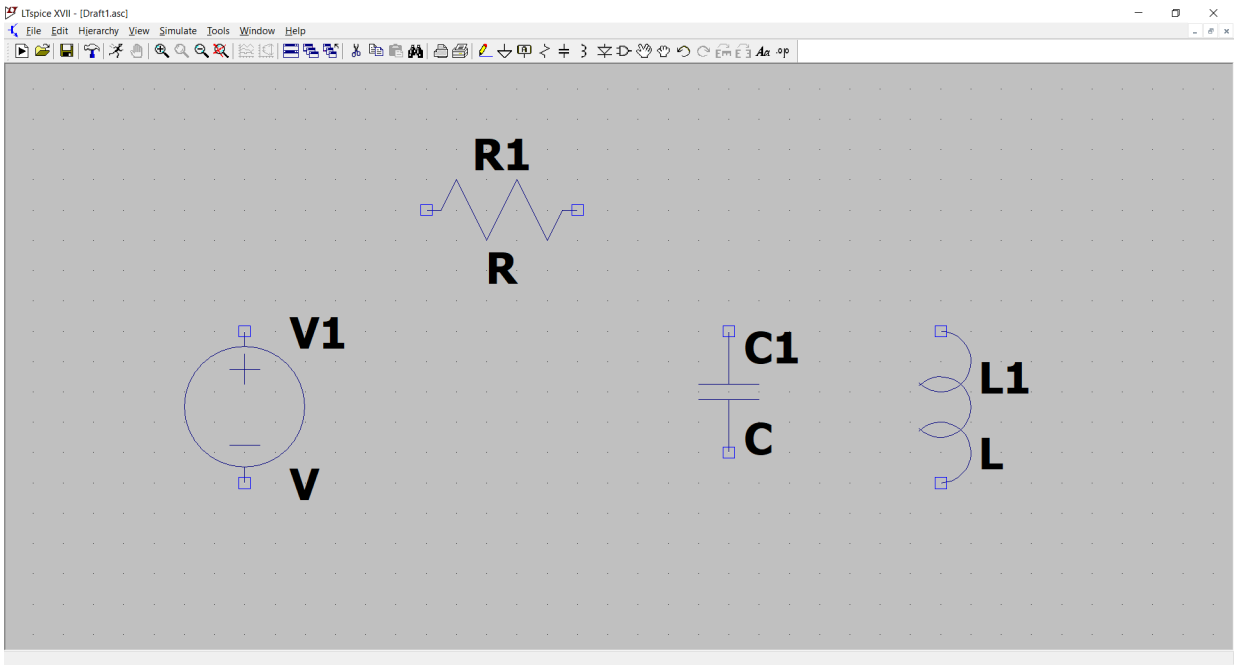

Fig.6

4.2.3 Voltage Reference (Ground)

The last component that is missing from our circuit is the voltage reference. Left click on the **Ground** icon (or press **G** on your keyboard). Place three references below the source, the capacitor and the inductor.

> **Note:** Any circuit you create in LTspice must contain a voltage reference. SPICE assigns the voltage at this node to be 0, and solves for the other nodal voltages relative to the reference.

4.2.4 Moving Components Around

Once the components are placed on the canvas, you still have the ability to change their position or even rotate them. Left click on the **Drag** icon in the toolbar, and left click on a component you want to re-position. You can now move the component around. You can also rotate it by clicking on **Rotate** while the component is selected, or by pressing **Ctrl + R** on your keyboard. Finally, you can mirror components across the vertical by clicking on **Mirror**, or by pressing **Ctrl + E**, while the component is selected. Although rotating and mirroring a component has, of course, no effect on circuit behavior, it allows for clean and readable circuits, and most importantly, it allows us to clearly see the connections.

4.3 Connecting Components

To electrically connect the circuit components together, left click on the **wire** icon. Alternatively, you can press **F3**. Once you have connected all of the components together, you should have the

circuit shown in **Fig.7**. Have another look at the produced netlist, and note that the electrical connections have indeed been made.

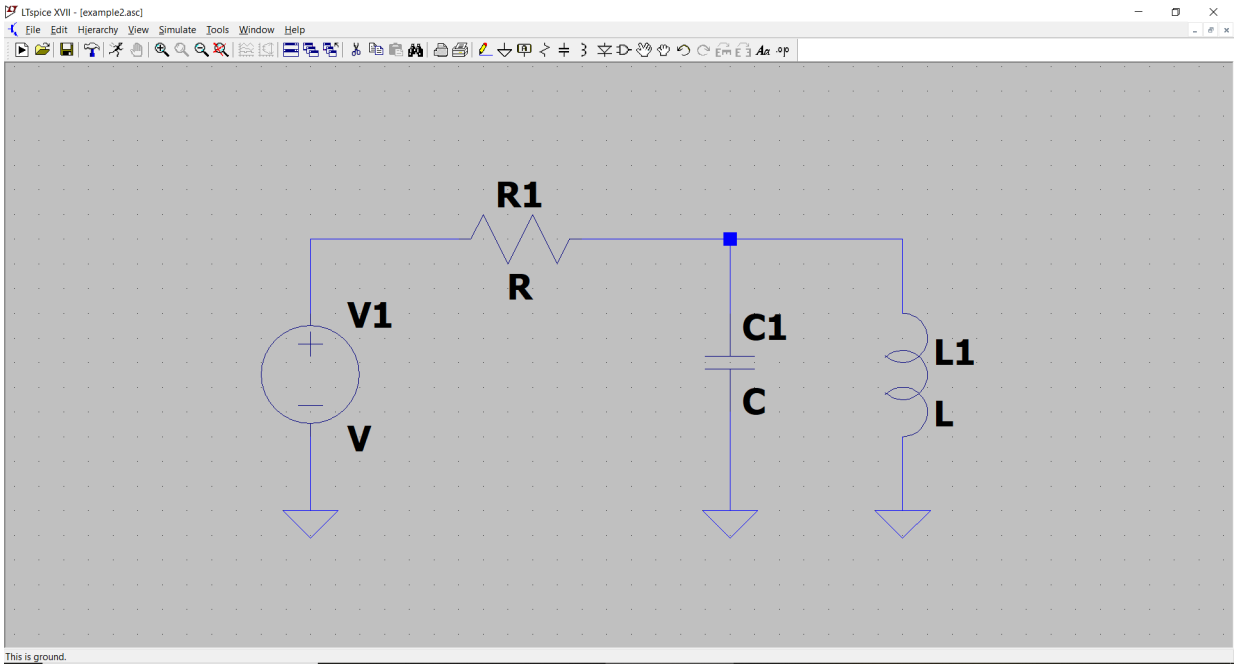

Fig.7

4.4 Assigning Parameter Values

We now assign parameter values to the placed components. By parameters, I mean voltage, resistance, capacitance and inductance in this case. The easy way to do this in LTspice is to simply right click on a component. For example, right click on the resistor. The window shown in **Fig.8** appears. In this window, LTspice has extracted for us the three most important parameters concerning resistors (although we'll only care about resistance in ECE 25500), and provided us with a simple way of assigning values to these. In the **Resistance** field, enter `1e3`. LTspice will understand this value to mean 1×10^3 Ω. Leave the other two fields blank. Let's assign a value of 1 µF to the capacitor. Again, right click on the capacitor symbol on your schematic and enter `1e-6` in the **Capacitance** field. Finally, enter `1e-3` for the inductor. Once that's done, your circuit will look like that shown in **Fig.9**.

> **Note:** Assigning parameter values to the voltage source is dependent on the type of analysis we want to run. I'll explain how to setup the source in section 5 on running analyses.

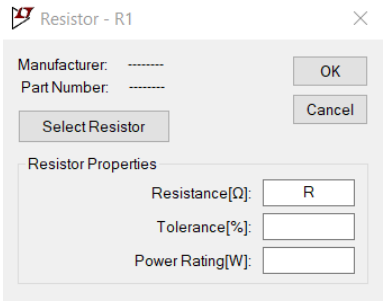

Fig.8

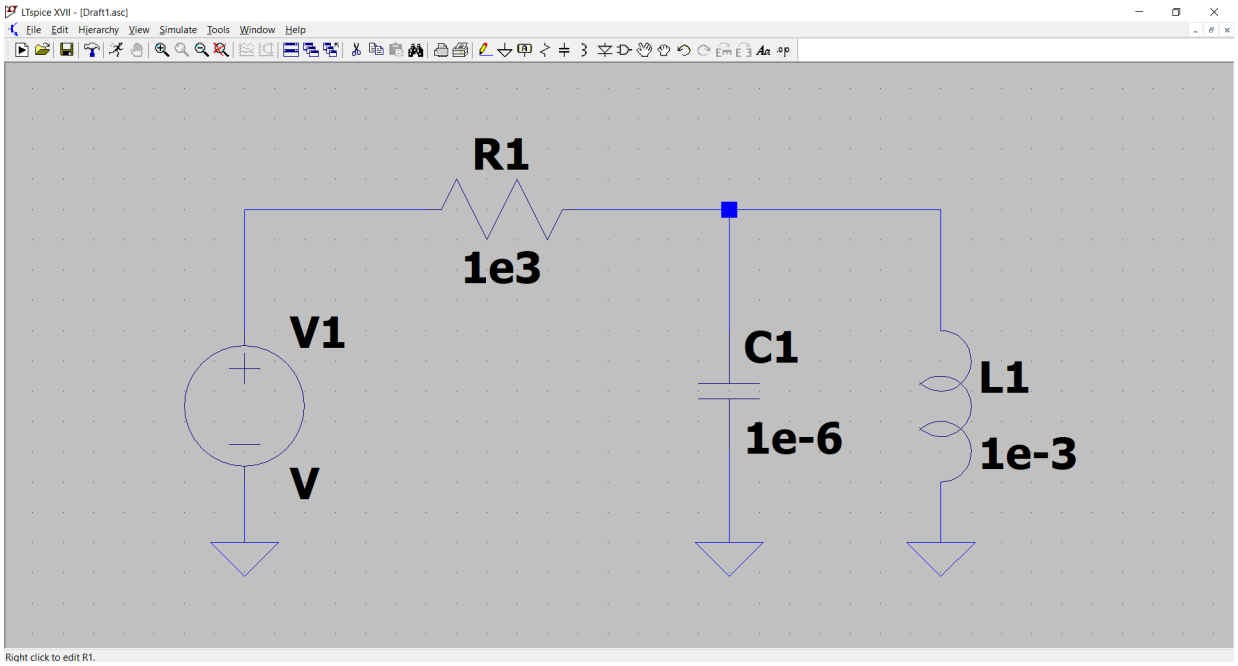

Fig.9

4.5 Naming Components

Assigning meaningful names to the components in your circuit has many advantages. One is that it makes the writing of simulator directives (see section 6) easier. Another is that you'll have a slightly easier time reading your circuit netlist in the event that you have to. Names are attached to one single instance of a component. As pointed out before, LTspice automatically assigned names to the components you placed. For example, LTspice named my resistor `R1` in **Fig.9** above. To change the name, right click on the name (not the component this time, but the name).

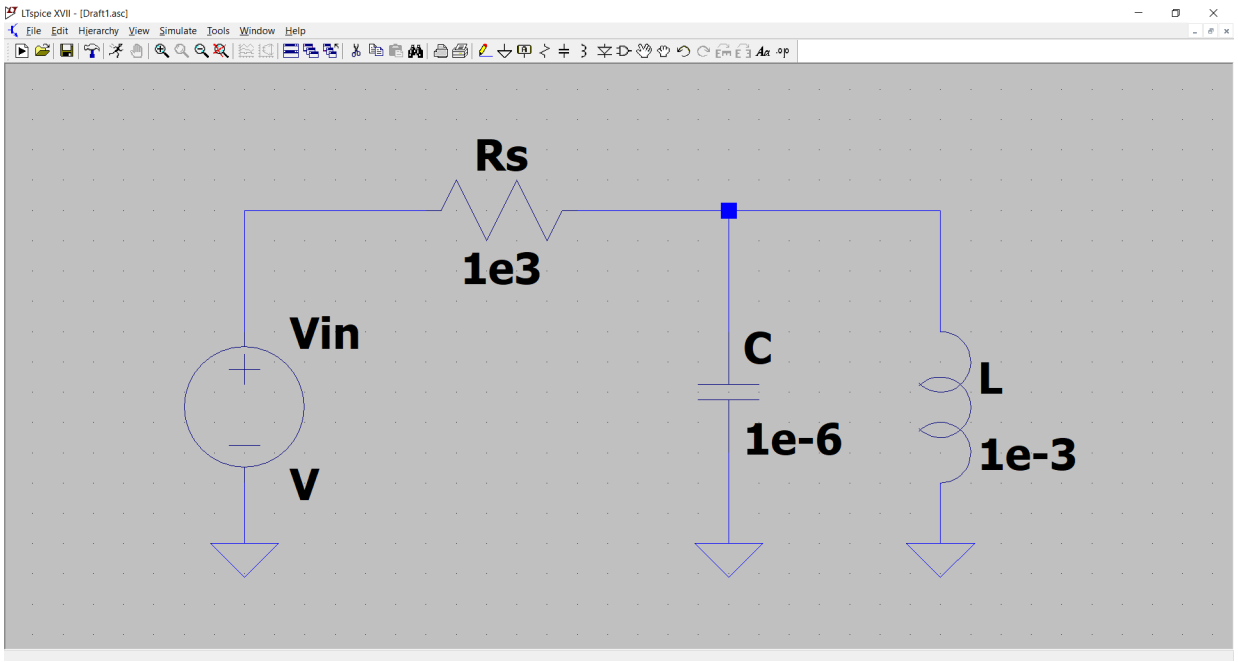

Fig.10

Note on naming and assigning values: To name our components and assign parameter values, we have used the GUI LTspice has provided us, which we access by right clicking on the components and the names. Later in the semester, you may need to access to parameters that aren't made available by LTspice through those GUIs. Instead of simply right clicking, you'll need to press **Ctrl** and then right click. This opens the more general **Component Attribute Editor** for that component. I'll discuss what you'll need to do then in later assignments.

4.6 Labelling Nets

Labelling nets with meaningful names is extremely important when creating circuit schematics. One reason is that, once again, the simulator directives will be easier to write, and the netlists easier to read. However, that is not all. Appropriately naming nets allows you to understand simulation results better. Finally, and probably most importantly, two nets bearing the same name are electrically connected, whether or not the connection is made visually or not. In **Fig.11** below, the voltage source is indeed connected to the rest of the circuit. To add a net name, all you need to do is click on **Label Net** in the toolbar. In the text box, type in the name you want, and then attach it to the nodes you want.

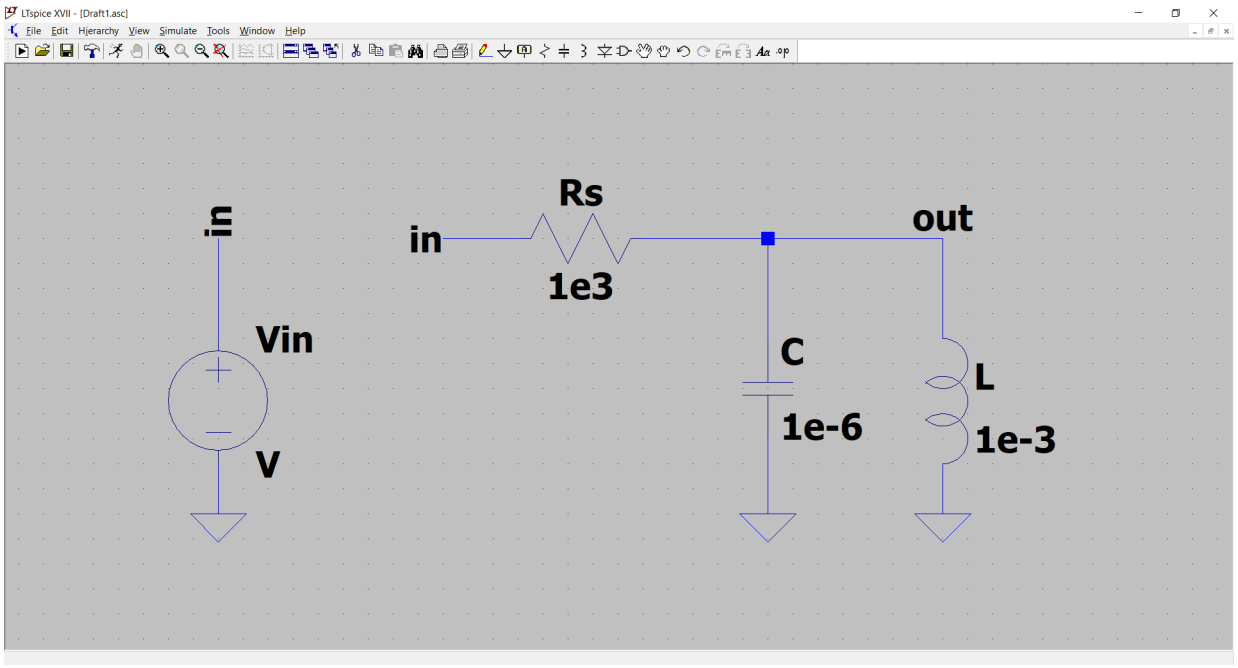

Fig.11

4.7 Printing your Circuit

If you every need to capture an image of your circuit within LTspice (for a report for example), you need to print your circuit to a PDF document first. This is done by clicking **File** → **Print**.

4.8 Conclusion

This is as far as I'll for now regarding creating circuits in the schematic editor. I hope that you know enough by now to be comfortable with LTspice's schematic editor. I encourage you to have another look at the netlist produced by LTspice before you move on, and to try to fully understand how the netlist translates to the circuit. The ability to read a netlist is a much more transferable skill than the ability to use LTspice's schematic editor.

5 Running Analyses

5.1 Overview

There are several analysis SPICE can run. Here, I describe the four most common ones that you'll need to know about this semester.

5.2 DC Operating Point Analyses

The DC Operating Point is the most basic analysis type provided by SPICE. Its purpose is compute the DC operating point of each component in your circuit (you'll understand what this means during your lectures on diodes and transistors). You can understand this to mean the combination of DC voltages and currents at each node and through each branch of the circuit that determine its AC behavior. When no AC signal is present, the analysis result represent the exact circuit solution.

5.2.1 Setup

The DC Operating Point analysis requires you to setup the voltage and current sources in your circuit. To do so, right click on the source. The GUI that appears allows you to quickly specify a DC voltage or current. Once you are ready to simulate, click on **Simulate** → **Edit Simulation Cmd**. In the *Edit Simulation Command* window that appears, select the **DC op pnt** tab (see **Fig.12**). Then, you simply click **Ok** and place the `.op` command anywhere on the schematic. Your circuit is ready to be simulated.

> **Note:** During DC operating point analyses, SPICE treats capacitors as open circuits, and inductors as short circuits.

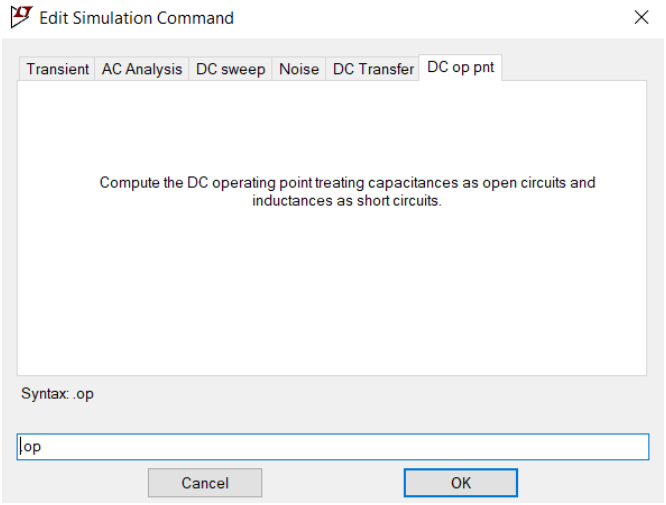

Fig.12

5.2.2 Example

In **Fig.13**, I have an example of the results from a DC operating point analysis. Notice the importance of labelling nodes to read the results here. As expected, `V(out)`, the voltage at node `out`, is 5 V.

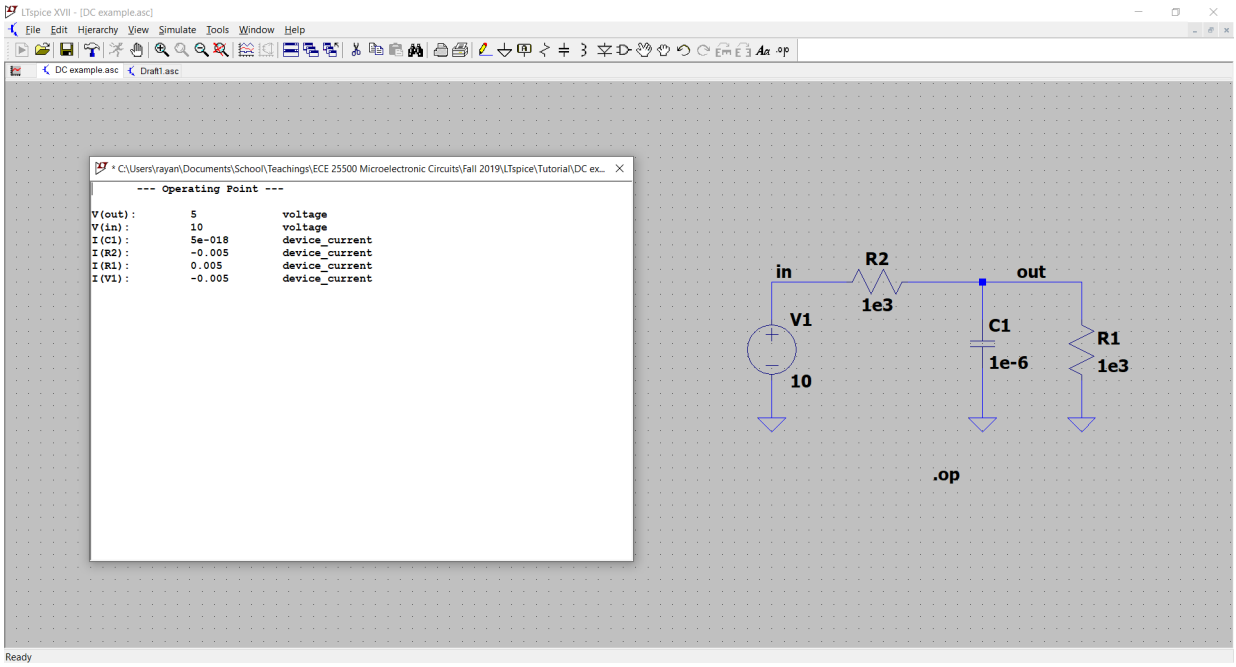

Fig.13

5.3 DC Sweep Analyses

DC sweep analyses are mostly used to compute the DC characteristics of circuits. It allows you to compute the DC operating point of a circuit while stepping independent sources, and while treating capacitors as open circuits and inductors as closed circuits.

5.3.1 Setup

Since the DC sweep analysis can only sweep independent sources, there needs to be at least one voltage or current source in your circuit. Take note of their names. Then, click on **Simulation** → **Edit Simulation Cmd**. Click on the tab named **DC Sweep**. You can sweep up to three sources. In the **Name of source to sweep** field, enter the name of the source you want to sweep. You then have the choice of sweeping linearly or logarithmically (**list** allows you to specify a list of value to step through. It isn't so useful).

5.3.2 Example

For this example, I've taken a simple rectifier circuit. You'll know what those are in just a few lectures. The DC sweep analysis will allow us to determine the DC characteristic of this circuit. See **Fig.14** for the simulation results. See **Fig.15** for the setup of the analysis.

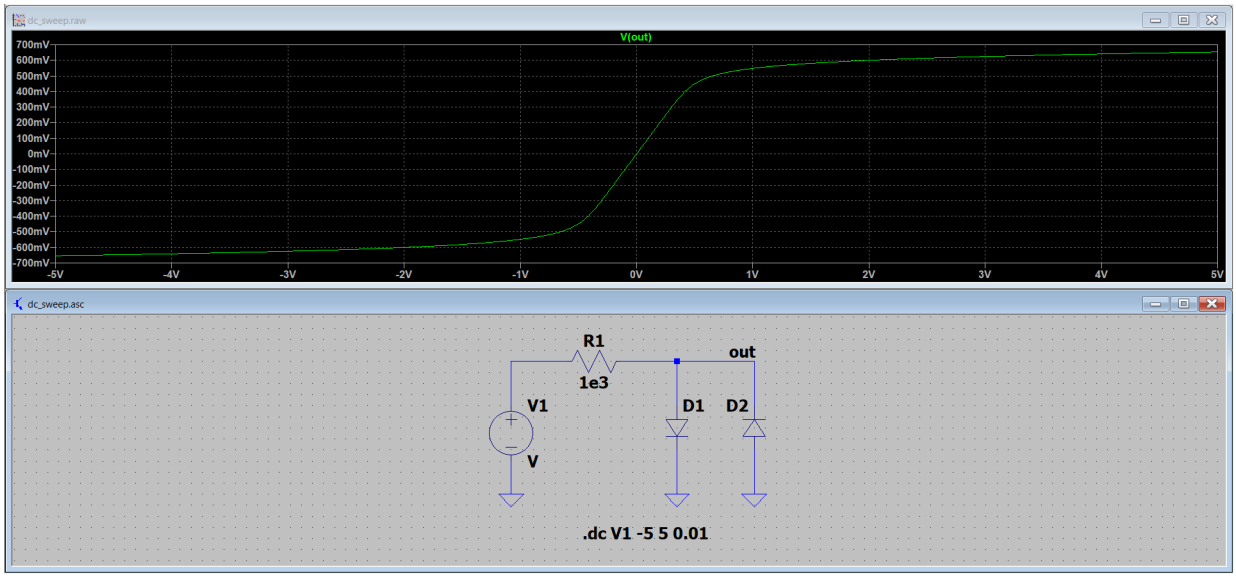

Fig.14

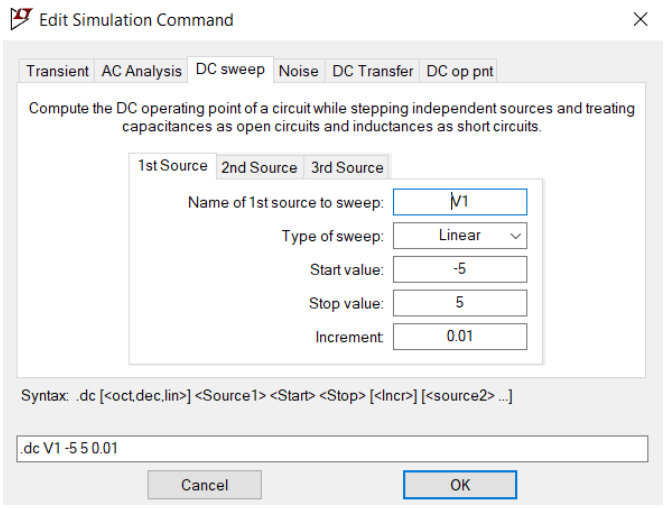

Fig.15

5.4 Transient Analyses

Transient analyses allow you to simulate circuit behavior over time. For this type of analysis, the dynamic relationships governing each components in your circuit are considered (e.g. capacitors are not open circuited).

5.4.1 Setup

To setup a transient analysis, you need to setup the sources. A typical example in ECE 25500 is to setup a source to output a sinusoidal waveform. Right click on a voltage source, and click on **Advanced**. You should have the window shown in **Fig.16** below. Out the available waveforms,

select **SINE**. In ECE 25500, we'll ignore the last four fields, and only care about DC offset, amplitude and frequency. To setup the actual analysis, open up the **Edit Simulation Cmd** window and click on the **Transient** tab. Usually, specifying the **Stop Time** is the only thing you'll have to do.

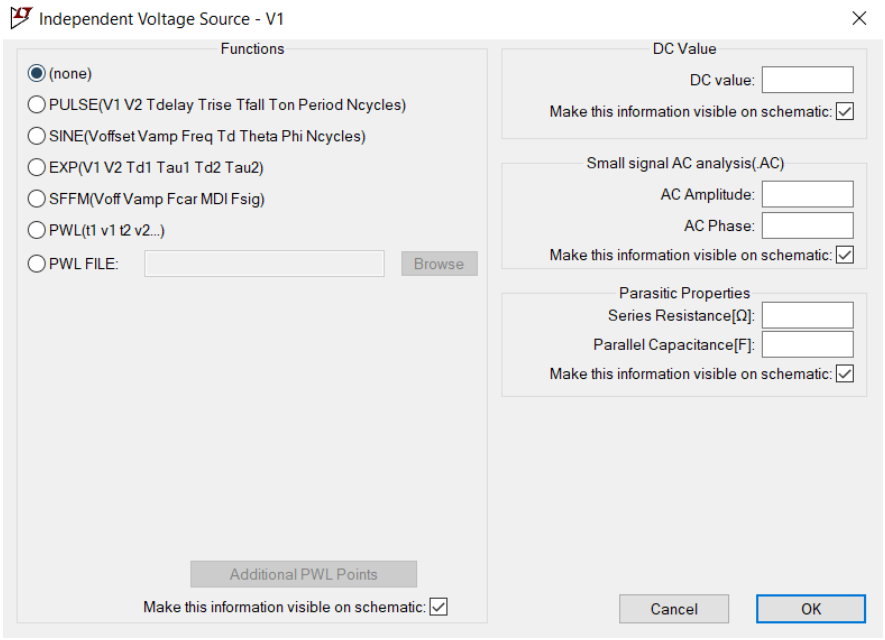

Fig.16

5.4.2 Example

I've taken the rectifier circuit from the DC sweep analysis example and specified an input sine wave of amplitude 2 V and frequency 1 kHz. The source setup is shown in **Fig.17** below. I've then setup the simulation stop time at 5 ms (see **Fig.18**). The simulation results are shown in **Fig.19**. I've plotted both the input and output in the results.

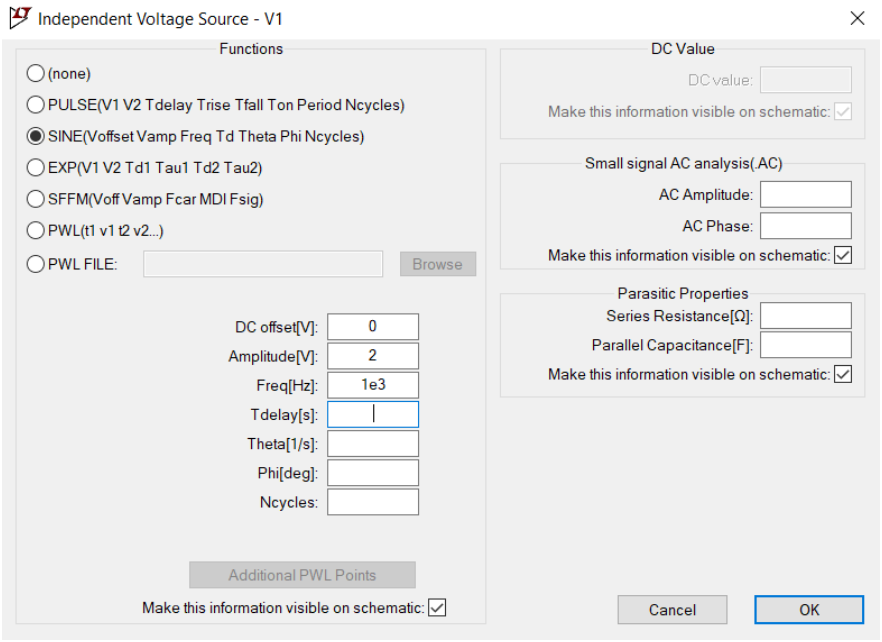
Independent Voltage Source - V1
Functions
(none)
PULSE(V1 V2 Tdelay Trise Tfall Ton Period Ncycles)
SINE(Voffset Vamp Freq Td Theta Phi Ncycles)
EXP(V1 V2 Td1 Tau1 Td2 Tau2)
SFFM(Voff Vamp Fcar MDI Fsig)
PWL(t1 v1 t2 v2...)
PWL FILE:
Browse
DC offset[V]: 0
Amplitude[V]: 2
Freq[Hz]: 1e3
Tdelay[s]:
Theta[1/s]:
Phi[deg]:
Ncycles:
Additional PWL Points
Make this information visible on schematic:
DC Value
DC value:
Make this information visible on schematic:
Small signal AC analysis(.AC)
AC Amplitude:
AC Phase:
Make this information visible on schematic:
Parasitic Properties
Series Resistance[Ω]:
Parallel Capacitance[F]:
Make this information visible on schematic:
Cancel
OK

Fig.17

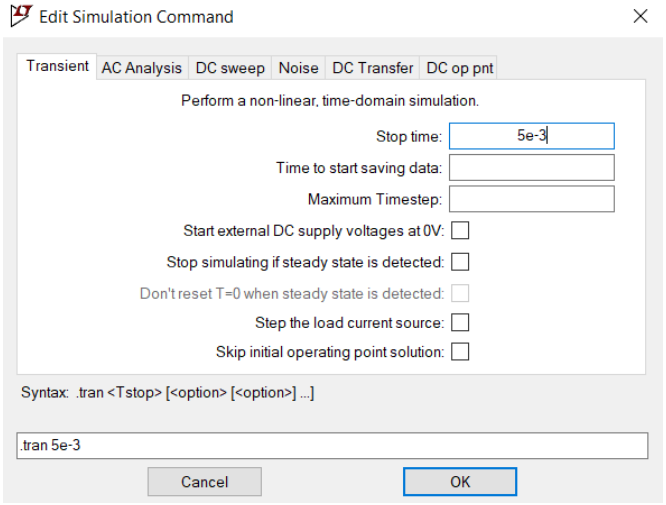
Edit Simulation Command
Transient
AC Analysis
DC sweep
Noise
DC Transfer
DC op pnt
Perform a non-linear, time-domain simulation.
Stop time: 5e-3
Time to start saving data:
Maximum Timestep:
Start external DC supply voltages at 0V:
Stop simulating if steady state is detected:
Don't reset T=0 when steady state is detected:
Step the load current source:
Skip initial operating point solution:
Syntax: .tran <Tstop> [<option> [<option>] ...]
.tran 5e-3
Cancel
OK

Fig.18

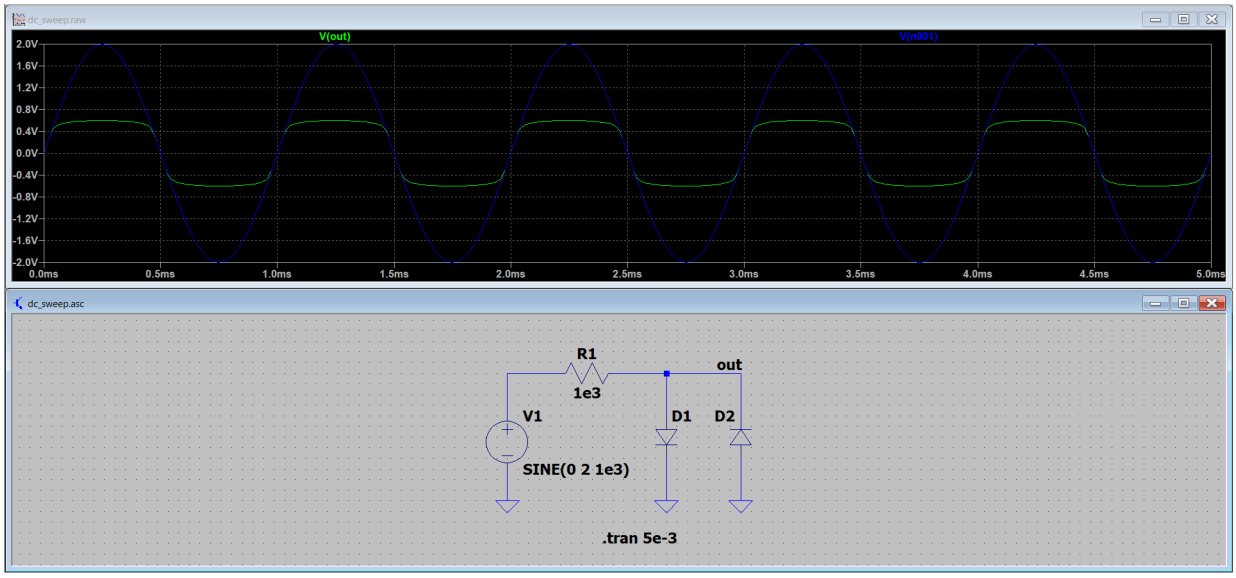

Fig.19

5.5 AC Analyses

The AC analysis is used to compute the transfer function of a circuit; that is, the circuit gain and phase response versus frequency.

5.5.1 Setup

For this type of analysis, the independent sources have a special setup as well. Right click on your voltage source, and then click on **Advanced**. In the window that appears, you need to specify the **Small Signal AC Analysis** section (see **Fig.20** below). For all circuits in ECE 25500, the **AC Amplitude** field should be 1, and the **Phase** field should be 0 (these will be otherwise only when you have more than one source is your circuit). To setup the actual analysis, once again open up the **Edit Simulation Cmd** window and select the **AC Analysis** tab. The setup is similar to the DC sweep setup. We will prefer a logarithmic (decade) sweep for most applications.

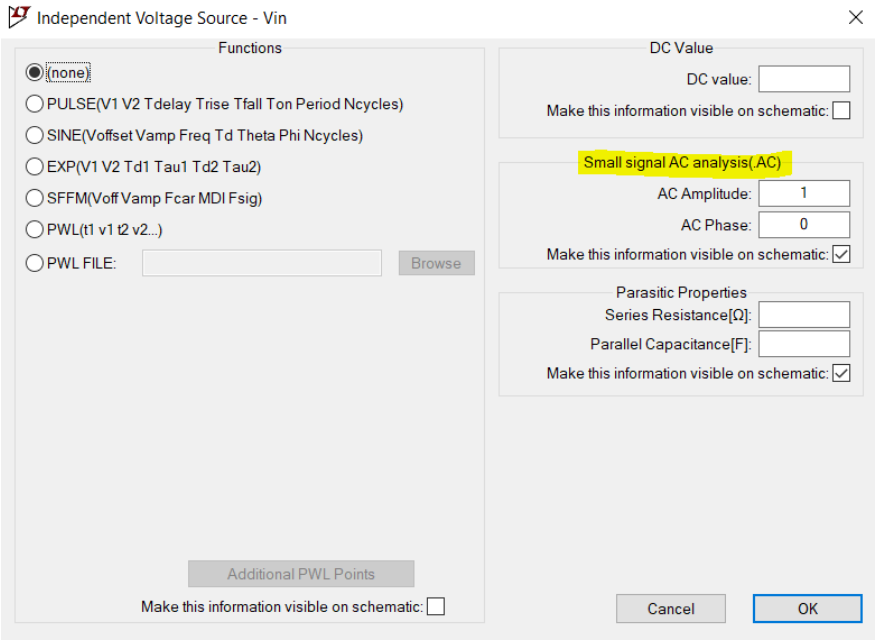

Fig.20

5.5.2 Example

I've taken the filter we build in section 4 and ran an AC analysis on it. The analysis setup is shown in **Fig.21** below. The simulation results are shown in **Fig.22**. Notice that both the phase and magnitude responses are plotted on the waveform viewer. You can remove the phase plot by right clicking on the phase axis and clicking on **Don't plot phase**.

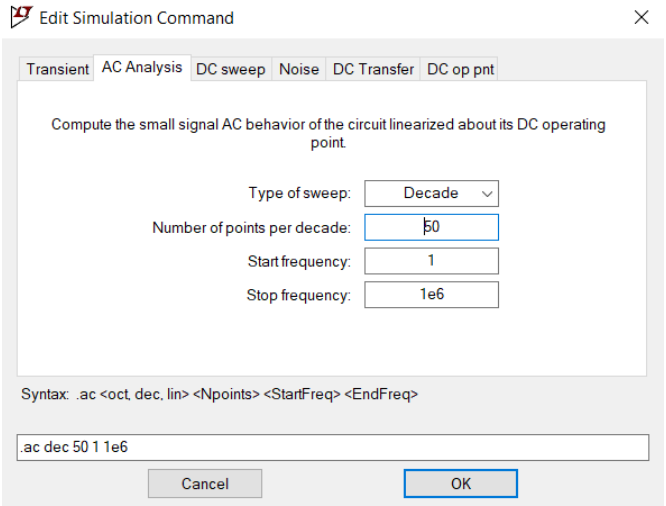

Fig.21

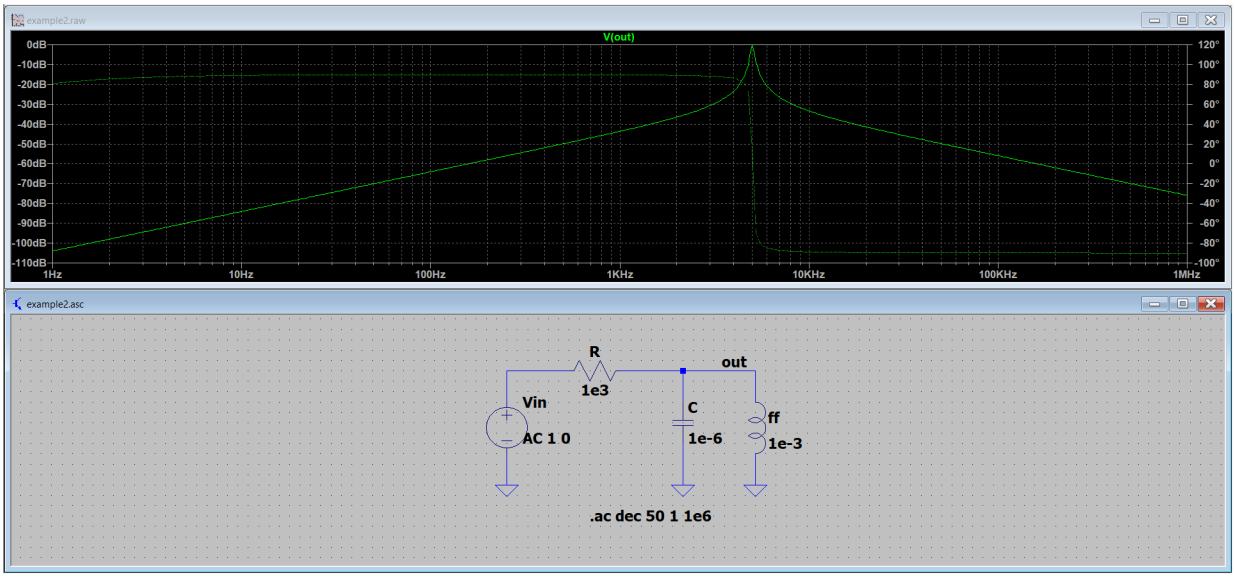

Fig.22

5.6 Printing your Plots

If you ever need to capture your plots, you'll need to print them. **This is imperative!** Taking a screen-shot of your plot on the black background will not be accepted. Firstly, remove the grid by right clicking somewhere on the plot background, and then going to **View** → **Grid**. Then, click on **File** → **Print**.

6 Using Simulator Directives

6.1 Overview

As we have discussed in the background section, a circuit netlist consists of two main entities. One is a component description, which, in one line, tells SPICE what component is instantiated, what name it bears, where it connects, and what the values of its parameters are. For example, in a previous circuit, we had placed an inductor. The corresponding line on the netlist looked like the following: `L1 out 0 1e-3`, where `L` indicates that the component is an inductor, `out 0` indicates that the inductor connects from node `out` to the reference node, and `1e-3` is the value of the inductance. The other entities that we find in netlists are **simulator directives**, or **dot commands**. These control the behavior of the simulator at run-time. For example, SPICE is told which type of analysis to run, along with how to set it up, through a directive. The following directive sets up an AC analysis: `.ac dec 20 1 1e6`, where the gain is computed between 1 Hz and 1 MHz at 20 points per decade. Beyond selecting an analysis type, directives are used for many other desirable behaviors such as specifying initial conditions (e.g. charge of a capacitor), analyzing the results of a simulation run, stepping a parameter or customizing a component's model. These are just a few of the supported directives. However, they are certainly some of the most common ones. Below, I describe in moderate detail the purpose and use of these directives I have just listed.

6.2 Placing a Directive

To place a simulator directive, click on **SPICE Directive** in the toolbar (the `.op` icon). A text box will appear on which you can type a directive. Once you have pressed **Ok**, you need only place it anywhere on the canvas. LTspice provided a convenient GUI to edit most of the available directives. In order to access it, you simply need to right click on the placed directive. For example, the GUI for a `.meas` directive will look like **Fig.23**.

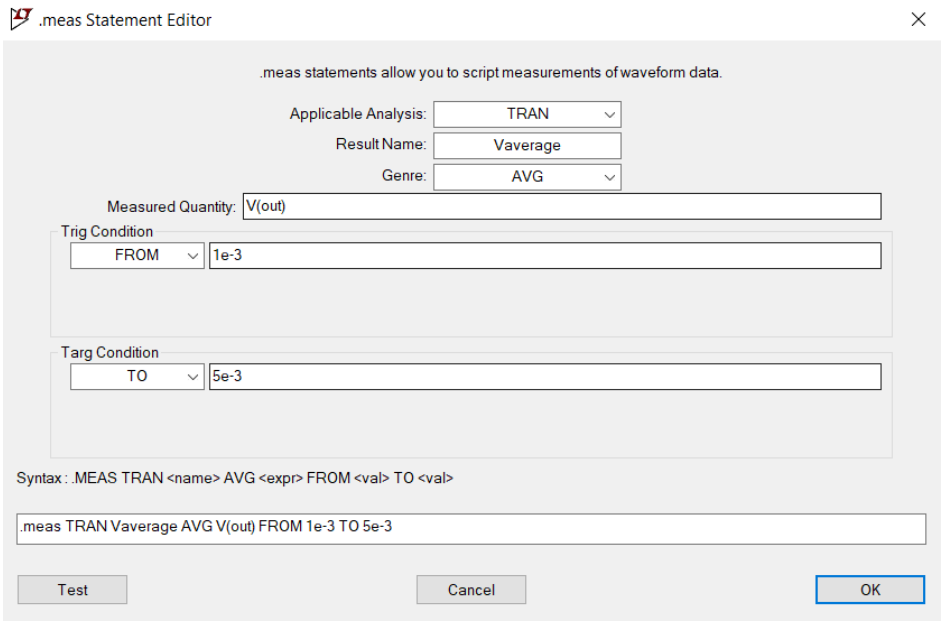

Fig.23

Note: Some simulator directives, such as the `.meas` directive, have complex syntax and are highly flexible. It is impossible to encompass the whole of their versatility in just a few lines of description. We can, however, describe the syntax of directive with something like this:

```
.ic [V(<n1>)=<voltage>] [I(<inductor>)=<current>]
```

In this notation, the square brackets indicate optional arguments, meaning that what is inside the brackets need to be included in the directive statement for it to be syntactically correct. The angled brackets indicate a parameter that needs to be populated. For example, in the above directive, if we wanted to initialize the voltage at node X to 1 V, we would write:

```
.ic V(X)=1
```

See below for a more detailed description of the `.ic` directive.

6.3 Specifying Initial Conditions (`.IC`)

6.3.1 Description

The `.ic` directive is used to initialize node voltages and branch currents before a transient analysis is run. Although SPICE will interpret the `.ic` directive in a specific way for a DC analysis, I encourage you to only use it for transient analyses.

6.3.2 Setup

The general syntax of the `.ic` directive is as follows:

```
.ic [V(<n1>)=<voltage>] [I(<inductor>)=<current>]
```

Here, for example, `V(<n1>)` refers to the voltage at some unspecified node `<n1>`. Node voltages and branch currents have to be specified one by one (more than one of each can be specified in one directive).

6.3.3 Example

As an example of the `.ic` directive, consider the discharging capacitor shown in **Fig.24** below. We can examine the rate of discharge over a period of 5 ms using a transient analysis. To specify an initial charge on the capacitor, I have named the node above `x`. Then, the required directive is simply `.ic V(x)=1`.

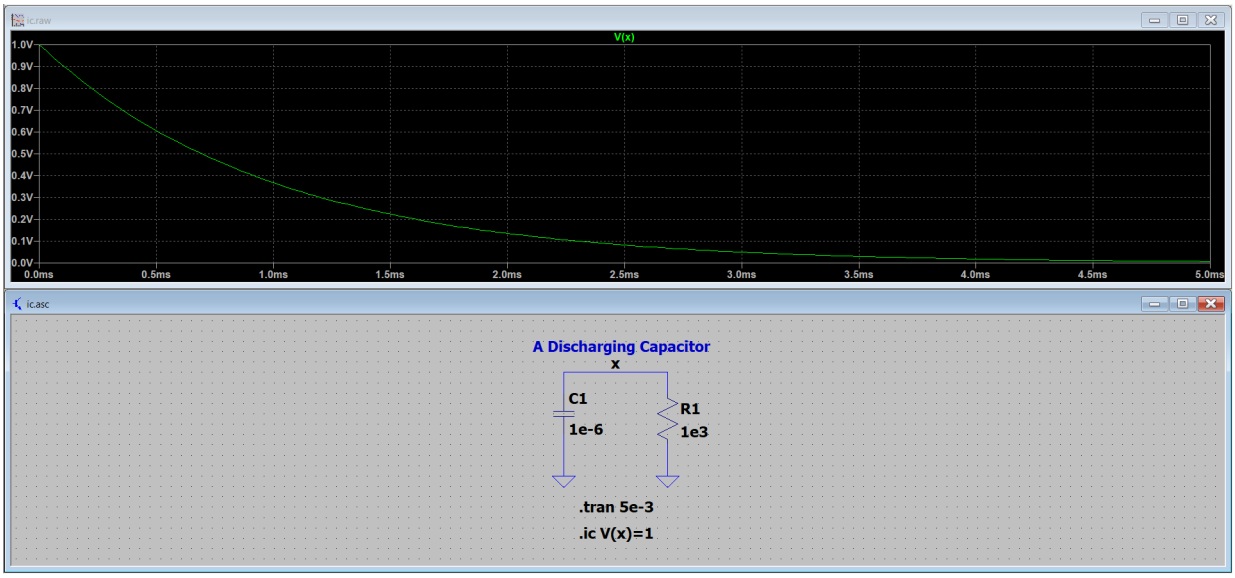

Fig.24

6.4 Evaluating Electrical Quantities (`.MEAS`)

6.4.1 Description

The `.meas` directive is an extremely versatile directive. It essentially allows you make measurements on the result of a simulation, such as computing the peak value of a waveform, finding out when a rising voltage reaches a particular value, calculating RMS currents, and much more. Click on **Help** → **Help Topics** and read the help page on the `.meas` directive to learn about how it is setup (you'll see that there is too much to describe here). Instead, I'll give one example of it in action.

6.4.2 Example

In this course, there will be many instances in which you will have to compute averages of a waveform. The `.meas` is the directive you are expected to use for that purpose. Consider the half-wave rectifier circuit shown in **Fig.25** below. This is a circuit you'll see again later. The source is a sine wave of amplitude 10 V and frequency 1 kHz. I have plotted the voltage waveform at node `out`. We can easily compute the average output voltage using the `.meas` directive shown in the figure.

> ***Note:*** The simplest way to setup a `.meas` directive is to use the built-in GUI. Refer to section 6.2.

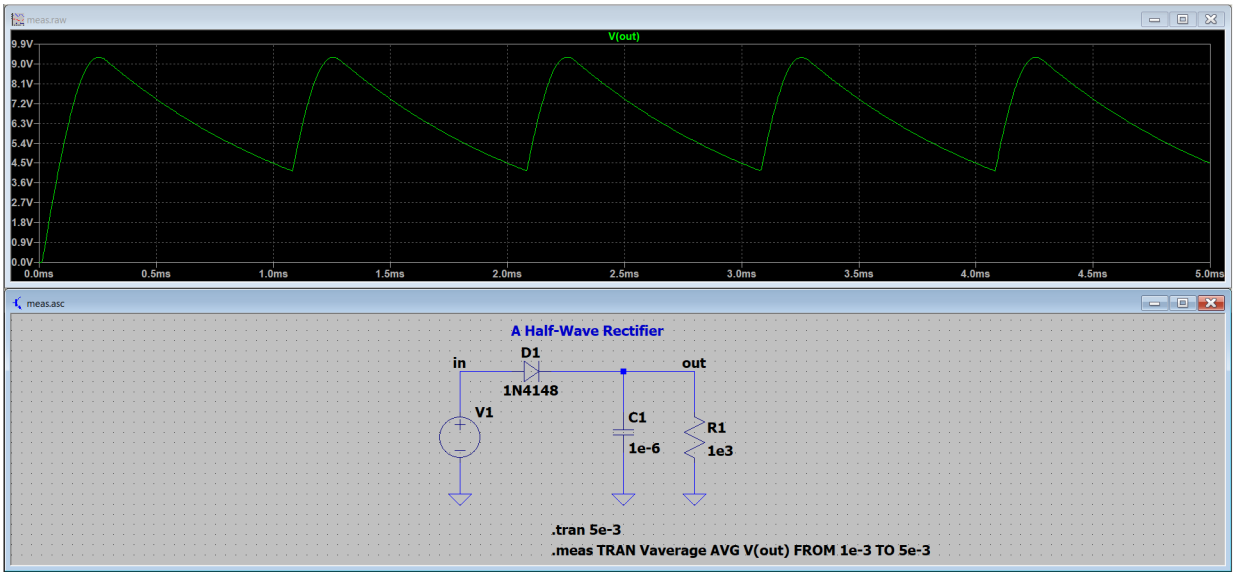

Fig.25

6.5 Sweeping a Parameter (`.STEP`)

6.5.1 Description

The `.step` directive allows you to repeat an analysis several times for several values of a particular parameter. This is useful when you want to figure out how your circuit's behavior changes as you increment a parameter, such as the resistance of some resistor. In essence, this directive automates the process of incrementing a parameter and running the analysis over a defined range. The DC sweep analysis we have discussed earlier is actually a `.step` directive in disguise, for which the parameter LTspice increments is the source voltage.

6.5.2 Setup and Example

As with most other popular directives, click on **SPICE Directives** in the toolbar, type `.step` in the text box, press **Ok**, and place the directive somewhere on your schematic. Then, right click on the directive to edit its syntax using the LTspice GUI for it (see **Fig.26** below). In the **Name of parameter to sweep** field, you should enter the name of whatever parameter you want to sweep. However, this name is **not** the name of a component. Instead, it is a name you have define yourself in the **Value** field of a component's attribute editor. For example, if I wanted to sweep the resistance of a resistor, instead of entering an absolute resistance for the value of a resistor, I enter {`R`} (see **Fig.27** below). This defines the parameter `R`, which can be swept by a `.step` directive. Looking again at **Fig.**, I have entered `R` in **Name of parameter to sweep**.

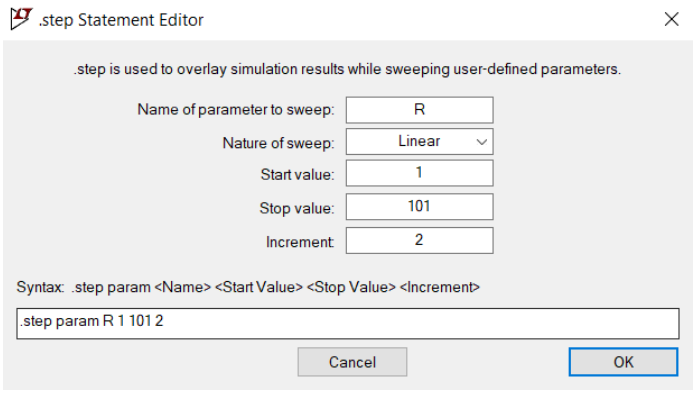

Fig.26

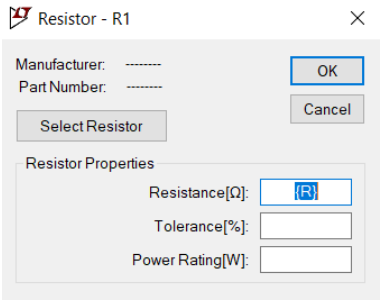

Fig.27

After the parameter name is defined, we can specify the type of sweep. The LTspice GUI for the `.step` directive should be rather intuitive to use. As example, I created a voltage divider and ran a DC analysis while sweeping the load resistance as shown back in **Fig.26** above. The result is shown in **Fig.28**.

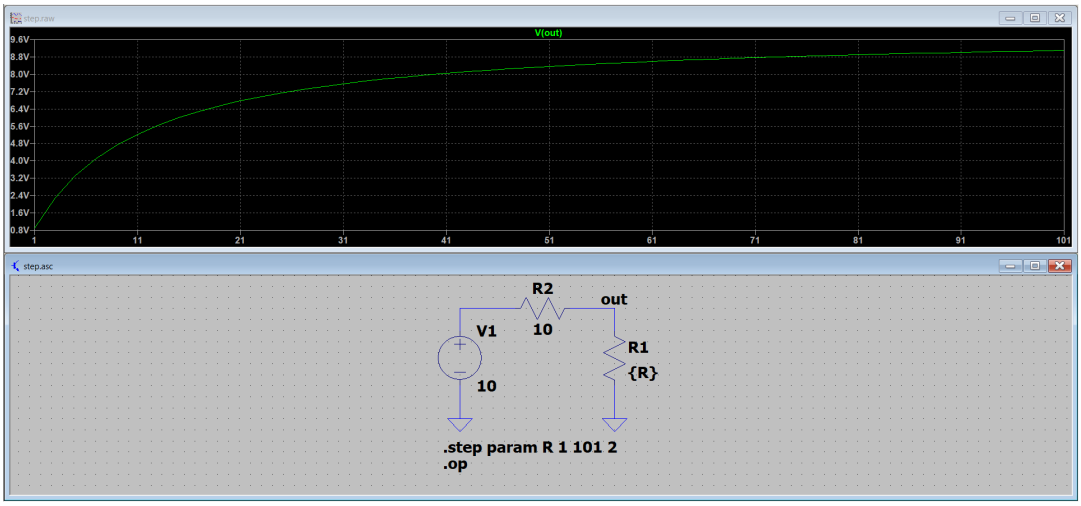

Fig.28

6.6 Customizing a Component's Parameters (`.MODEL`)

6.6.1 Description

As I have discussed in section 3.3, SPICE ships with device models. The user needs only to specify the device parameter model of a device for it to function. For most circuits you'll be creating in LTspice, the device parameter models LTspcie provides will be sufficient. However, towards the end of the semester, we will need to create our own. This is what we will use the `.model` directive for. I'll describe exactly how you'll need to use it then.

6.6.2 Setup

The syntax of the `.model` directive is `.model <model name> <type>[(<parameter list>)]`. Here, you specify the name of new model, the type of model (as in, which device is being given parameters here), and the list of parameters.

6.7 Conclusion

The idea behind this section on SPICE directives is to provide quick examples on how they are usually used. The information is not exhaustive. To learn more about how to use the directives, in LTspice, click on **Help** → **Help Topics**.