Physics 80 Lab Manual

December 22, 2018

Chapter 1

Introduction to Plotting

1.1 Introduction

This lab will introduce calculations and plotting techniques using numpy arrays within Scientific Python.

1.2 Plotting discrete data and continuous functions

Consider the Jupyter notebook example in Fig. **??** which plots the sin function sampled a discrete values. Note the following key features, which you will use repeatedly today and in future labs:

- Use of global variables `UPPER` and `STEP` at the top of the snippet, allowing for easy adjustment of parameters that affect the plot.
- Use of `np.arange` to define an array of x values.
- Creation of the array y, defined by $y = \sin(2\pi x/5)$ for each value of x. One of the great joys of using numpy is the ability to avoid getting bogged down with explicit for loops.
- Use of slicing techniques `x[:5]` to show only the first five entries for debugging.
- Plotting the arrays of x and y values with `plt.plot` using the `"bo"` option for blue circles.
- Defining appropriate axis labels with `plt.xlabel` and `plt.xlabel`.
- Creation of a legend using the `label` option to `plt.plot` and the plot.legend() command.

Notice that even in this simple example, I've added some intermediate feedback from my code in the form of the screen dumps of the first few values of x and y. It's a common pitfall to try and rush ahead to the final product when programming. But it is must faster and reliable to break your task into small steps, and establish feedback at each small step. To plot a continuous function with Scientific Python, you will still use discrete data but:

- Use much finer binning of the x axis variable to draw a smooth curve.
- Use the line option `"-"` or dashed line `"--"` instead of points with `"o"`.

Plot 1: Plot the quadratic function $y = x^2$ with the following requirements:

```
# plot a sin function
UPPER = 10
STEP  = 0.25
x = np.arange(0,UPPER,STEP)
y = sin(2*pi*x/ 5.0)
print("dumping first five entries:")
print("x[:5]:", x[:5], "...")
print("y[:5]:", np.around(y[:5],2), "...")
plt.plot(x,y,"bo",label="sin")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
```

```
 dumping first five entries:
 x[:5]: [0.   0.25 0.5  0.75 1.  ] ...
 y[:5]: [0.   0.31 0.59 0.81 0.95] ...
```

```
<matplotlib.legend.Legend at 0x11781ef98>
```

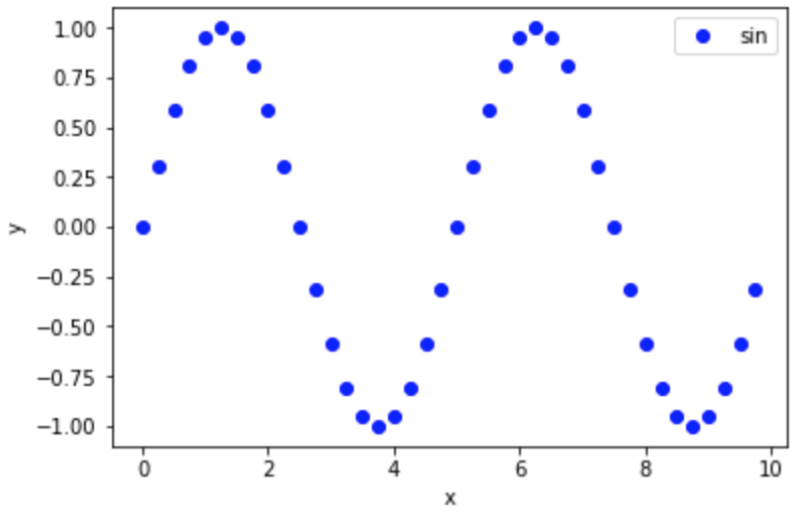


Figure 1.1: Circuit for verifying Ohm's law as a (a) circuit diagram, and (b) implemented using your lab equipment.

- Plot in the range $x = [0, 20)$.
- Plot discrete samples with a step size of 2 using blue circles.
- On the same axis, plot the corresponding smooth function using a red solid line.
- Add appropriate axis labels.
- Add a legend for the "discrete" and the "smooth" function.

1.3 Multivariate analysis using boolean masks

```
x = np.array([1,2,3,4,5,6])
y = np.array([1,2,1,2,1,2])
cut = (y > 1)
print("cut:  ", cut)
print("y subject to cut:  ", y[cut])
print("x subject to cut:  ", x[cut])
```

```
cut:   [False  True False  True False  True]
y subject to cut:   [2 2 2]
x subject to cut:   [2 4 6]
```

Figure 1.2: Using boolean masks to cut on variable y.

A powerful technique in Scientific Python for performing analysis involving multiple variables uses boolean masks as shown in Fig. **??**. In the example:

- Two numpy arrays x and y `of the same length` are defined to contain the collected data.
- The cut defined by $y > 1$ is a boolean array of the same length as x and y which is true at indices where the condition is met and false where it is not.
- The subset of the entire y array defined by `y[cut]` consists only of those entries of y for which the condition $y > 1$ is met.
- The subset of the entire x array defined by `x[cut]` consists only of those entries of x for which the condition $y > 1$ is met for the corresponding y value.

The last item shows the real power of this technique, one can look at one variable subject to constraints on another variable.

Next consider the sample data in Table **??** which comes from experimental measurements of a voltage level v at discrete times t. The measurement is subject to a high-frequency noise monitoring by the variable n. The noise is only present for $n > 6.0$. A straightforward way to load this data into scientific python is by defining numpy arrays for each variable as follows:

Table 1.1: Sample data for a voltage measurement subject to high frequency noise.

t (s)	v (V)	n
0.4	0.25	2.8
1.1	2.37	7.3
1.4	1.69	9.7
1.9	0.93	1.3
2.5	-1.0	6.2
3.0	0.95	4.8
3.4	1.22	6.9
4.1	0.54	4.0
4.4	0.37	1.9
4.8	0.13	4.0
5.5	-2.04	9.5
6.2	-2.06	8.7
6.5	-0.81	2.3
7.0	-0.95	5.3
7.5	0.98	9.7
7.9	0.27	8.3
8.5	-0.81	0.1
9.0	-0.59	5.1
9.4	-0.37	4.4
9.9	0.56	9.9

```
t = np.array([0.4, 1.1, 1.4, 1.9, 2.5, 3.0, 3.4, 4.1, 4.4, 4.8,
                      5.5, 6.2, 6.5, 7.0, 7.5, 7.9, 8.5, 9.0, 9.4, 9.9])
v = np.array([ 0.25, 2.37, 1.69, 0.93, -1.0, 0.95, 1.22,
                       0.54, 0.37, 0.13, -2.04, -2.06, -0.81, -0.95,
                       0.98, 0.27, -0.81, -0.59, -0.37, 0.56])
n = np.array([2.8, 7.3, 9.7, 1.3, 6.2, 4.8, 6.9, 4.0, 1.9, 4.0,
                       9.5, 8.7, 2.3, 5.3, 9.7, 8.3, 0.1, 5.1, 4.4, 9.9])
```

Plot 2 Prepare a plot the sample data subject to the following:

- Plot the voltage as a function of time as discrete data using red points.
- Define the boolean array `keep` based on the noise reducing condition $n <= 6.0$.
- Plot the voltage as a function of time, subject to the noise reducing condition using blue points.
- Plot the function $\sin(2\pi x/10)$ as a smooth function.
- Add appropriate axis labels.
- Add a legend for "raw" data with no cut, "clean" data with noise removed, and your continuous "sin" function.

Your plot will reveal a clear sine function in the discrete data (after noise removal) consistent with the continuous function.

1.4 The Logistics Map

The logistics map is the recurrence relation

$$x_{n+1} = r\, x_n\, (1 - x_n)$$

with the variable x between 0 and 1. The variable x can be thought to represent the ratio of a population to its maximum possible value. The population increases due to birth and decreases due to starvation as the population approaches it's maximum value (x near 1). This leads to the non-linear relationship that defines the logistic map. The mapping keeps the variable x between 0 and 1 as long as the parameter r is in the range $[0, 4]$.

This logistic map is frequently encountered as a simple example of a chaotic system emerging from a simple non-linear system. If we consider the long term behavior of the population x as a function of the parameter r, as shown in Fig. **??**, we see that for values of r less than 3 the population approaches a single fixed value. At the value $r = 3$ the non-linear system exhibits bifurcation with the population oscillating between two values. As r increase, further bifurcations occur at an ever increasing rate until the systems exhibits chaotic behavior alternating with occasional returns to stable oscillations.

The long term behavior of the logistics map can be easily modeled in Scientific Python. A start is shown in Fig. **??** where you should understand:

- An array of r values is defined.

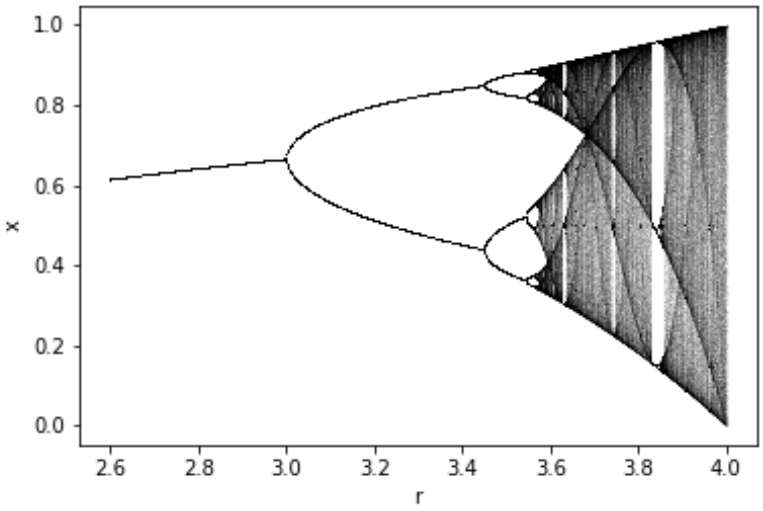


Figure 1.3: Long term behavior of the logistics map.

- An array of x values of the same size as r is defined and initialized to an arbitrary non-zero value (0.01).
- Two example iterations of the logistic map are applied.
- The next two iterations of the values of x are plotted as function of r on the same plot.

Plot 3: Reproduce the figure in Fig. **??** by doing the following:

- Define two global variables `ITER = 10` and `PLOT = 5`.
- Apply the logistics map `ITER` times by using a for loop.
- Apply the logistics map an additional `PLOT` times, plotting the values of x as a function of r, as in the example, each time.

You'll observe the long term behavior by increasing the value of `ITER` to a large value, such as 10,000. You'll see the full dependence on r by decreasing the step size in the initialization of the numpy array r to something like 0.001. You'll observe the chaotic behavior by increasing the value of `PLOT` to 100 or even 1000 iterations. To make a prettier plot using finer points (once you have a large number of points) you can reduce the size by adjusting the `s=10` parameter in the call to `plt.scatter` to something like `s=0.0001`.

```
r = np.arange(2.6,4.0,0.2)
print("r:  ", r)
R_SIZE = r.size
x = np.full(R_SIZE, 0.01)
print("initial x:  ", x)
x = r * x*(1.0 - x)
print("one iteration x:  ", np.around(x,2))
x = r * x*(1.0 - x)
print("two iterations x:  ", np.around(x,2))
# plot the next two iterations:
x = r * x*(1.0 - x)
plt.scatter(r,x,s=10,color="black")
x = r * x*(1.0 - x)
plt.scatter(r,x,s=10,color="black")
plt.xlabel("r")
plt.ylabel("x")
```

```
r:   [2.6 2.8 3.  3.2 3.4 3.6 3.8]
initial x:   [0.01 0.01 0.01 0.01 0.01 0.01 0.01]
one iteration x:   [0.03 0.03 0.03 0.03 0.03 0.04 0.04]
two iterations x:   [0.07 0.08 0.09 0.1  0.11 0.12 0.14]
```

```
Text(0,0.5,'x')
```

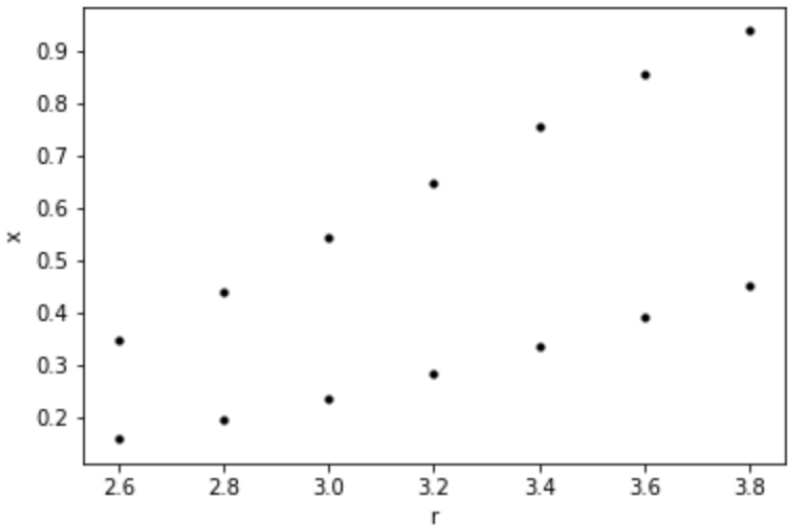


Figure 1.4: Modeling the logistics map.

Chapter 2

DC Circuits

2.1 Introduction

In this lab, you will learn how to use your digital multimeter (DMM) and bench-top DC power supply to explore DC circuits involving resistors. You will experimentally verify Ohm's law and the equivalent resistance for resistors in series and parallel. You will solder two resistor circuits to explore the Δ-Y transformation for three terminal networks.

2.2 Benchtop Power Supply

2.3 Digital Multimeter

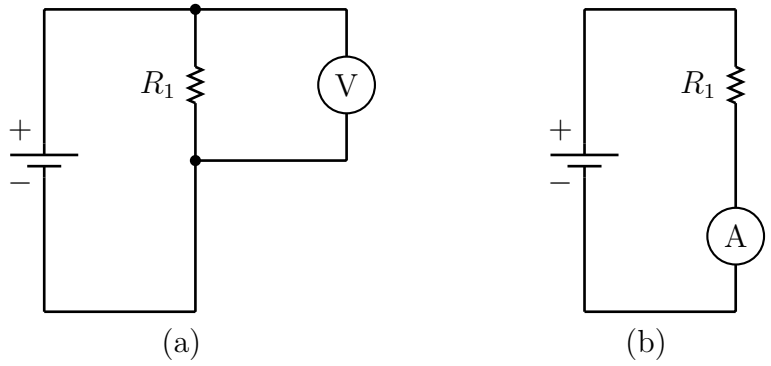


Figure 2.1: Circuits for verifying Ohm's law.

In this lab, you will use two new pieces of lab equipment: your bench-top power supply and your digital multimeter (DMM).

Your bench-top supply provides up to XX volts of DC power on two independent channels. For this lab, we'll only be using a single channel. Each supply has two associated knobs, one controlling the maximum allowed current, and one controlling the maximum voltage. The supply will provide the maximum voltage subject to those constraints. Usually your are primarily concerned with setting the voltage by the voltage knob, but it is a good habit, which will save you from losing components, to set the current as low as possible. An LED indicates if your circuit is current

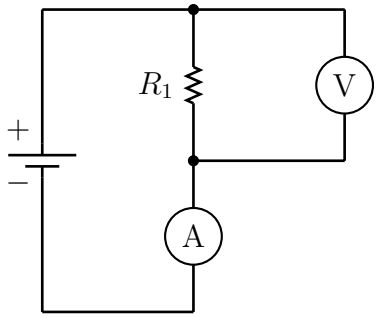
limited or voltage limited. You can adjust the current limit until just beyond the point where your circuit become voltage limited.

The supply is floating, it provides the specified voltage between the black and red outputs without referencing either to ground. If you want to provide a ground referenced voltage you explicitly connect the green output to the red (positive) terminal or the black (negative) terminal.

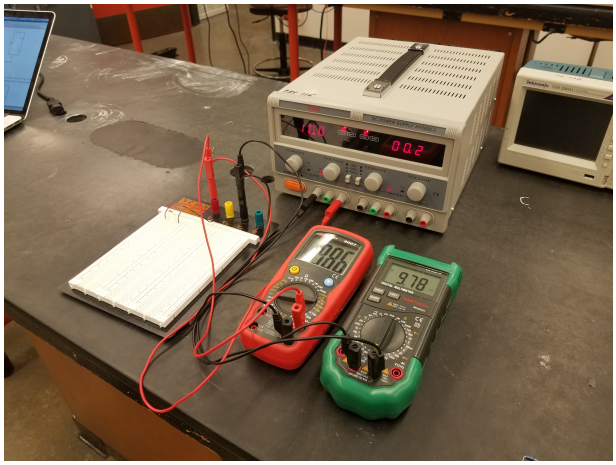
(Discuss limits)

Your DMM can measure voltage, resistance, and current.

2.4 Verification of Ohm's Law



(a)



(b)

Figure 2.2: Circuit for verifying Ohm's law as a (a) circuit diagram, and (b) implemented using your lab equipment.

Build the circuit in Fig. **??**. Use a resistor $R_1 = 1.0$ kΩ with a 1% tolerance. Use your Triplett 9007 as the voltmeter and the Mastech MS8624 as the current meter. Use your benchtop power supply to provide the voltage.

By adjusting the voltage setting of the power supply, take a series of voltage and current measurements with voltage across the resistor at target voltages from 1 to 10 V in steps of 1 V. Generally, you can measure more precisely than you can control, so never fuss about trying to measure the voltage at exactly the target value, instead, simply record e.g. $V = 1.04$ V along with your current measurement and move on to the next target value.

While recording data, check that the current values you measure are consistent with what you expect given the voltage across the resistor and resistance. You should *always* make quick sanity calculations when collecting data, otherwise you risk wasting time collecting useless data!

Plot 1: Plot the current versus voltage of your ten data points (using option `"o"`). Draw a line (using option `"-"`) for the current versus voltage curve of a 1.0 kΩ resistor. Make certain your plot has appropriate axis labels, including appropriate units in parenthesis, and a legend distinguishing data from your expectation ("expected"). **Measurement 1:** After taking your last measurement, leave all the connections in place and the power-supply at 10 V. Record in your log book the resistance of the resistor R_1 reported by your DMM. Is this a reasonable measurement? **Measurement 2:** Turn off the DC supply and record the resistance reported by the DMM. Is this

accurate? **Measurement 3:** Remove the resistor from your circuit and measure the resistance with your DMM. Is this accurate?

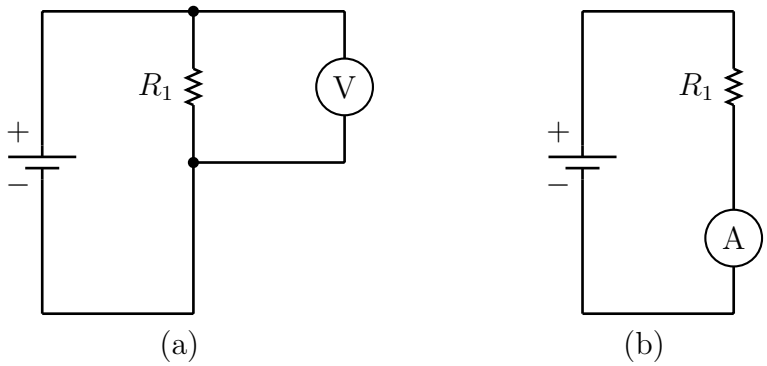


Figure 2.3: Circuits for verifying Ohm's law.

2.5 Voltage Divider

One circuit you will encounter again and again is the humble voltage divider circuit of Fig. **??**a. Modify your setup to include an additional resistor $R_2 = 4.7$ kΩ in series with your resistor $R_1 = 1$ kΩ. Before installing it in your circuit, record the actual value of your resistor R_2 in your log book.

Measurement 4: adjust the supply voltage to 10 V and record the voltage across resistor R_1, the voltage across resistor R_2, and the current through the divider. Compare these measured values to your expectation.

Now adjust your circuit so that R_1 and R_2 are in parallel and set the supply to 10 V **Measurement 5:** Record the voltage across the resistors R_1 and R_2 and the total current through both resistors. Compare the measured current to your expectation.

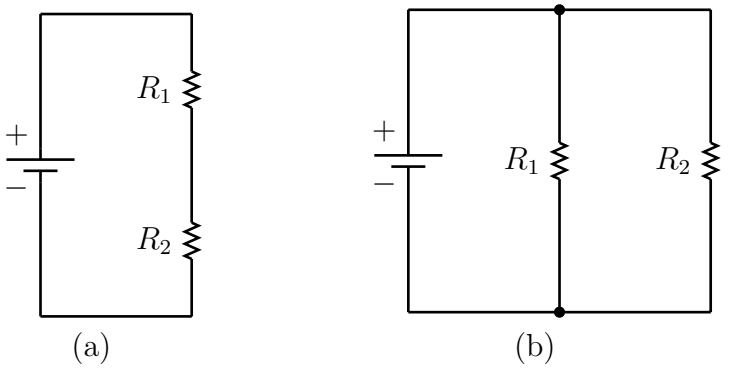


Figure 2.4: Circuits for driving an LED (a) directly from the signal voltage and (b) using a diode switch.

2.6 Δ-Y transformation

Consider the two different circuits shown in Fig. **??**. If we are willing to neglect the central vertex in the left hand circuit, the two circuits are equivalent in the case that $R_1 = R_2/3$. Using your

soldering iron,

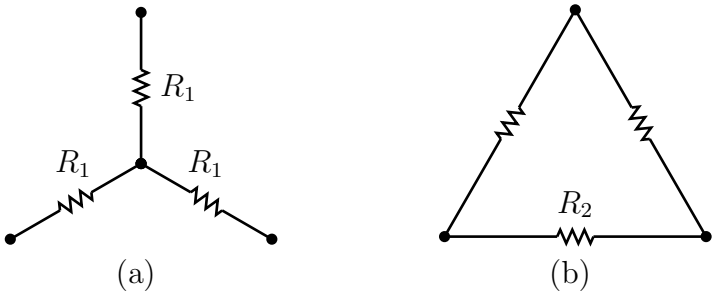


Figure 2.5: Equivalent three-node circuits.

2.7 Additions

Effect of resistance measurement with current in resistor?

Loading of circuit by DMM.

Chapter 3

Thevenin Equivalent Circuits

3.1 Pre-lab Calculation

1) Determine an equation for the Thevenin equivalent voltage V_{th} and resistance R_{th} from the values V_1, V_2, R_1, R_2, R_3 for the circuit shown in Fig. **??**. Hint: Use the superposition principle. Find the equivalent resistance by setting the voltage V_1 and V_2 to zero, i.e. shorting them in the circuit. Then calculate two contributions to the Thevenin voltage, one with V_1 set to zero and one with V_2 set to zero. The actual Thevenin voltage is the sum of these two contributions. Play close attention to the polarity of V_2 as drawn, i.e. that a positive value of V_2 tends to make the voltage V_{ab} negative.

2) Compute V_{th}, R_{th}, and the short-circuit current I_{sc} for the particular values of R_1,R_2,R_3,V_1, and V_2 you will be using in the lab.

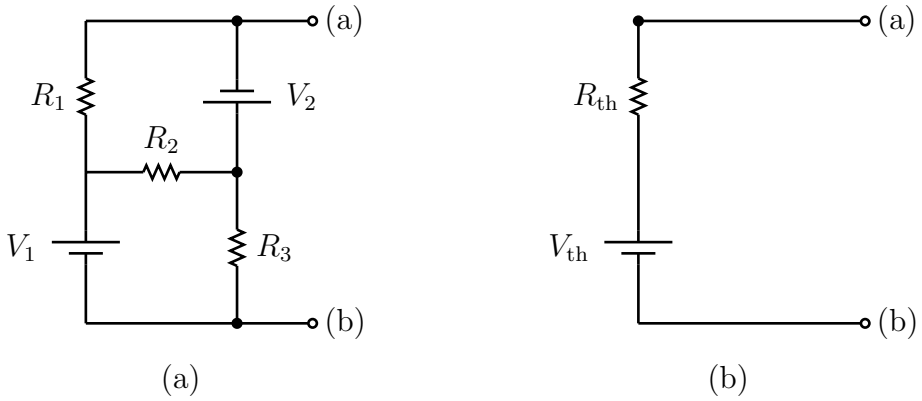


Figure 3.1: The circuit (a) you will be building in lab and it's (b) Thevenin Equilvalent.

3.2 Thevenin Equivalent Circuit

Build the circuit in Fig. **??** using $R_1 = 3.3$ kΩ, $R_2 = 4.7$ kΩ, and $R_3 = 3.9$ kΩ. Supply $V_1 = 10$ V and $V_2 = 5$ V using your two channel bench-top power supply. In the diagram, the supplies are not referenced to ground or each other, so make certain that your supply is set to provide independent outputs and do not add any jumpers to ground. Take careful note of the polarity of the supplies, so

e.g. the negative (black) output of V_1 is connected to point (b) whereas the negative (black) output of V_2 is connected to point (a).

Use your Triplett 9007 as a voltmeter and the Mastech MS8624 as a current meter. First measure the open circuit voltage V_{ab}. Next short the points (a) and (b) through your current meter. These values should closely match the Thevenin voltage and short-circuit current which you have already calculated. If not, you should check your work and find the discrepancy before proceeding.

Next you will measure the voltage across and current through a load resistor connected between the terminals at (a) and (b) to experimentally determine the IV curve for your circuit. Recall from the previous lab that you measure the current by connecting your meter in series and the voltage by connecting your meter in parallel. As before, use your Triplett 9007 as a voltmeter and the Mastech MS8624 as a current meter.

Make simultaneous current and voltage measurements for three different values of the load resistance $R = 470$ Omega, 1.2 kΩ, 4.7 kΩ.

3.3 Analysis

Plot 1: To present your analysis you should produce a part like that of Fig. **??**. Your plot should show the Thevenin equivalent source IV curve for the circuit you built in lab. You should also draw theoretical load IV curves for the three resistor values you used to make current and voltage. Finally, you include data points for the five measurements you made.

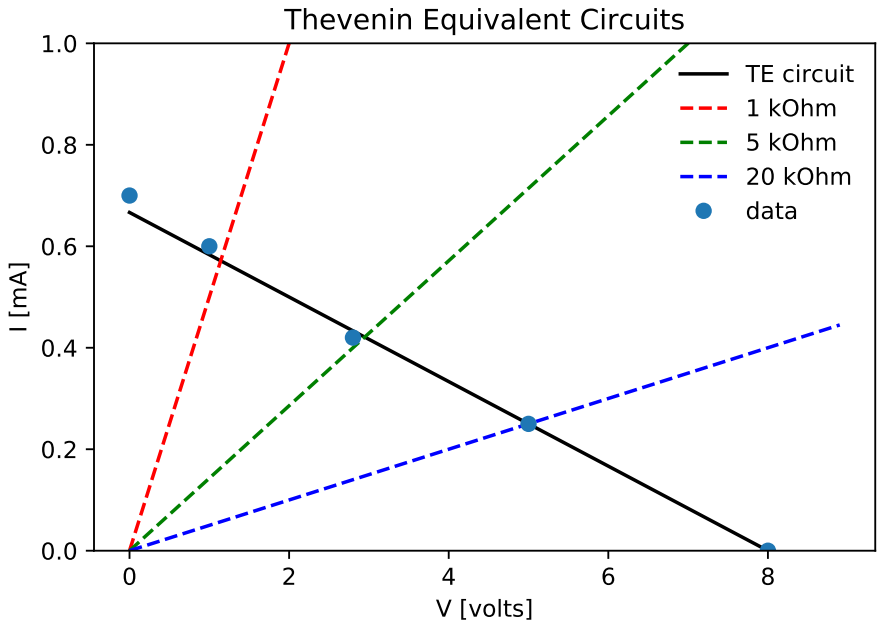


Figure 3.2: Using boolean masks to cut on variable y.

Chapter 4

Alternating Current and Time Varying Signals

4.1 Introduction

In this lab you will use two essential new pieces of lab equipment (scope and function generator) to study alternating current and time varying signals.

4.2 Function Generator

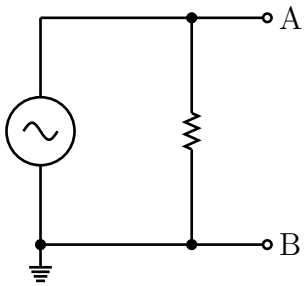


Figure 4.1: A function generator driving a resistor.

Connect the output of Channel 1 directly to the Voltage measurement input of your Triplett 9007 DMM, using a BNC to banana plug adapter as shown in Fig. **??**. Set your DMM to the 20 V AC scale (the V with a squiggly line) .

Now set your function generator to the factory default:

Utility Button → System → Set to Default → Select.

You must perform this step today for the instructions that follow to make sense. With shared equipment, it is essential to know how to restore the factory default, in case another user has left the device with strange settings. You don't need to start with this step every lab, but it is a fast way to recover when you encounter strange behavior.

The factory default settings place the function generator in Sine function mode, so leave that set as is.

Adjust the frequency to 10 kHz by turning the large knob. Press the Freq/Period choice button twice to see how you can switch between specifying frequency and period. Leave it as Frequency for now.

You can likewise switch between Ampl and Offset to specify an amplitude and an offset, or set the high and low values. Adjust the amplitude to 1 V RMS by:

$$\text{Ampl} \rightarrow 1 \rightarrow \text{Vrms}$$

Your function generator should now be set to provide a 10 kHz Sine function with an RMS amplitude of 1 V. Confirm this from the status screen.

Note that your DMM reads 0 V. Because the output is not yet enabled! Press the "On/Off" button above Channel 1 to enable the output.

Turn the output frequency of your function generator down to 2 kHz. Now your DMM should read an RMS voltage near 2 V, about twice the value that we expect. (EXPLAIN)

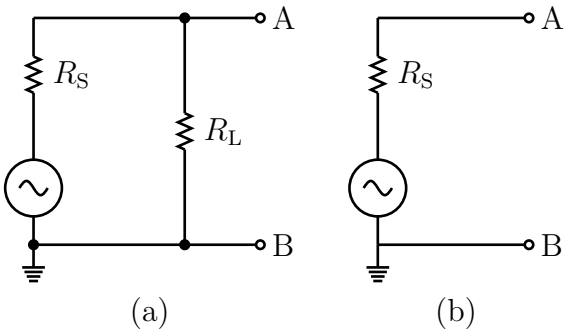


Figure 4.2: A function generator with source impedance explicitly shown.

(Adjust DC offset and measure DC voltage with DMM)

4.3 Oscilloscope

Trigger...

AC coupled versus DC bias...

Grounding???

To observe the time dependence of signals, the tool of choice is the oscilloscope.

Press the default setup function of yours scope...

Adjust the trigger level and observe that the phase of the channel 1...

4.4 Lissajous Figures

Lissajous figures are the graph of system of two parameterized functions:

$$\begin{aligned} x &= A_1 \sin(2\pi f_1 t + \delta) \\ y &= A_2 \sin(2\pi f_2 t) \end{aligned}$$

which produces a closed loop if the ratio A_1/A_2 is rational. The appearance of the figure is of a 3 dimensional knot with the viewing angle determined by the parameter δ. Two examples are shown in Fig. **??**.

To begin, adjust channel 2 of your scope as you did channel 1: probe to 1x, with 500 mV range. Connect the output of Channel 2 on your function generator to the channel 2 input on your ... Set the amplitude of both Channel 1 and Channel 2 to 3 V peak-to-peak.

The relative phase between the two output channels of your function generator shifts whenever you adjust the frequency of one of the signals. For consistent results with offline plots and the scope traces shown here, you'll need to align the phase of the two channels every time you adjust the frequency:

$$\text{InterChbutton} \rightarrow \text{AlignPhase}.$$

You should now have reproduced the "start" pattern from Fig. **??**a.

Adjust the phase of Channel 2 until the pattern collapses into a Fish pattern (or greek letter α). Save a scope trace by inserting your USB drive into the scope and pressing the Save button. Then produce the parabola and lace patterns, according to the settings in the table, saving a scope trace each time. Remember to align the phase each time you change the frequency.

Next, produce the crown pattern, shown in Fig. **??**b. For the right proportions, you'll need to adjust the amplitude of Channel 2 to 1 V peak-to-peak, leaving Channel 1 at 3 V peak-to-peak. Notice that as you adjust the phase of Channel 1, the crown appears to rotate. Adjust the frequency of Channel 2 to 4.0002 kHz. The crown should now appear to rotate constantly at low speed. This is a **sign off** point in the lab.

4.5 Analysis

From the previous section, you should have scope traces for the fish, parabola, and crown. Reproduce each of these figures using scientific python to draw the parameterized shape. For example. Fig. **??**.

One way to approach this problem is to set the period to 1 μs, with fundamental angular frequency $\omega = 2\pi$ kHz.

One way to approach this problem is to set the period to 1 μs. The functions should be evaluated at 1000 discrete times within the interval from 0 to 1 μs.

```
t = np.linspace(0,1,num=1000)
```

Define a fundamental angular frequency $\omega_0 = 2\pi$ kHz:

```
w = 2*np.pi~\rm kHz
```

With these definitions, we would define:

```
x = np.sin(4*w*t)
```

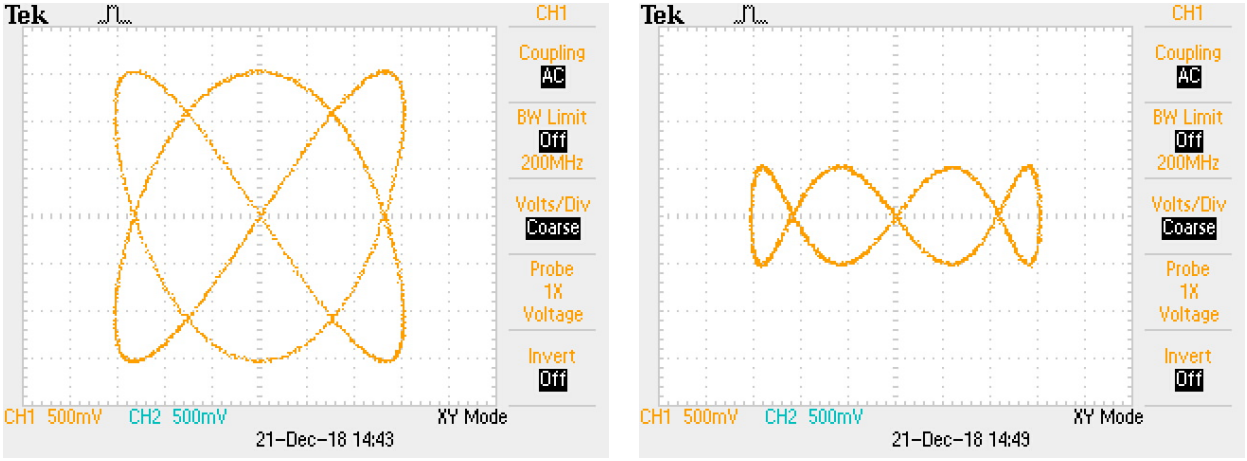
to obtain x points corresponding to $f = 4$ kHz sine function.

When plotting your curves, use:

```
plt.axis('equal')
```

to keep the unit aspect ratio used by your scope.

figs/labs/lissajous/pythonlissajous.pdf



(a) (b)

Figure 4.3: Scope traces from Lissajous figures from settings for (a) start, and (b) crown.

Table 4.1: Settings for various Lissajous figures.

pattern	f_1 (kHz)	f_2 (kHz)	δ_1
start	2	3	0
fish	2	3	30°
parabola	1	2	45°
lace	13	12	0
crown	1 kHz	4 kHz	0

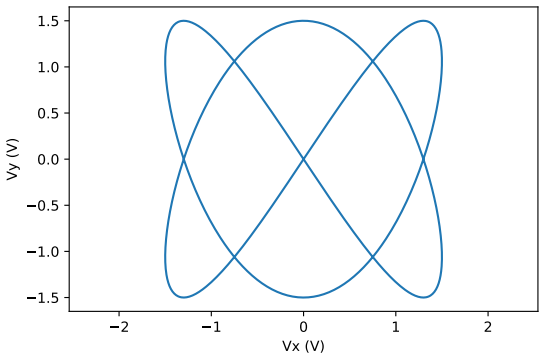


Figure 4.4: Lissajous curve constructed using Scientific Python corresponding to the scope trace in Fig. **??**a.

Chapter 5

RC and RL Transient Signals

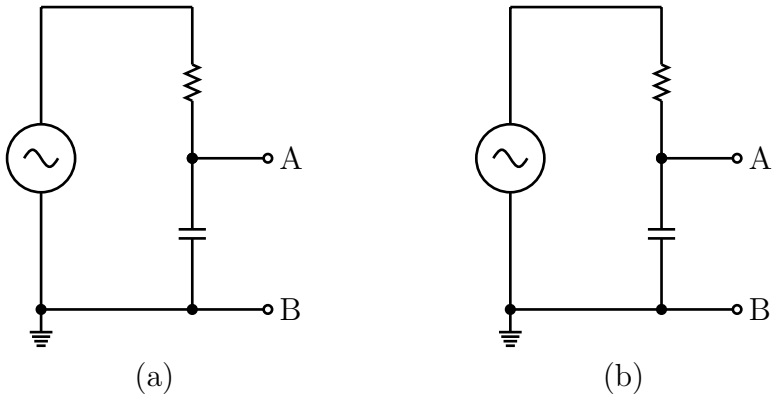
Chapter 6

Passive Filters and Resonance

6.1 Introduction

In this lab you will use two essential new pieces of lab equipment (scope and function generator) to study the transient response of resistors and capacitors.

6.2 RC Low-pass and High-pass Filters



(a) (b)

Figure 6.1: A function generator driving a resistor.

6.3 Resonant Band-pass Filter

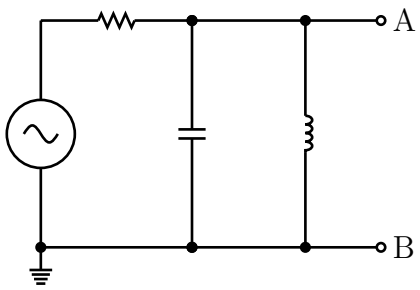


Figure 6.2: A function generator driving a resistor.

Chapter 7

Complete List of Labs

Electron:

1. DC Circuits
2. Thevenin Equivalent Circuits
3. Alternating Current and Time Varying Signals
4. RC and RL Transient Signals
5. Passive Filters and Resonance
6. The Diode

Scientific Python Analysis Labs:

7. Introduction to Plotting
8. Histograms and Distributions
9. The Central Limit Theorem
10. Error Propogation
11. Curve Fitting
12. Fourier Transforms
13. Monte Carlo Techniques

Advanced Labs:

14. Geiger Counter
15. Planck's Constant
16. Speed of Light in Cable
17. Speed of Light in Vacuum
18. Speed of Light Analysis
19. Muon Lifetime Analysis