Physics 80 Lab Manual

February 1, 2019

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. 1.1 which plots a sine function sampled at 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 much 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>
```

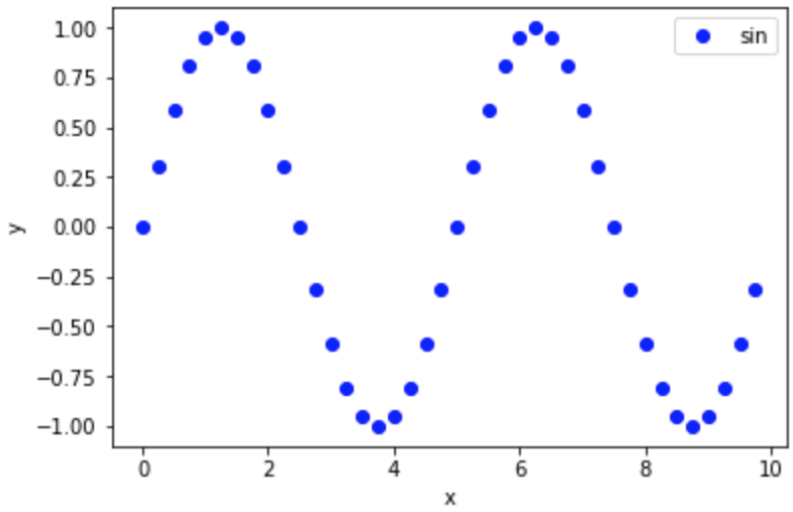


Figure 1.1: Circuit for verifying Ohm's law as a (a) circuit diagram, and (b) implemented using your lab equipment.

- 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. 1.2. 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 1.1 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 of 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. **This is a sign-off point for the lab.** (Each student should be prepared to explain any line in your code.)

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 $[0, 4]$.

The logistics 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. 1.3, 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. 1.4 where you should understand:

- An array of r values is defined.

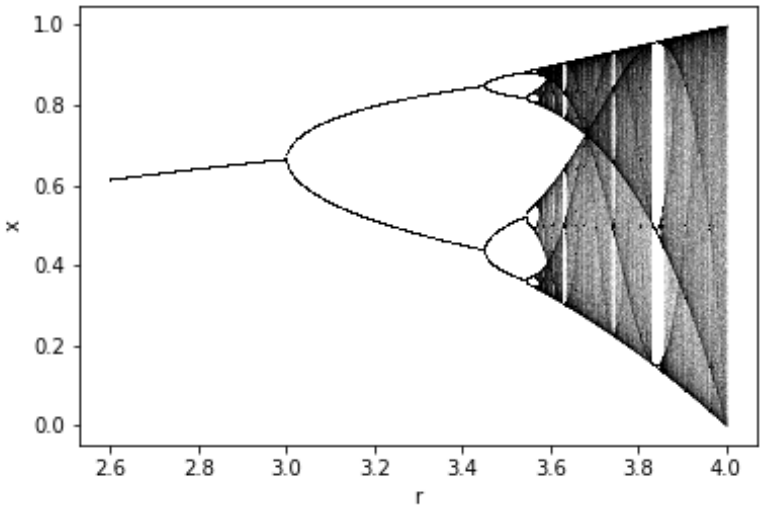


Figure 1.3: Long term behavior of the logistics map.

- 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. 1.3 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')
```

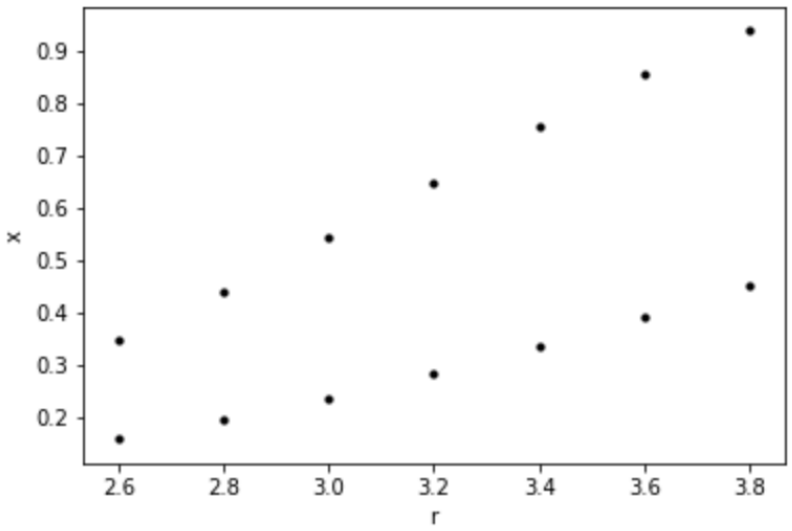


Figure 1.4: Modeling the logistics map.

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.

If time permits, you will also solder two resistor circuits to explore the Δ-Y transformation for three terminal networks.

2.2 Benchtop Power Supply

Figure 2.1: Your bench-top DC power supply, the MASTECH 3005F-3.

In this lab you will use one channel of your MASTECH 3005F-3 DC power-supply as the voltage source in your experimental circuits. See Fig. 2.1 for the location of the control features. Since you won't know initially what settings your power-supply was left at, leave the supply disconnected while you configure the supply to provide 10 V.

First, check that the supply is set to provide two independent outputs (you will only use one for this lab) by making sure both bush buttons are out. Next, power the device by pressing the power button. Your power supply has both a current limit knob and a voltage limit knob. Usually, we specify the voltage you want in your circuit, but even in this case, the current limit is useful for protecting your circuit and measurement equipment. When the LED labeled "CV" is lit, the voltage limit is controlling the output. When the LED labeled "CC" is lit, the current limit is controlling the output.

Turn the voltage limit knob clockwise and watch the voltage increase on the display. If the CC LED lights, it indicates that the current limit is active, and you will not be able to raise the voltage. Turn the current limit knob until just a bit after the CV LED lights, indicating the voltage limit is active. Continue raising the voltage until you reach the 10 V. Once you reach your target voltage, turn the current limit counter-clockwise, to lower the current limit, until the LED labeled "CC" lights, indicated you have reached the current limit. Turn the current limit clockwise just past the point that the device goes back to being voltage limited. If you get in the habit of doing this every time you use your bench-top supply, you will safe many components for accidental damage!

The voltage on the display is maintained between the black (-) terminal and the red (+) terminal. It is floating, meaning that only the difference is set by the device, just like a battery, and you may connect ground wherever you choose (within the limits of the supply). If you wish to provide a ground referenced DC voltage, you connect the green terminal to the appropriate terminal. For example, connecting green to red would provide -10 V referenced to ground at the negative terminal. You will leave the supply floating for this experiment.

2.3 Voltage Measurement

Your primary digital multimeter (DMM), the Triplett 9007 shown in Fig 2.2, can be used to make a number of measurements including resistance, DC current, and DC voltage. We'll measure a DC voltage to start. This lab will be most convenient if you use alligator clip probes in your DMM. Install a black alligator clip probe in the "Common" terminal, and a red alligator clip probe in the "Voltage" terminal directly above the Common terminal. Clip the alligator clips together so that you expect to measure 0 V. Now power the DMM by pressing the power button, and turn the voltage dial to the DC voltage measurement, the V with one straight and one dashed line next to it. Most measurements on your DMM have a number of different full range settings. For the highest precision, you should use the smallest full-range setting larger than your measurement. We'll be measuring 10 V, so use the 20 V (DC) setting.

With the alligator clip probes connected together, your DMM should read 0.00 V. Now touch the probes to the terminals on your bench-top DC voltage supply: red probe to red terminal, and black probe to black terminal. You should read 10 V. Next touch red probe to black terminal and black probe to red terminal. You should read -10 V. This is because the "Common" (black) probe is the reference point for the measurement, and the red probe is currently connected to a point 10 V lower than this reference point.

Figure 2.2: Your (a) digital multimeter (DMM), the Triplett 9007, and (b) alligator clip probes.

2.4 Resistance Measurement

Pick two different resistors from the random collection of resistors at the front of the lab. Connect the alligator clip across one of the resistors, and turn the dial to the resistance measurement, marked Ω. When the measurement reads "1" the resistance is larger than the full range. Find the smallest range larger than your resistor and record the measured value. Using the resistor color coding guide in Fig. 2.3, determine the nominal resistance of your selected resistor. In your logbook, record your resistor color pattern, the nominal value you determined, and the measured value. Repeat this for a second resistor.

2.5 Current Measurement

Your DMM is designed to have minimal impact on the circuit your are measuring. When used to measure the voltage across a component, such as a resistor, it is connected in parallel, as in Fig. 2.4a. The ideal voltmeter therefore has infinite resistance, so that no current from the circuit is diverted through the voltmeter. Your DMM has a 10 MΩ resistance when used as a voltmeter.

When used to measure the current through a component, your DMM is connected in series, allowing the current to pass through the device. The ideal ammeter therefore has zero resistance, so that there is no voltage drop across the voltmeter. For this reason, most DMMs have separate terminals for measuring currents.

You make a current measurement with a DMM by installing the red probe in a current terminal (instead of the voltage terminal) the black probe in the common terminal as before, and installing the probe in series with the component you wish to measure the current through.

However, the current measurement in your Triplett 9007 is fused, and you can fairly safely assume that the fuse is blown. Because the current measurement is nearly a short circuit, it is very

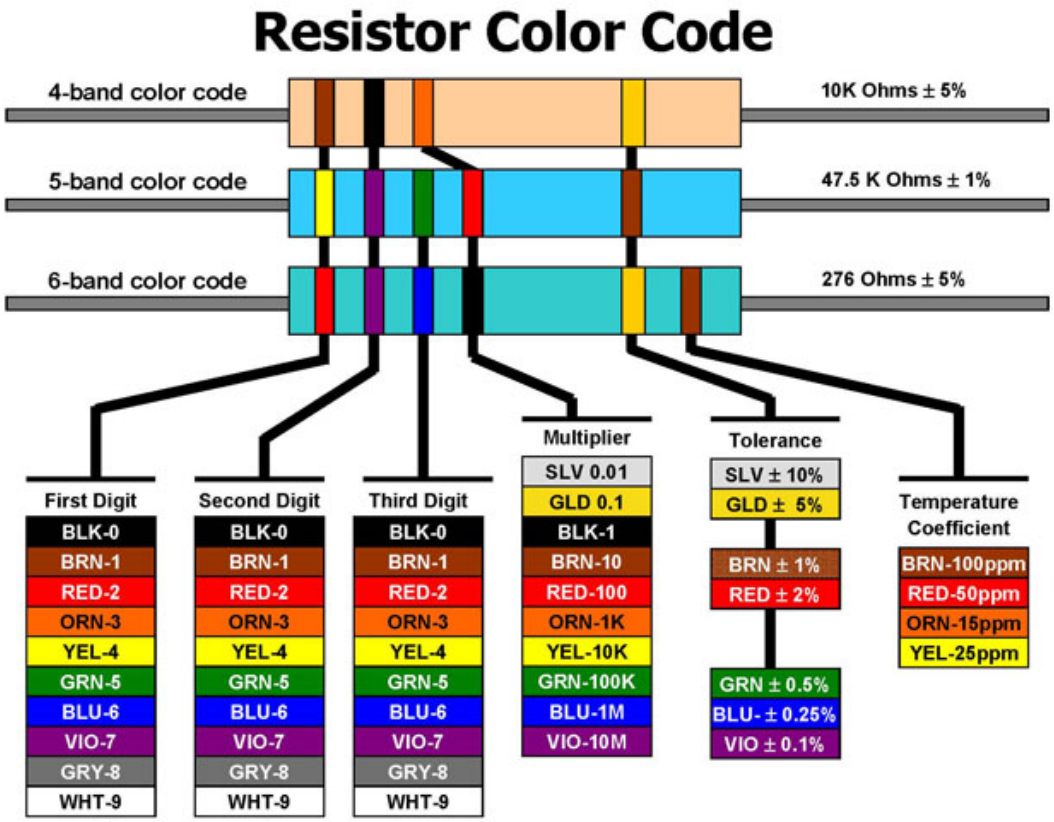


Figure 2.3: The resistor color code.

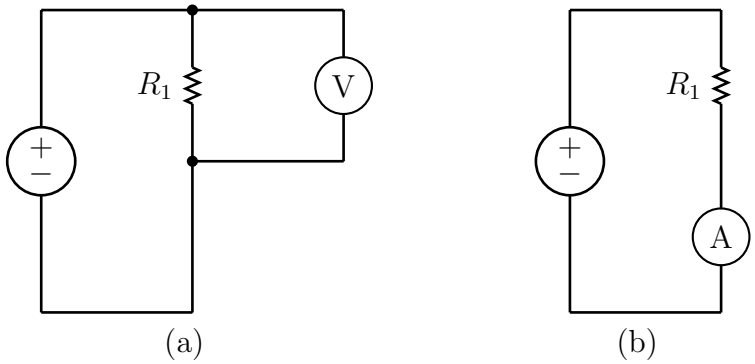


Figure 2.4: Appropriate connections for measuring (a) the voltage across resistor R_1 and (b) the current through the resistor R_1.

easy to introduce an unintentionally large current by connecting a voltage across the terminals, as if to measure a voltage. Even experience is no sure remedy for this common lab mishap.

Instead, we will use the Mastech MS8264 which features a resettable fuse for making current measurements.

2.6 Breadboard

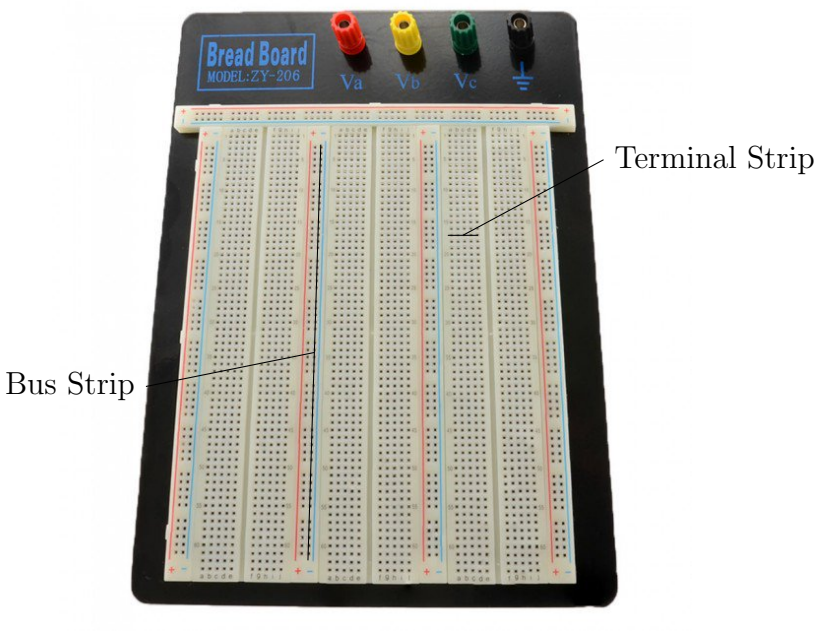


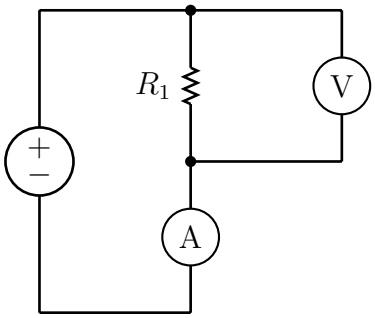
Figure 2.5: A typical breadboard.

Breadboards are a convenient way to prototype circuits without having to solder. When the leads of discrete components are inserted in the holes in the breadboard, they make electrical contact with metal strips inside the breadboard that connect with additional holes. The short terminal strips are used to make electrical connections between components. The longer bus strips are a convenient way to bring in ground or voltage supplies that need to connect to many places in circuit.

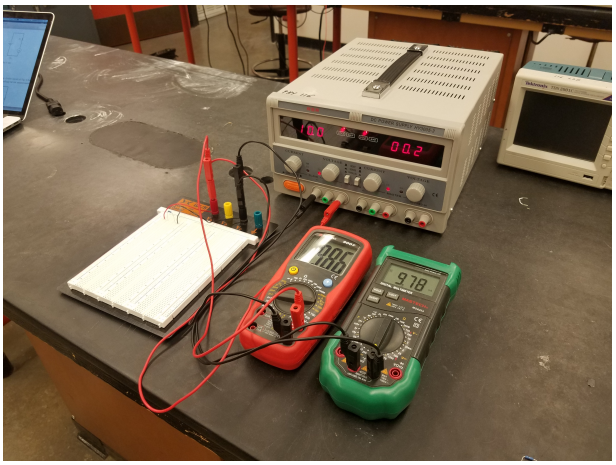
2.7 Verification of Ohm's Law

Build the circuit in Fig. 2.6. 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 DC voltage. Use banana plug connectors for the current measurement, installing the Mastech MS8624 between the supply and the breadboard.

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.



(a)



(b)

Figure 2.6: Circuit for verifying Ohm's law as a (a) circuit diagram, and (b) implemented using your lab equipment.

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?

2.8 Voltage Divider

One circuit you will encounter again and again is the humble voltage divider circuit of Fig. 2.7 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 current through each resistor. Compare the measured currents to your expectation.

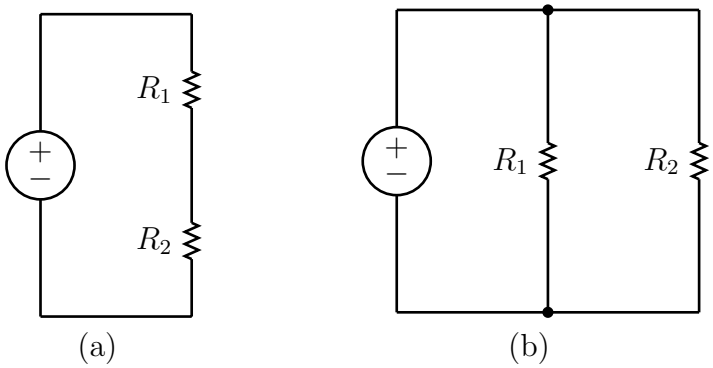


Figure 2.7: Circuits for studying resistors (a) in series, and (b) in parallel.

Chapter 3

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. 3.1. 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.

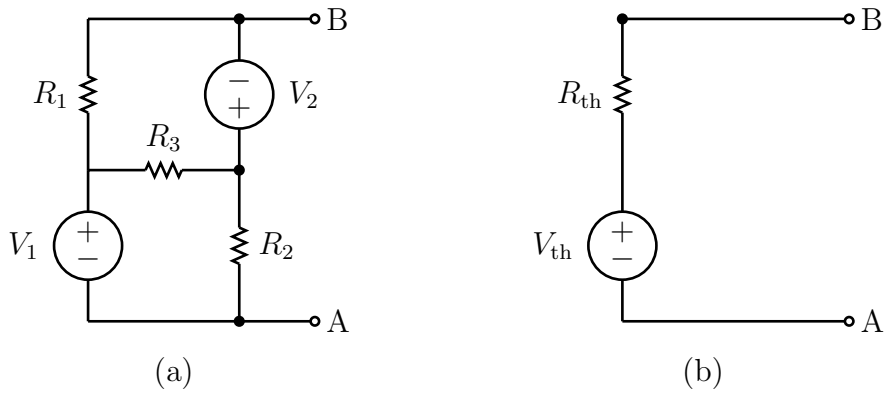


Figure 3.1: The circuit (a) you will be building in lab and it's (b) Thevenin Equilvalent.

3.2 Thevenin Equivalent Circuit

Build the circuit in Fig. 3.1 using $R_1 = 3.3$ kΩ, $R_2 = 3.9$ kΩ, and $R_3 = 4.7$ 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 $R = 470\ \Omega, 1.2\ \text{k}\Omega, 4.7\ \text{k}\Omega$.

3.3 Analysis

Plot 1: To present your analysis you should produce a part like that of Fig. 3.2. 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.

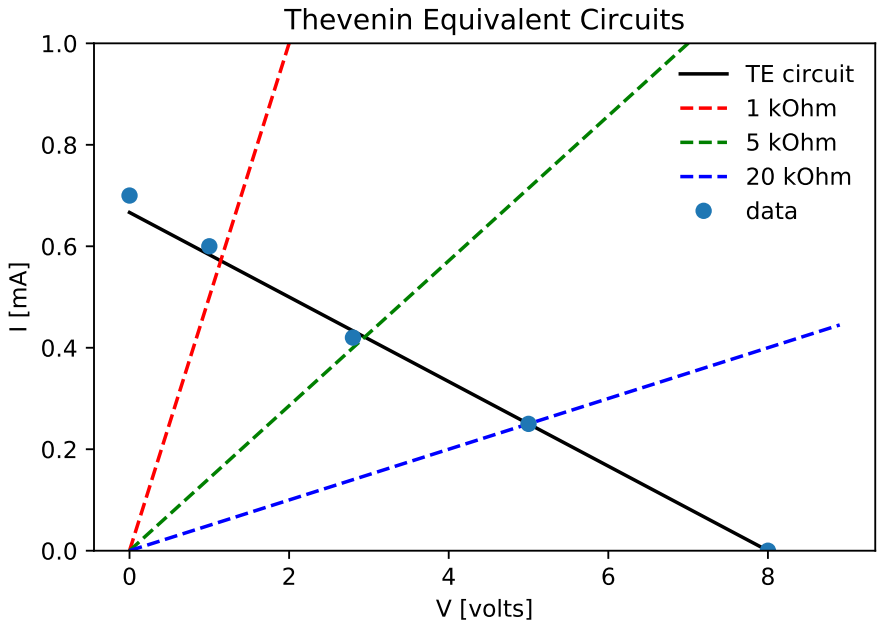


Figure 3.2: Using boolean masks to cut on variable y.

3.4 $\Delta - Y$ Transformation

Consider the two different networks shown in Fig. 3.3. If there are no external connections to the central node in the left-hand circuit, the two networks are equivalent if:

$$R_A = \frac{R_{AC}R_{AB}}{R_{AB} + R_{AC} + R_{BC}}$$

as well as two similar equations for R_B and R_C. Going in the other direction we have:

$$R_{AB} = \frac{R_A R_B + R_A R_C + R_B R_C}{R_C}.$$

These transformations are more general than the series and parallel laws, which you can derive by considering the case that $R_{BC} = 0$ for parallel resistors, and $R_C \to \infty$ for series resistors. They allow one to simplify more complicated networks for which the series and parallel equivalence relations are insufficient.

In the special case that $R_A = R_B = R_C = R$ it follows that

$$R_{AB} = R_{AC} = R_{BC} = 3R.$$

If time permits, use your soldering iron to construct the left-hand network using $R_A = R_B = R_C = 1$ kΩ. Then construct the equivalent right-hand network using $R_{AB} = R_{AC} = R_{BC} = 3.0$ kΩ. Since 3 kΩ is not a standard sized 10% resistor, you can construct one by using a 33 kΩ in parallel with a 3.3 kΩ resistor.

Make sure the soldering iron is on, and the sponge is moist. Twist the leads of the resistor together to make initial connections, then hold the arrangement securely in the clamp. Wipe the tip of the hot iron on the sponge to clean it, then apply a small amount of solder to the tip by touching the hot iron to the solder wire.

Heat the connection by holding the soldering iron against it, then bring the solder wire in contact with the heated connection (not the soldering iron) You want the iron to heat the connection, and then the connection to melt and draw in the solder. The little bit of solder on the tip is only there to ensure good thermal conduct between the tip and the connection: don't "paint" the solder onto the connection.

Measurement 1: Check the resistance between pairs of terminals on your creations, and compare with your expectation. You can bring your creations home if you like.

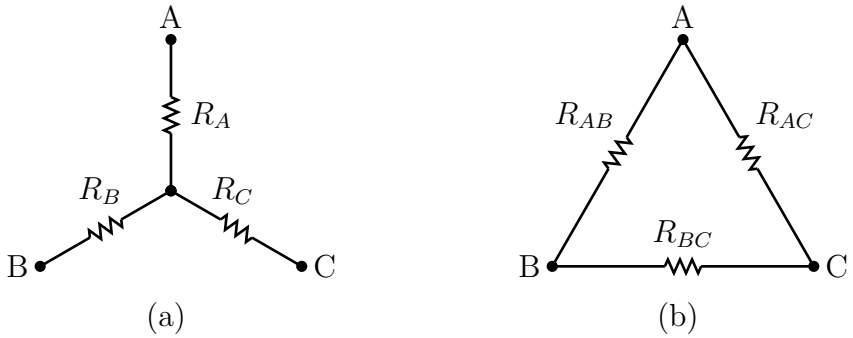


Figure 3.3: Equivalent three-node circuits.

Chapter 4

Alternating Current and Time Varying Signals

4.1 Introduction

In this lab you will use two essential new pieces of lab equipment: the digital oscilloscope and the function generator. You will learn how to measure AC voltages with your DMM, as well as how to view time-dependent wave forms on your digital oscilloscope. You'll view Lissajous curves in 2-D using the X-Y mode of your scope and reproduce the same curves using parameterized equations in Scientific Python.

4.2 Function Generator

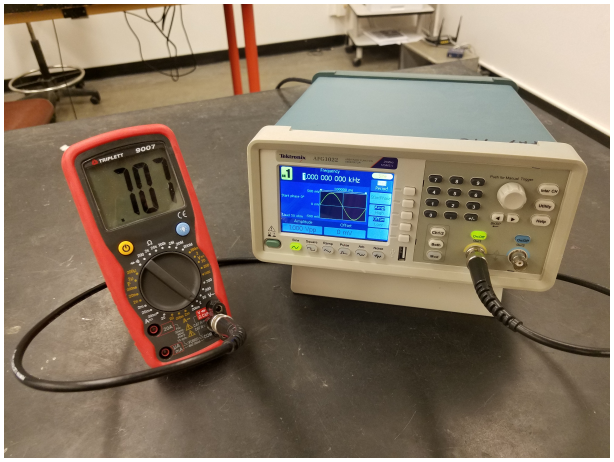


Figure 4.1: Connect the Channel 1 Output of your function generator directly to your DMM.

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. 4.1. Turn on power to the function generator. Then 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 in your equipment.

The factory default settings are set to produce a Sine function with a peak-to-peak voltage $v_{\rm pp} = 1.0$ V and a frequency $f = 1$ kHz. We'll leave that as is for now. To turn on the output, push the "On/Off" directly above the coaxial output for Channel 1, and then ensure that the button is lit.

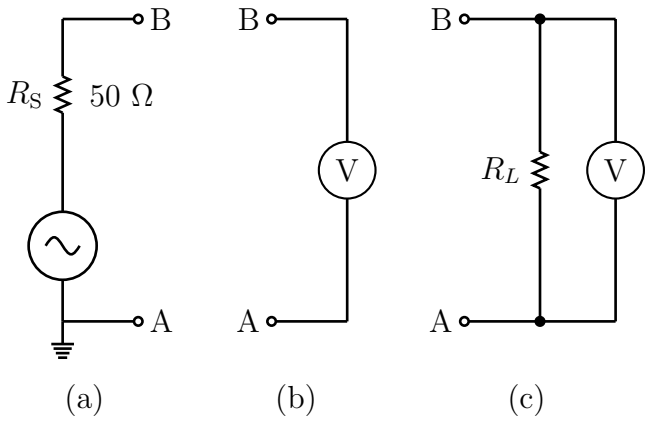


Figure 4.2: Equivalent circuit for (a) your function generator, which includes a 50 Ω source resistance, and two typical terminations for a coaxial signal: (b) infinite resistance voltmeter or scope, or (c) a terminating resistorin parallel.

Set your DMM to the 2 V (AC) scale (the V with a squiggly line). And you should now measure the AC voltage

$$v_{\rm rms} = 0.707 \text{ V} \sim \frac{1}{\sqrt{2}} \text{ V}$$

However, recalling the relationships between the peak-to-peak voltage, the peak-voltage, and the RMS voltage of an AC sine wave:

$$v_{\rm pp} = 2\, v_{\rm p} = 2\sqrt{2}\, v_{\rm rms}$$

we expect our function generator, set to $v_{\rm pp} = 1.0$, to produce output with:

$$v_{\rm rms} = \frac{v_{\rm pp}}{2\sqrt{2}} \sim 0.353 \text{ V}$$

Clearly someone is lying to us! In fact, we've encountered a very common source of factor of two mistakes. The equivalent circuit for your function generator is shown in Fig. 4.2a. Notice that it includes a 50 Ω source resistance in series with the AC voltage produced by the function generator. This internal resistance is important for a number of reasons, most notably making it impossible to short-circuit the output and destroy the equipment!

In our setup, we've connected the function generator output directly to your DMM, which has a very high input resistance, effectively infinite, as shown in Fig. 4.2b. However, the standard termination for coaxial cables is 50 Ω, and the default setting for your function generator expects the load shown in Fig. 4.2c with $R_{\rm L} = 50$ Ω. In this case, the internal resistance and load resistance form a voltage divider, so that the output voltage $V_{\rm AB}$ seen by the user is 1/2 the internal AC

voltage. The function generator is designed to produce an internal AC voltage which is twice the value selected by the user, so that the output voltage is precisely the value specified by the user. We are seeing twice our requested value, because we have no load resistor, and so no voltage divider, and instead see the full value of internal AC voltage. To fix this discrepancy, we simply have to configure our generator to expect a high load resistance at both outputs:

$$\begin{aligned}\text{Utility Button} \rightarrow \text{CH1Load} &\rightarrow \text{HighZ} \\ \text{CH2Load} &\rightarrow \text{HighZ}\end{aligned}$$

Press the "Ch1/2" button until you return to the Channel 1 menu. Adjust the amplitude to 1 V peak-to-peak by:

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

Your DMM should now read the expected value:

$$v_{\text{rms}} \sim 0.353 \text{ V}$$

Press the button next to Ampl a couple of times. There is a slightly annoying feature of your function generator which allows you to specify either the Amplitude and Offset or the High and Low voltage values. So if you want to adjust the Amplitude, you have to press the button next to Ampl until the Ampl label is highlighted. Often you'll end up setting the wrong value by mistake. But in general, whatever parameter is highlighted along the side of the screen is the parameter which you can specify by either the knob or the key pad. Keeping this is mind, set your function generator to produce 1 V RMS output:

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

Now your DMM should also read a value quite close to one.

Now let's adjust the frequency. Highlight the frequency parameter by pressing the button next to the "Freq" option until it is highlighted:

$$\text{Freq} \rightarrow 10 \rightarrow \text{kHz}.$$

You can also adjust the selected parameter with the multipurpose knob. Turn the multipurpose knob until the frequency is around 100 kHz and observe what happens to your DMM measurement. The reason your measurement is now inconsistent with the setting in the function generator, is that your DMM is only rated to 2 kHz. It isn't intended for measuring high-frequency AC signals. Turn the frequency back down to 1 kHz.

Next highlight the Offset parameter on your function generator and adjust it to 2 V. This will add a DC offset to your function generator output. After settling down, the measured value of the AC voltage on your DMM should be unchanged at 1 V. Switch your DMM to measure the DC voltage and you should now measure the 2 V DC offset. Pay attention to the sign. If you see a negative value, it is because you installed your BNC-to-banana adapter incorrectly. Notice that one side of the adapter has a small raised tab, indicating which side connects to the coaxial cable shield. The side with the raised tab should be plugged into the Common socket. Whether you got lucky this time or not, change the orientation of the adapter a few times and observe how the sign of the voltage changes, finally plugging it back in with the correct orientation. Now adjust the DC level with the multipurpose knob and observe the change on your DMM. When satisfied, set the offset back to zero.

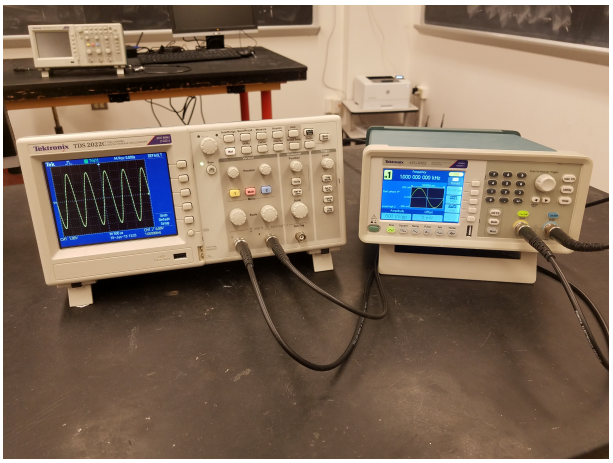


Figure 4.3: Connect the Channel 1 Output of your function generator directly to your DMM.

4.3 Oscilloscope

Put your DMM aside. Connect the Channel 1 output of your function generator to the Channel 1 input of your digital oscilloscope. Do the same for Channel 2. The setup is shown in Fig. 4.3. Set your function generator to provide a 1 kHz sine wave with peak-to-peak voltage of 600 mV. (A DC offset of zero is implied unless otherwise stated) Press the "Default Setup" button on your digital scope. You should immediately observe a sine wave on your Digital scope just as in Fig. 4.3.

Press the button labeled "Square" on your function generator to change the output from a Sine wave to a square wave and observe the waveform on your scope. Do the same for the Ramp and Noise functions. Then return to a Sine wave.

Press the yellow button labeled "1" several times. This button turns on and off the display of channel 1, and brings up the Channel 1 parameter menu. Notice that the voltage scale for Channel 1 is indicated as 1.0 V. This means that the difference between each pair of consecutive horizontal lines corresponds to 1 V. We say "One volt per division". By counting divisions, you should be able to see that your waveform has a peak-to-peak voltage of 6 V. Yet your function generator is set to produce 600 mV = 0.6 V. Clearly someone is lying to us!

Although we won't be using them in this lab, most sensitive measurements with an oscilloscope are made using a scope probe, as shown in Fig. 4.4. To protect the circuit being measured from being effected by the insertion of the probe, there is usually a large resistance in the probe. This means that the oscilloscope itself measures the output of a voltage divider, and the signal is attenuated, most often by a factor of 10. The oscilloscope simply adjusts the voltage scale so that values you read are not attenuated. To make consistent measurements, you simply have to make sure that the oscilloscope is configured for the attenuation factor we are using.

In our case, we are connecting coaxial cables directly between the oscilloscope and the function generator, and so their is no attenuation. But the default setup for the scope assumes that you are using a probe with a 1/10 attenuation, called a 10X probe. Look at the options next to the menu buttons and find the one that says "Probe 10X Voltage". Press this menu button, and then press the Attenuation button until it reads 1X, appropriate for a coaxial cable with no attenuation factor. The waveform is unchanged, but now the voltage scale is correctly set to 100 mV. And your signal now appears to be 600 mV, consistent with the setting from your function generator, as shown in Fig. 4.5. Next turn the knob labeled "scale" located under the yellow channel "1" button.

Figure 4.4: An example scope probe.

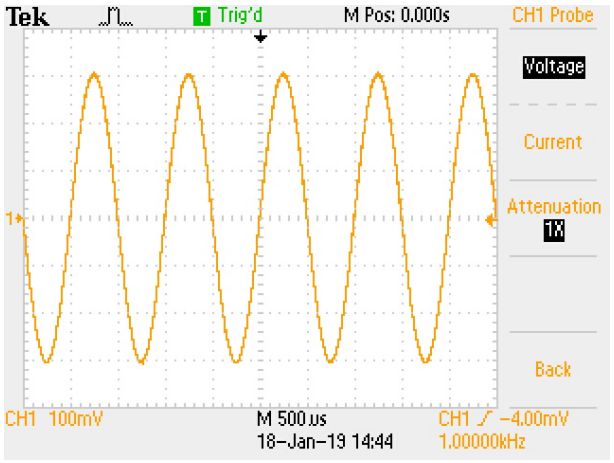


Figure 4.5: Correctly scaled scope output.

Adjust this knob until the scale for CH1 is listed as 200 mV per division. The apparent size of the waveform will be reduced by a factor of two, because each division is now 200 mV and so your 600 mV signal appears three divisions high.

Next note that the function repeats every two divisions. Since the time scale is listed as 500 μs, the period is therefore 1 ms, corresponding to a frequency of 1 kHz. Adjust the time scale, using the large knob in the Horizontal column, until the time scale is 100 μs per division. This is still a 1 kHz signal, but one period now takes up the entire display.

Using the multipurpose knob on your function generator, adjust the frequency up to 10 kHz, and observe how the waveform changes. Then adjust the voltage between about 100 mV and 2 V peak-to-peak. When finished, leave the function generator producing a 5 kHz sine wave with 600 mV peak-to-peak voltage. Your scope should remain at a voltage scale of 200 mV and time scale of 100 μs. Next, set the DC offset of the signal on the function generator by pressing:

$$\text{Offset} \rightarrow 10 \rightarrow \text{mV}$$

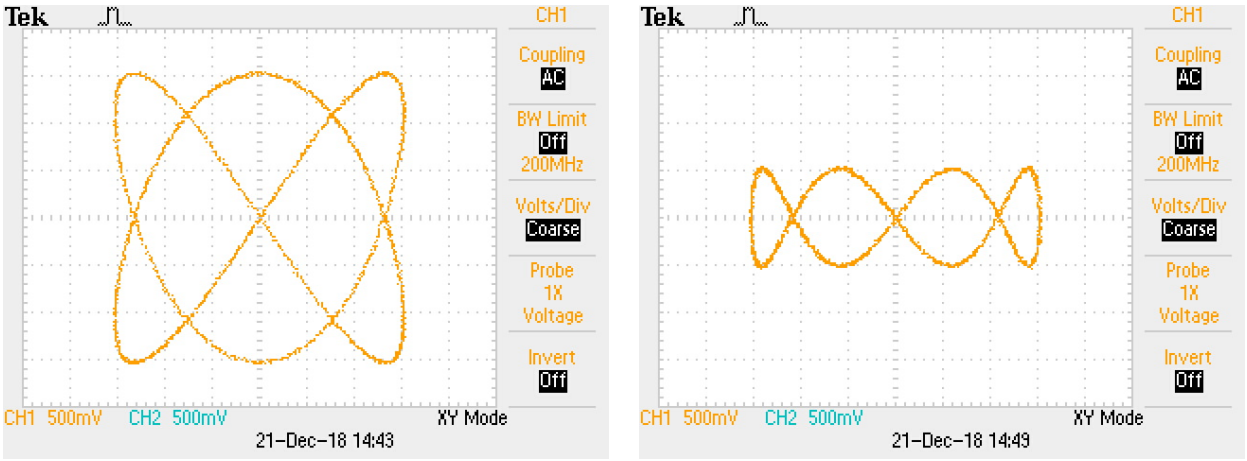
Turn the multipurpose knob to adjust the DC offset between -100 mV and 100 mV. Your waveform will rise and fall on your display. By default, your scope includes the DC offset, but often this is not what you want. On your scope, press the button labeled "Coupling DC", until the DC becomes AC. When AC coupled, the DC component of your waveform is ignored. When AC coupled, observe that changing the DC offset on the function generator does not change the position of the waveform.

You can adjust the position of the waveform using the small knobs labeled "Position" to adjust the offset in vertical and horizontal. Try this out. To return a waveform to (0,0), notice that the offset is displayed while you are turning the knob.

On your function generator, set the output to a 5 V peak-to-peak sine wave with frequency of 100 kHz. Adjust the voltage scale and time scale until you can clearly see the sine wave.

You now know most of the key features of your scope apart from the trigger, which we'll leave for another day!

4.4 Lissajous Figures



(a) (b)

Figure 4.6: Scope traces from Lissajous figures from settings for (a) start, and (b) crown.

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. 4.6.

To produce these figures on your scope, we'll need to use two channels. To begin, enable the output of both Channel 1 and Channel 2 on your function generator, and set them both to produce sine waves with amplitude 3 V peak-to-peak. Adjust the frequency of channel 1 to 2 kHz and the channel 2 to 3 kHz. Note that you can switch between the Channel 1 and Channel 2 parameter menus with the button labeled "Ch1/2".

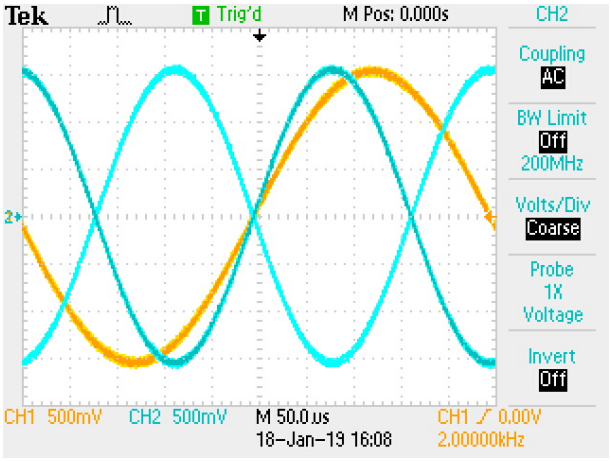


Figure 4.7: Correctly scaled scope output.

On your scope, switch to the Channel 2 parameter menu by pressing the blue button labeled "2". Set the coupling of Channel 2 to AC, and probe attenuation to 1x, just as you did previously for Channel 1. Next adjust the voltage scales of each channel to 500 mV and set the common time scale to something appropriate, so that you can view both Sine waves on the display. As shown in Fig. 4.7, you will see two versions of the Channel 2 output, inverted with respect to each other, because the frequency of Channel 2 is 1.5 times the frequency of Channel 1.

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 on the function generator:

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

Usually, scopes are used to display the inputs as a function of time. In this case, the voltage level is along the y-axis, and time is the x-axis. This mode is called YT mode. Occasionally, however, it is useful to display things in XY mode. In this mode, the x-axis is used for the voltage of Channel 1 and the y-axis is used for the voltage of Channel 2. Each point on the curve represents a particular point in time. Switch to XY mode by pressing the Display button and then pressing the button next to the Format menu item until the mode is XY. You should reproduce Fig. 4.6a exactly. If

Table 4.1: Settings for various Lissajous figures.

pattern	f_1 (kHz)	f_2 (kHz)	δ_1
start	2	3	0
fish	2	3	135°
parabola	1	2	45°
lace	13	12	0
crown	1 kHz	4 kHz	0

not, check that you have aligned the phase as described above and that frequencies are set correctly as in Table.

Adjust the phase of Channel 2, under menu item StartPhase, until the pattern collapses into a Fish pattern (or greek letter α) at 135 degrees. 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 Table 4.1, 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. 4.6b. For the right proportions, 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. 4.8.

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
```

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. You can display your scope traces in python using the Image library like this:

```
import Image

image = Image.open('myscope.jpg')
image.show()
```

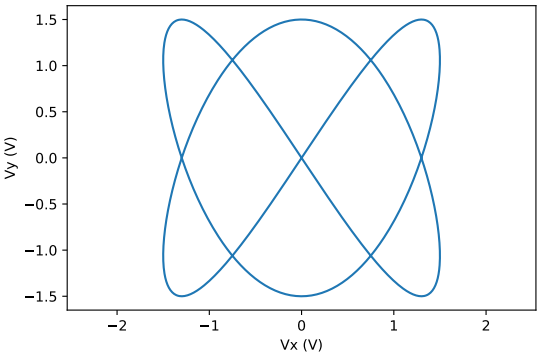


Figure 4.8: Lissajous curve constructed using Scientific Python corresponding to the scope trace in Fig. 4.6a.

Chapter 5

RC and RL Transient Signals

5.1 Pre-lab Calculation

1) Show that for an exponential decay with time constant τ, the rise-time, when defined as the time interval between 10% and 90% values, is given by:

$$t_{90} = \ln(9)\ \tau \sim 2.2\ \tau$$

2) Calculate the inductance of a solenoid with N=20 turns, length $\ell = 4$ cm, a radius of 1 cm^2 using the formula:

$$L = \frac{\mu_0 N^2 A}{\ell}$$

where A is the cross-sectional area and $\mu_0 = 1.257 \times 10^{-6}$ H/m.

5.2 Oscilloscope Trigger

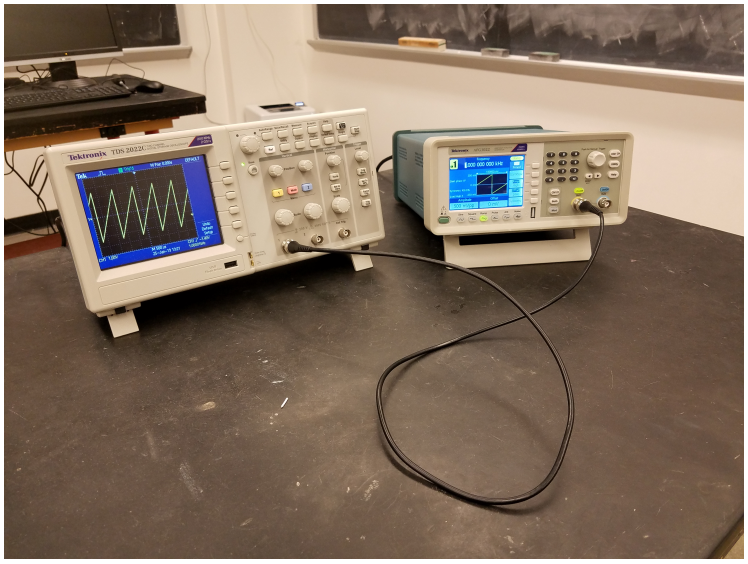


Figure 5.1: Initial setup.

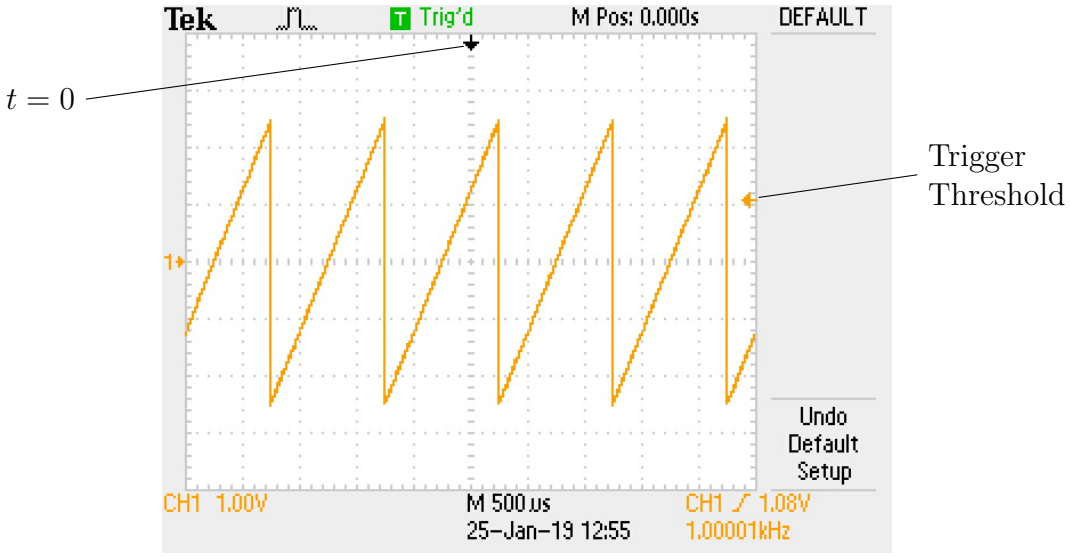


Figure 5.2: A scope trace triggered by the rising edge of ramp function. The point $t = 0$ is the location where the waveform first crosses the trigger thresholds, which is indicated by the arrow to the right.

Connect the output of Channel 1 on your function generator directly to Channel 1 of your oscilloscope with a BNC cable, as in Fig. 5.1. Set your function generator to produce a 1 kHz Ramp function with a peak-to-peak amplitude of 600 mV. While the Ramp function is selected, you will have an option "Symmetry" available. Press the menu button next to Symmetry and set the value to 100%. Turn on your oscilloscope and press the "Default Setup" button. Recall from last week that you need to remember to (1) set your function generator to expect high-impedance output, (2) enable the function generator output by pressing the "On/Off" button above the BNC connector for the channel, so that button is lit (this is the last time these steps will be explicitly mentioned.) The ramp function should be immediately visible on your scope. Adjust the attenuation factor of Channel 1 to unity (X1) appropriate for BNC cable and confirm the peak to peak amplitude and the period are as expected (Leave Channel 2 alone for now.)

Your scope is continuously updating the display with the most recently collected waveform, which at 1 kHz is effectively instantaneous. You may have wondered how these images all appear on the oscilloscope with the exact same phase. This crucial feature of your scope is a result of its trigger feature, which we'll explore now. Turn the knob labeled "Trigger Level" and you should see the phase of your waveform change.

As shown in Fig. 5.2, your scope display has an arrow at the top of the display which indicates the point at which the trigger condition was met for the current waveform, which is taken as $t = 0$. A second arrow, along the right side, indicates the trigger threshold. Your scope is configured to trigger on the rising edge, which means it will trigger at the instant where the trigger first goes above the trigger threshold.

As you adjust the trigger threshold, the point at which the waveform reaches this threshold changes. Since $t = 0$ is defined by the instant at which the trigger condition is met, this effectively changes the phase of your waveform. As you adjust the threshold up and down, the waveform moves forward and backward along the time axis. This is an example of an effect called "time-slewing", which is important to consider when making time dependent measurements of wave forms, as we will be doing today.

Notice what happens if you dial the threshold until it exceeds the amplitude of your waveform. Suddenly, your waveform will appear to dance across the screen. This is because your scope trigger is set to "Auto" trigger. In this trigger mode, when too much time has passed without the trigger condition being met, a waveform is displayed for the current input using a random time for $t = 0$. This can be convenient for finding a signal of unknown size and location. Press the "Trigger Menu" button, and then set the Trigger Mode to "Normal" by pressing the corresponding menu button. In this Mode, the waveform is not update until a trigger is received. While the scope is in the Normal trigger mode, adjust the threshold so that it is above the amplitude of the waveform. You will notice that when the trigger condition is not being met, the scope indicates the state "Ready" at the top of the screen, meaning that the scope is armed to collect data when the trigger condition is met, but there has not recently been a trigger. Now move the threshold below the amplitude for the waveform, and you should see the scope indicates the state "Trig'd", or triggered, on this display. A common mistake for students is to look at stale data, without realizing that the signal is lost. Avoiding this pitfall is one benefit of the Auto trigger mode.

At times you may wish to start and stop the data acquisition. This can be accomplished using the "Run/Stop" button to pause and then resume waveform acquisition. Try this feature out now. You can also capture and display one single wave form by pressing the "Single Seq" button. This can be handy when looking at things like detector pulses, which are different each time, and which may occur at a low rate. Resume normal data acquisition with the "Run/Stop" button.

Your scope trigger has the capability of triggering on either the rising edge or the falling edge of the input. When triggering on the rising edge, the trigger fires at the instant the waveform first exceeds the trigger threshold. When triggering on the falling edge, the trigger fires at the instant the waveform first falls below the trigger threshold. Using the appropriate menu button, set your scope to trigger on the "Falling" edge. You should observe that now the $t = 0$ occurs at the nearly straight vertical line in our ramp function. You'll also notice that adjusting the trigger threshold now has no discernible effect on the phase of the waveform. Triggering on sharp edges eliminates the effect of time-slewing, a fact we will exploit to make accurate time measurements.

5.3 Transient Response of an RC Circuit

Set your Channel 1 of your function generator to produce Square wave output with a peak-to-peak voltage of 6 V and a frequency of 1 kHz. As shown in Fig. 5.3, use a BNC Tee adapter to split the output into two copies. Send one copy directly to the Channel 1 input of your scope. Send the other copy to a BNC-alligator-pair cable.

Install an oscilloscope probe at Channel 2 of your scope. Some probes in the lab have the ability to switch between attenuation factors near the probe tip. If you have such a probe, select the 10X setting. The remaining probes have a fixed attenuation of 10X.

So far, we haven't had to worry about proper grounding procedure, because this is automatically handled by the BNC cable. Your scope probe has two main parts, the larger probe tip, which slides to reveal a hook which can be attached to wires and components in your circuit, and a short lead ending with a black alligator clip called the "Grounding clip". Handheld devices like your DMM have no connection to earth ground. The voltage reference point at the Common terminal can be connected anywhere you would like in a circuit. Your scope is quite different! It plugs into a wall outlet for power and is referenced to earth ground. To provide a scope which is both safe and cost effective, most scopes are limited to making measurements which are referenced to ground when using ordinary probes. The ground clip of your scope can only be connected to earth ground. If you connect it anywhere else in your circuit, that part of your circuit will be short-circuited to ground.

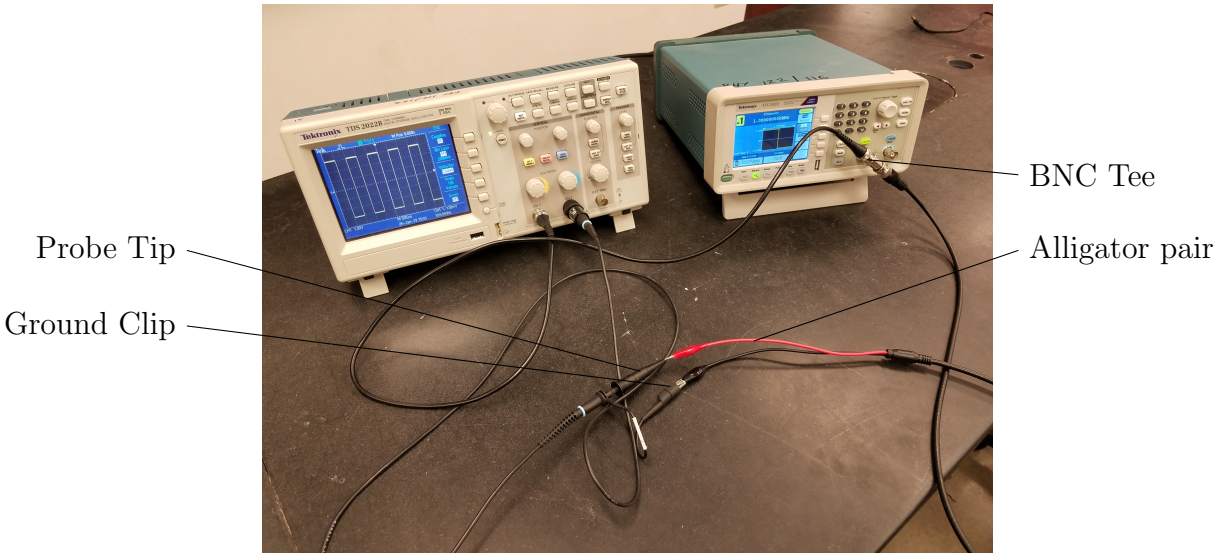


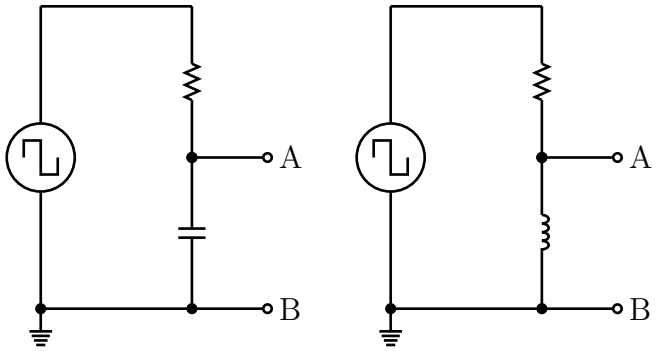
Figure 5.3: A scope trace triggered by the rising edge of ramp function. The point $t = 0$ is the location where the waveform first crosses the trigger thresholds, which is indicated by the arrow to the right.

For now, connect the probe directly to the output of the function generator. Your function generator is also referenced to earth ground. In particular for this setup, the black alligator clip is earth ground. Connect the black clip from the function generator to the grounding clip for the scope probe. Next connect the scope probe to the red alligator clip from the function generator. You now have two copies of the function generator output being sent to the scope. One directly through the BNC cable, and one through a 10X attenuation scope probe.

Adjust your scope to view the function generator output as measured by Trigger 1. Set the voltage scale as large as possible while observing the entire wave form. Leave the timescale at the default setting of 500 μs for now. Check that trigger is on the Falling edge, as set in the last section. Set the trigger near threshold near 0 volts. You should observe that the position of the waveform does not vary much with trigger threshold: the steep falling edge of the square wave function gives us a solid reference point for defining $t = 0$ in the measurements that follow.

Now enable Channel 2 on your scope. It should be an identical copy of Channel 1. Too see both channels at the same time, you'll have to move the vertical position of Channel 1 slightly. Closed loop tests like this are the way experienced scientists and engineers always start. It allows you to setup your signal generator and scope properly, without adding the complexity of the circuit you are working on. In general, avoiding confusion by taking small incremental steps is the fastest, most reliable way to proceed in lab.

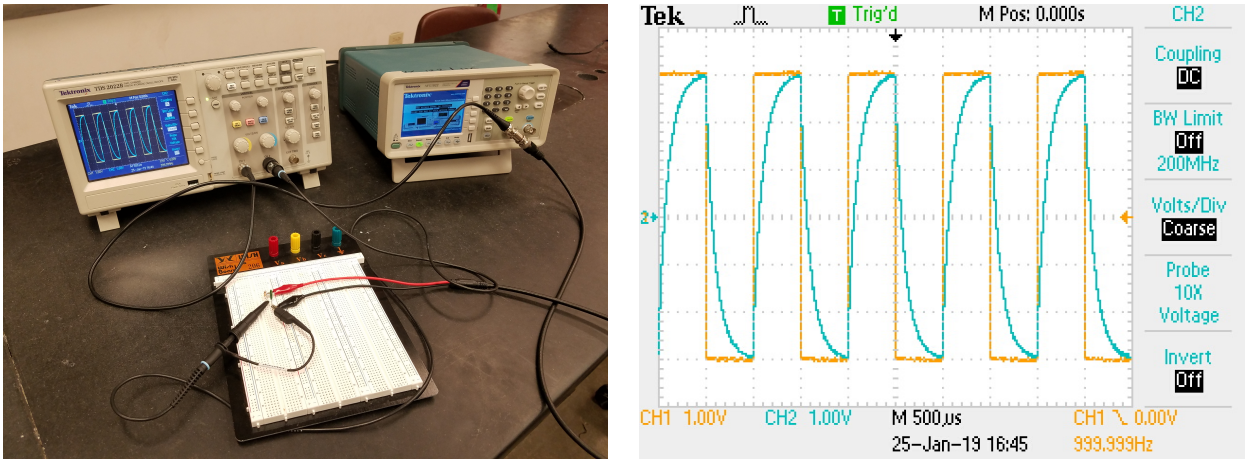
Construct the circuit in Fig. 5.4a on your breadboard, as shown in Fig. 5.5a. Use $C = 10$ nF capacitor and $R = 10$ kΩ. Using your DMM, measure and record the actual resistance and capacitance of your components before installing them. Use your function generator as the square wave voltage source. Recall that the black alligator clip is earth ground. The red alligator clip is the square wave output relative to earth ground, and corresponds to the upper terminal of the voltage source in the diagram. The only valid place to connect the grounding clip of your scope probe is to earth ground, so connect that to point B in your circuit. Connect the scope probe tip to point A. As shown in the Fig. 5.5b, each time the square wave changes polarity, the capacitor beings charging or discharging until it reaches equilibrium with the new voltage, revealing the characteristic



(a) (b)

Figure 5.4: A function generator driving an (a) RC circuit, and (b) RL circuit.

exponential transient response.

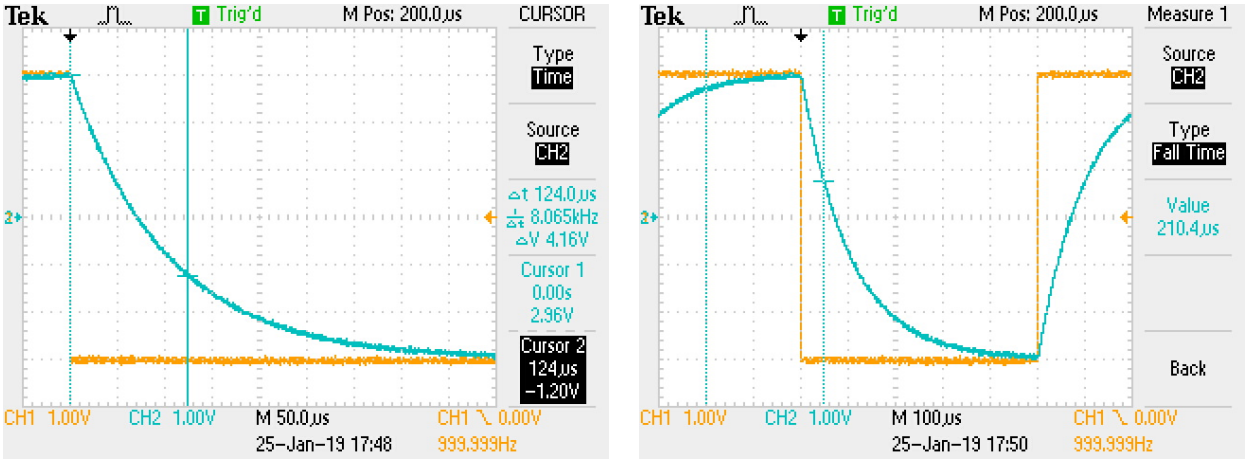


(a) (b)

Figure 5.5: Setup for the (a) RC circuit measurement, and (b) example scope trace showing exponential curve.

Now adjust the timescale to zoom in on the exponential decay portion of the curve, making sure to keep the trigger position at the left side of the display, as shown in Fig. 5.6a. Press the Cursor button, then set Type to Time, and Source to CH2. This feature allows you to make measurements of different points along the curve. Leave Cursor 1 located at $t = 0$. Highlight Cursor 2, by pressing the corresponding menu button, and adjust it's position using the multipurpose knob. Now you can make accurate measurements of the waveform by reading off the voltage and time at anywhere that you place the cursor. Record one measurement every ~ 25 μs starting from $t = 25$ μs to 350 μs. (Recall that when making measurements at target values, you need not hit the target value exactly, simply record the actual position at which you made your measurement.)

Your scope can also directly measure the fall time of a waveform. Setup the function so that one complete falling edge is on screen, as shown in Fig. 5.6b. Press Measure and then set source to CH2 and Type to "Fall Time". Check that the measured fall time is consistent with your measured values of R and C, using the formula from prelab.



(a) (b)

Figure 5.6: Scope traces showing (a) use of cursor to measure the waveform at $t = 124$ μs and $V = -1.20$ V, (b) use of the built in fall time measurement.

5.4 Transient response of an RL circuit

Wrap an inductor around the provided dowel, and estimate it's inductance by modifying your pre-lab calculation accordingly. Turn down the supply to 2.5 V peak-to-peak. Build the circuit in Fig. 5.4b using your homemade inductor and a resistor of $R = 47$ Ohms. Measure the fall-time of your R-L circuit and use it to determine a measured value for the inductance. Determine the inductance of your coil and compare to your theoretical estimate.

5.5 Analysis

Plot your collected RC circuit exponential decay data as discrete data points and compare with the expected exponential decay function as a continuous curve using the expected RC time constant calculated from your measured value of R and L. Make sure to have appropriate axis labels and a legend indicating "Data" and "Prediction".

Chapter 6

Passive Filters and Resonance

6.1 Pre-lab Calculations

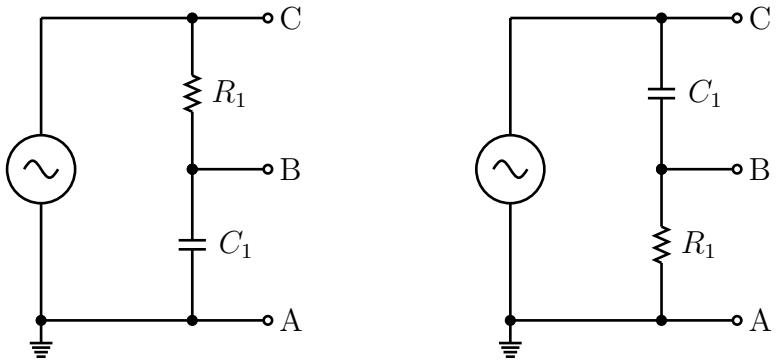


Figure 6.1: Circuit diagrams for RC (a) low-pass filter and (b) high-pass filter. The input voltage is $V_{\text{in}} = V_{\text{AC}}$ while the output voltage is $V_{\text{out}} = V_{\text{AB}}$.

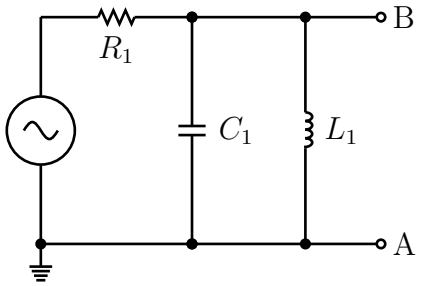


Figure 6.2: A function generator driving a resistor.

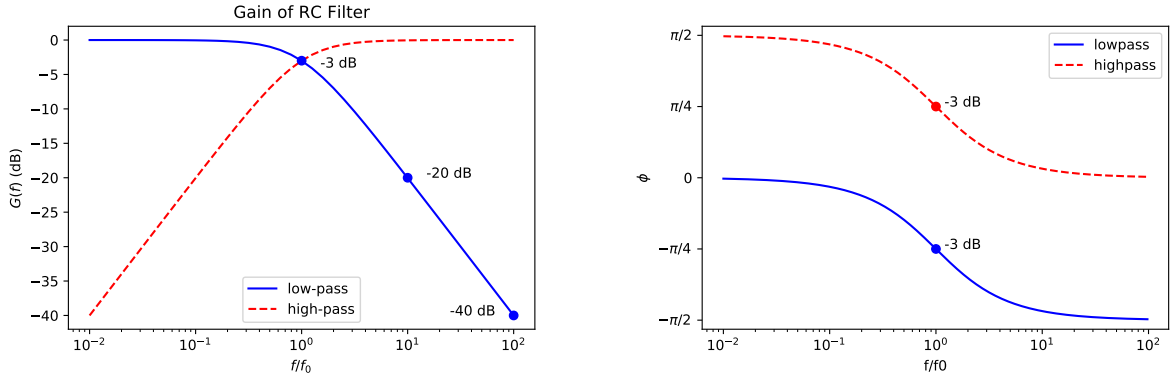
1) Calculate the crossover frequency f_0 (aka the frequency of -3 dB point) for the RC filters shown in Fig. 6.1 for $R_1 = 1.5$ kΩ and $C_1 = 10$ nF.

2) Calculate the resonant frequency:

$$f_0 = \frac{1}{2\pi\sqrt{LC}}. \tag{6.1}$$

of the resonant circuit in Fig. 6.2 for $R_1 = 1.5$ kΩ, $C_1 = 10$ nF, and $L_1 = 1$ mH.

6.2 Introduction



(a) (b)

Figure 6.3: Bode plots for high-pass and low-pass filters showing the (a) gain on a dB scale, and (b) phase, both as a function of the ratio of frequency f to the crossover frequency f_0 on a log scale.

In this lab, you will build and measure the performance of a low-pass and a high-pass RC filter, and produce Bode plots to compare your circuit with the impedance model derived in class and shown in Fig. 6.3. You will also build an RLC band-pass filter and determine its resonant frequency and quality (Q) factor.

6.3 Low-pass Filters

Set your function generator to produce a sine function with a peak-to-peak voltage of 4 V and set the frequency to 10 kHz. Build the circuit shown in Fig. 6.1a using $R = 1.5$ kΩ and $C = 10$ nF. Measure and record the actual resistance and capacitance of the components before installing them in your circuit. Use a BNC-alligator-pair cable to connect the function generator output to your proto-board, to provide the AC voltage source. Keep in mind that the black alligator clip from your function generator is earth ground, while the red alligator clip is the function generator output referenced to ground, i.e. the top of the AC source as drawn in the circuit diagram.

You will be using two scope probes in this lab, and as always some care must be taken with respect to grounding when using your scope. Your function generator has already set the earth ground point in your circuit at the black alligator clip, which is point A in the circuit diagram. The scope grounding clips should both be connected to point A. To measure V_{in} on your scope Channel 1, connect the scope probe tip of Channel 1 to point C in your circuit. To measure V_{out} on your scope Channel 2, connect the scope probe tip of Channel 2 to point B in your circuit.

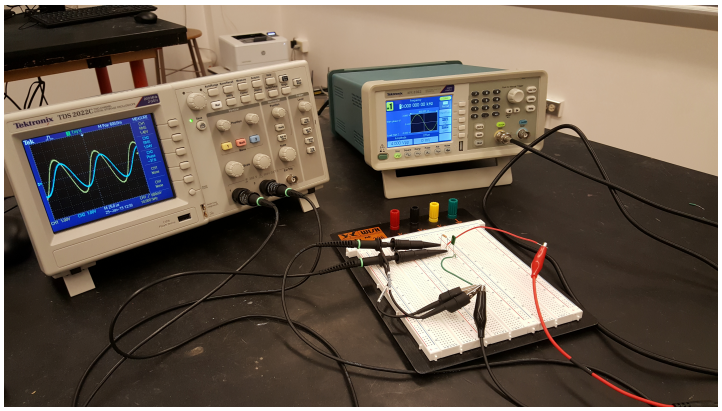
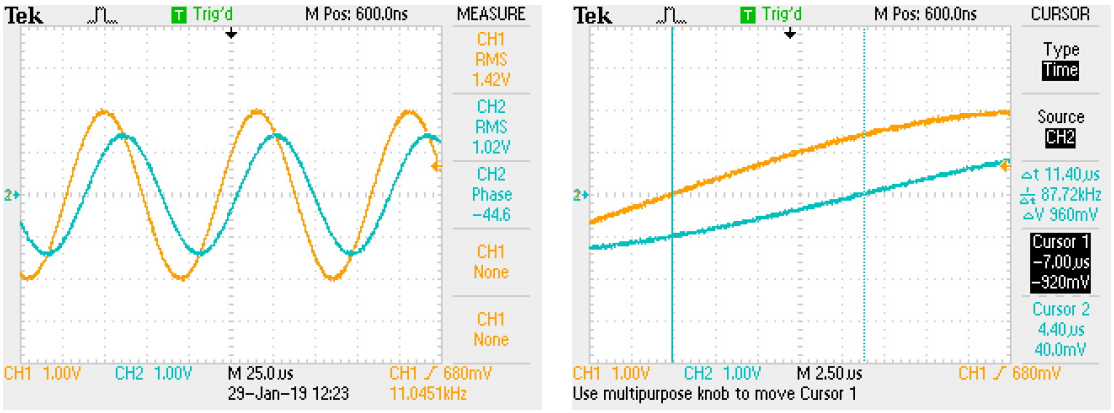


Figure 6.4: Setup for RC low-pass filter.

The setup, including the scope, is illustrated in Fig. 6.4. Note in particular that the two scope grounding clips and function generator ground are all connected to a single (green) wire, which is used to set the ground point in the circuit.

Set your Scope to the default setup, then adjust the timescale appropriately and enable the output of channel 2. Calculate at 1% precision the corner frequency f_0 from the measured values of your resistor and capacitor, and set the frequency of your function generator to that value. To produce the Bode plots for your filter, we will be measuring the voltage gain $G = V_{\text{out}}/V_{\text{in}}$ and the phase shift ϕ of $V_{\text{out}}(t)$ relative to $V_{\text{in}}(t)$.



(a) (b)

Figure 6.5: Measuring the gain and phase with your scope.

The gain is the ratio of the amplitudes which can be read from the scope traces. Using the example in Fig 6.5, the output has an amplitude of 1.4 divisions while the input has an amplitude of 2.0 divisions, so the gain is $G = 1.4/2.0 \sim 1/\sqrt{2}$. After adjusting the horizontal scale and using the cursors menu, as shown in Fig 6.5, the offset in time Δt between when each signals crosses zero is determined to be 11.4 μs. The offset in degrees is simply:

$$\phi = \cdot f_0\, \Delta t, \cdot (360^\circ).$$

In this case, $\phi \sim 45^\circ$. The phase can also be estimated by noting that the output crosses zero half-way between where the input crosses zero and its peak value, which is 1/8 of a period, or 45°.

A gain of $1/sqrt2$ and a phase-shift of magnitude 45° is expected at the cross-over frequency (or −3 dB point).

Once you have estimated these quantities from the wave forms, you can setup your scope to measure the appropriate quantities for you, using the Measure menu, as shown in Fig. 6.5. The RMS voltage measurement is generally the most reliable amplitude measurement, so measure the RMS of each channel plus the phase difference of channel 2 relative to channel 1.

Measure and record the RMS amplitudes of channel 1 and channel 2, and the phase difference, at nine different frequencies, chosen to cover four orders of magnitude:

$$f = \{f_0/100, f_0/30, f_0/10, f_0/3, f_0, 3f_0, 10f_0, 30f_0, 100f_0\}$$

where f_0 is the cross-over frequency. You'll have to change the horizontal scale appropriately as you change the frequency, as well as the voltage scale for Channel 2 when the output is attenuated. When using the scopes automatic measurement functions, you should always check at a few points by estimating the quantities yourself as shown above. You should note these cross-checks in your logbook.

6.4 High-pass Filter

Using the same components as in the previous section, build the high-pass filter of Fig. **??**b, and repeat your measurements at the nine different frequencies:

$$f = \{f_0/100, f_0/30, f_0/10, f_0/3, f_0, 3f_0, 10f_0, 30f_0, 100f_0\}$$

6.5 Band-pass Filter

In lecture, we showed that the resonant angular frequency of an RLC band-pass filter is given by

$$\omega_0 = \frac{1}{\sqrt{LC}}. \tag{6.2}$$

At this frequency, the RLC resonant circuit has unit gain and no phase shift. As the frequency moves away from the resonant frequency, the gain drops. We define two points ω_+ and ω_- as the two frequencies, one above and one below ω_0, at which the gain drops below $1/\sqrt{2}$. These are the two −3 dB points which define the bandwidth of the system. We define the quality of the resonance by the ratio of resonant frequency to this bandwidth:

$$Q = \frac{\omega_0}{\omega_+ - \omega_-} = \frac{f_0}{f_+ - f_-}$$

Build the circuit in Fig. 6.2 using $R = 1.5$ kΩ , $C = 10$ nF, and $L = 1$ mH. Use an RLC meter to measure the actual inductance of the inductor. Calculate f_0 using the measured values of your components. Set the frequency of the function generator to f_0 and the peak-to-peak voltage to 4 V.

We'll measure the actual resonant frequency using a trick. During normal use, your oscilloscope displays the voltage of each channel versus time. But there is an XY mode available under the display options. In this mode, the scope displays the voltage of channel one versus the voltage of channel two. When in XY mode, two out-of-phase signals yield an ellipse. But, as shown in Fig. 6.6 when both channels are perfectly in phase, the ellipse collapses to make a diagonal line. You can set

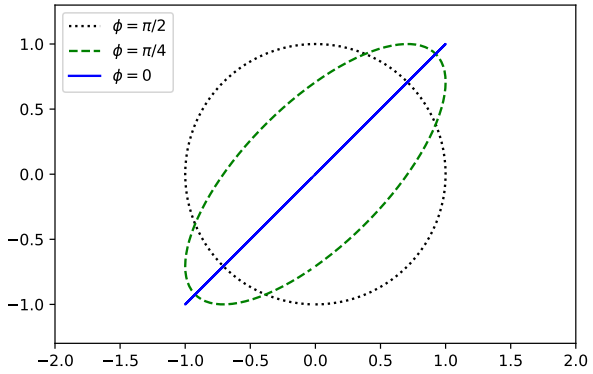


Figure 6.6: Expected scope traces in XY display mode for a relative phase $\phi = \pi/2$, $\phi = \pi/4$, and $\phi = 0$. It is easy, accurate, and somehow deeply satisfying to tune the frequency until the ellipse collapses into itself, forming a line.

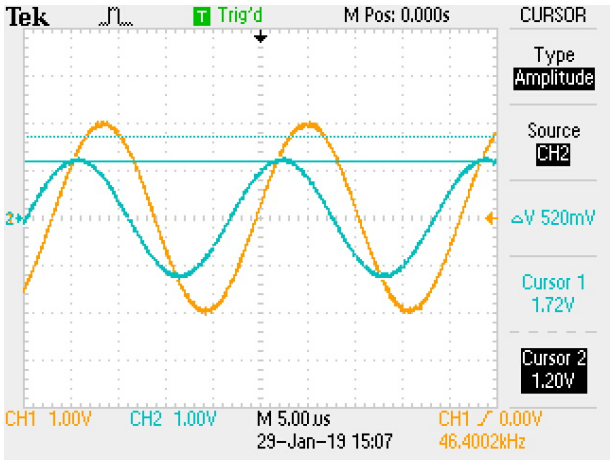


Figure 6.7: Set Cursor 1 to the peak of Signal 2 at the resonant frequency, then set Cursor 2 to this value times $1/sqrt(2)$. The frequencies f_+ and f_- can be determined by adjusting the frequency until the output amplitude reaches the level of Cursor 2, as shown here for f_-.

you scope in XY mode and adjust the frequency until the ellipse collapses to quickly and accurately find the resonance frequency. This is a **sign-off point** for this lab.

One you determine the resonant frequency, switch back to the normal display mode and determine the bandwidth. The easiest way to obtain this is to use the cursors to measure the peak voltage of your output signal. Multiply this by a factor of $1/\sqrt{2} = 0.7$ to determine the -3 dB amplitude and set the second cursor at this value, as shown in Fig. 6.7. Now adjust the frequency above and below the resonant frequency and note at which frequencies the amplitude reaches this line.

From your measurement of f_0, f_+, and f_- calculate the measured Q-factor of your circuit.

6.6 Analysis

Plot the measured gain as a function of frequency for your high-pass and low-pass filter, and compare to the expected response. Plot the measured phase shift as a function of frequency for your high-pass and low-pass filter, and compare to the expected response.

Chapter 7

Probability Distributions

7.1 Introduction

In this lab, you will create your own numerical simulation of a binomial process, and compare the results of your simulation with the PDFs for the binomial, Poisson, and Gaussian processes in the appropriate limits.

This lab includes a number of code snippets to illustrate the ideas that are being discussed. However, the entire source listing is not available to you, as this would amount to giving away the answer, and we all learn best by doing things ourselves! The examples should help you understand what you should do, but you will have to write your own code. **Do not expect the code snippets as written to simply work for your code without any modification... they are not intended to!**

7.2 Simulating the Binomial Process

Your first task is to create a Monte Carlo simulation for a binomial process. The Monte Carlo method, named after the casino in Monaco, refers to the repeated sampling of random variables to obtain numerical results.

Suppose one single experiment consists of n_{try} trials with a probability ϵ of success. The outcome of each experiment is a single number from 0 to n_{try} reporting the number of the n_{try} trials from that particular experiment which were successful. To study the distribution of outcomes, you will repeat the experiment n_{exp} times.

There are library functions that will simulate this process for you, but for this lab you will create your own simulation. While developing your code, start with a small test. For instance $n_{\text{try}} = 3$ and $n_{\text{exp}} = 5$, as shown in the example below. Implement your Monte Carlo simulation in the following manner:

- Create a 2-D array of shape n_{exp} by n_{try} filled with random values chosen uniformly from 0 to 1.0:

```
nexp = 5
ntry = 3
x = np.random.uniform(size=(nexp,ntry))
print(x)
```

```
[[0.90348221 0.39308051 0.62396996]
 [0.6378774  0.88049907 0.29917202]
 [0.70219827 0.90320616 0.88138193]
 [0.4057498  0.45244662 0.26707032]
 [0.16286487 0.8892147  0.14847623]]
```

This array associates a random value with each trial from each experiment.

- Consider a trial successful if the randomly chosen value is less than ϵ. In this example, taking $\epsilon = 0.5$ and assuming 1 indicates success and 0 a failure, we obtain:

```
[[0 1 0]
 [0 0 1]
 [0 0 0]
 [1 1 1]
 [1 0 1]]
```

  In the first simulated experiment, only the second trial was successful. In the second experiment, only the last trial successful. And so on.

- Next, count the number of trials that were successful in each experiment. The result will be a 1-D array of length n_{exp}. Consider using the `np.sum` function with the `axis` parameter. In this example, we obtain the array of outcomes m:

```
[1 1 0 3 2]
```

  This is the outcome of our five simulated experiments. The first and second experiment have one out of three trials successful, the third experiment had zero out of three trials successful, and so on.

Work through the example and make sure you see how each result follows from the previous step. Then implement and test your own version of this algorithm in Scientific Python. If you are an experienced programmer, you should put your simulation into a function like:

```
def throw_binomial(nexp,ntry,eps):
    x = np.random.uniform(size=(nexp,ntry))
    #...your simulation follows...
```

This will make your code much easier to manage, because you can simply call this function each time you need to run your simulation, but it is optional. Cutting and pasting is also permissible.

When validating a numerical calculation, think hard about good test cases. For instance, if you only test with the value of $\epsilon = 0.5$, you won't catch a bug that mistakes success for failure. Try a few different test cases, with reasonably small values for n_{try} and n_{exp} and check your numerical simulation. Boundaries often make a good check, for instance $\epsilon = 0$ and $\epsilon = 1$.

Another effective validation strategy is to check known mathematical relations using your simulation. For instance, we know that the mean value of a Binomial distribution is given by:

$$\bar{m} = n_{\text{try}} \cdot \epsilon \tag{7.1}$$

and that the variance is given by:

$$\sigma^2 = n_{\text{try}} \cdot \epsilon\,(1-\epsilon) \tag{7.2}$$

and so these predicted values, calculated from your parameters ϵ and n_{try} can be compared to the mean and variance of the outcome array m from your simulation, calculated using the numpy functions `np.mean` and `np.var`. A comparison with $n_{\text{exp}} = 1000$, $n_{\text{try}} = 10$, and $\epsilon = 0.5$ should result in something like:

```
print("mean expected:    ", ntry * eps)
print("mean simulated:   ", np.mean(m))
print("var expected:     ", ntry * eps * (1.0 - eps))
print("var simulated:    ", np.var(m))
```

```
mean expected:     5.0
mean simulated:    5.015
var expected:      2.5
var simulated:     2.3847750000000003
```

Produce a large number of pseudo-experiments by setting $n_{\text{exp}} = 1000$ and comment out any debugging print statements which will get very long. Pick five well chosen test cases with different values of n_{try} and ϵ and compare the expected and simulated mean and variances. The simulated values will fluctuate by a fractional amount $1/\sqrt{n_{\text{exp}}}$. So for $n_{\text{exp}} = 1000$, we expect the expected and simulated values to agree within about 3%. If you increase to $n_{\text{exp}} = 100000$, the agreement should improve to better than 1%. Be certain to record your test cases and the results in your logbook.

7.3 Histogram for the Binomial Process

We use histograms to represent the distribution of numerical data. In this case, our histogram simply reports the number of times each particular outcome occurs. Fill an output array m with the simulated results of $n_{\text{exp}} = 1000$ experiments each consisting of $n_{\text{try}} = 10$ trials with success rate $\epsilon = 0.25$. There are eleven possible outcomes for each of these experiments: $0, 1, 2, \ldots, 10$. We want to know how often each of these outcomes occurred.

Construct a histogram from your simulated results in array m using the `np.histogram` function:

```
counts,edges = np.histogram(m,bins=11,range=(0,11))
print("counts:     ", counts)
print("total:      ", np.sum(counts))
print("bin edges:  ", edges)
```

```
counts:      [ 65 172 284 269 122  64  19   4   1   0   0]
total:       1000
bin edges:   [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11.]
```

This calculates a histogram with eleven bins covering the range from zero to eleven, and reports the number of outcomes from m that occur in each bin. It returns two arrays, which we save as `counts` and `edges`. We interpret the `counts` array as follows: the first entry tells us that 65 experiments

had zero successes, the second entry that 172 experiments had one success, the third entry that 284 experiments had two successes,and so on. The exact values will vary each time you run the simulation, but in all cases the sum across all bins will equal the total number of experiments $n_{\text{exp}} = 1000$.

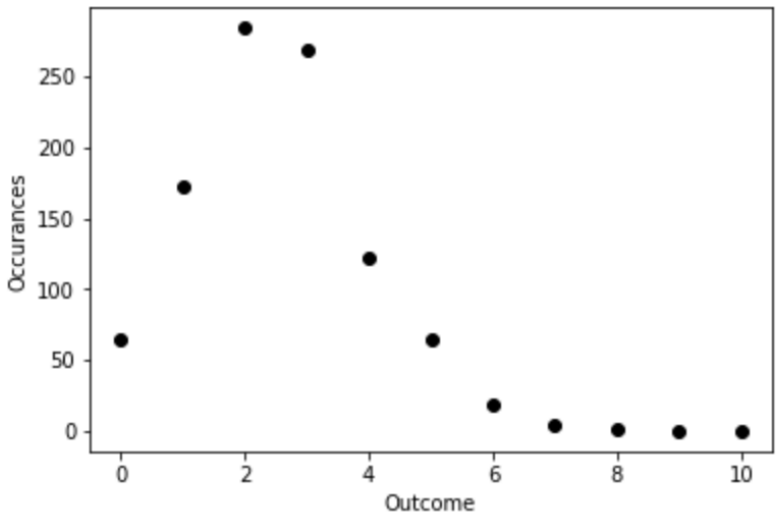
Histograms are most often used to handle continuous data, so you have to take a bit more care when using them to display discrete data (integers) as we are doing here. Consider the bin edges array `edges` that was returned in this example:

```
bin edges:   [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11.]
```

which you will notice has length 12. This is because the array contains the **edges** of eleven consecutive bins. Technically the first bin is the count of all outcomes in the range from zero (inclusive) to just below one (exclusive). The next bin is the count of all outcomes in the range from one (inclusive) to just below two (exclusive). In this case, however, we are using discrete data, and so there are no entries with values like 1.73 to consider. All the entries in the first bin are from experiments with the outcome zero, while all the entries in the second bin are from the outcome one. The best way to plot this histogram, therefore, is to associate each count with the leading bin edge, that is:

```
plt.plot(edges[:-1],counts,"ko")
plt.xlabel("Outcome")
plt.ylabel("Occurances")
print("edges[:-1]:  ", edges[:-1])
```

```
edges[:-1]:   [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```



Notice the essential trick for plotting histogram with discrete data in integer bins is to use the slice `edges[:-1]` of the bin edges data

```
edges[:-1]:   [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```

as the x values for plotting the occurances of each outcome. This slice (all but the last entry) essentially throws out the superfluous bin edge 11. As we will see later, this trick only works for discrete data in integer bins. Notice that in resulting plot, we have the number of occurances at each of the eleven possible outcomes correctly centered over the numbers $0, 1, 2, \ldots, 10$.

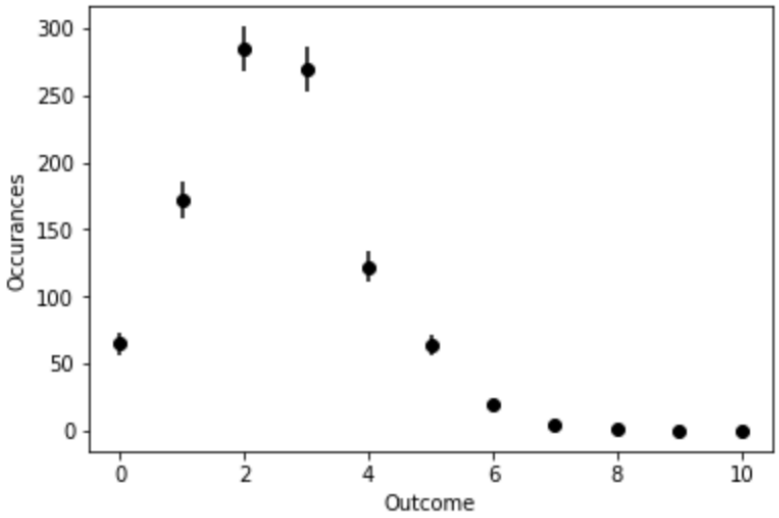
7.4 Error Bars

Run your simulation a few times and you will observe that the simulated values fluctuate slightly each time you run your code. But how much should we expect these values to fluctuate? If you consider a single bin of your histogram in isolation, it contains a single count N, which is itself the result of a simple counting experiment. Counting experiments are described by the Poisson distribution. Our best estimate for the mean value λ of this counting experiment is simply our count N. For the Poisson distribution the variance $\sigma^2 = \lambda$, and so we expect $\sigma = \sqrt{\lambda} = \sqrt{N}$. We expect each bin in our histograms to fluctuate by an amount σ which is the square root of the value of the bin.

To aid in interpreting histograms, it is useful to indicate the amount we expect each bin to fluctuate by adding to the data point an "error bar" with a length equal to the square root of the size of the number of events in the bin:

```
errs = counts**0.5
plt.errorbar(edges[:-1],counts,yerr=errs,fmt="ko")
plt.xlabel("Outcome")
plt.ylabel("Occurances")
```

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



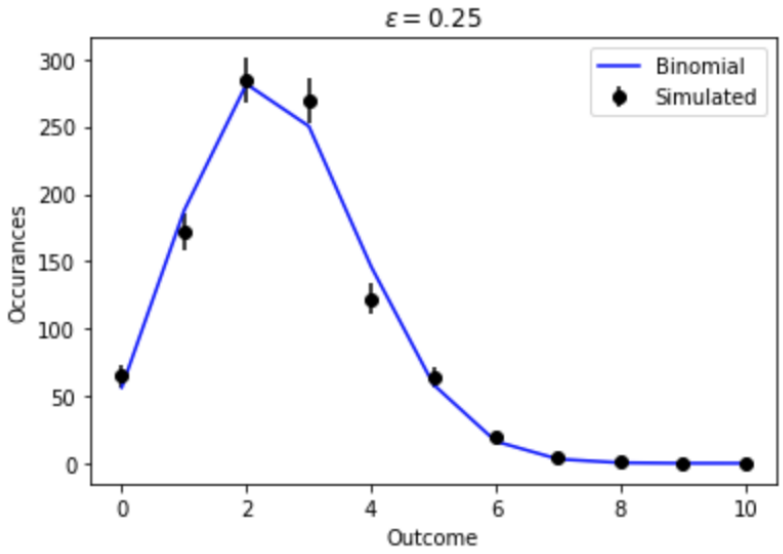
This provides an intuitive visual indication for the size of the statistical fluctuations associated with the counts reported by the histogram. Notice that the poorly named `errorbar` function plots *both* the data point and the error part, so it is a replacement for the `plot` function, not something you must call in addition.

7.5 Comparison with Binomial PDF

Next we'd like to compare our Monte Carlo simulation with the PDF for the binomial distribution. We'll use the `scipy.stats.binom.pmf` function, which is the PDF for the discrete binomial distribution available in Scientific Python. This PDF is only non-zero at integer values, so we'll simply evaluate it at each discrete occurance, i.e. the same slice `edges[0:-1]` which we used to plot the histogram:

```
from scipy.stats import binom
errs = counts**0.5
plt.errorbar(edges[:-1],counts,yerr=errs,fmt="ko",
             label="Simulated")
plt.xlabel("Outcome")
plt.ylabel("Occurances")
xpred = edges[:-1]
ypred = nexp * binom.pmf(xpred, ntry,eps)
plt.plot(xpred, ypred,"b-",label="Binomial")
plt.legend()
plt.title("$\epsilon=0.25$")
```

```
Text(0.5,1,'$\\epsilon=0.25$')
```



Notice that we scale the PDF by the number of experiments n_{exp}. The PDF is the expected frequency of each outcome for a single experiment, but we are plotting the number of occurrences for n_{exp} experiments.

Plots 1-3: Compare the output of your Monte Carlo simulation with the Binomial distribution PDF with $n_{\text{exp}} = 1000$ and $n_{\text{try}} = 10$, and for three different values of ϵ: 0.25, 0.5, and 0.75. For example, Plot 1 should look like the plot above.

7.6 The Poisson Limit

The Poisson distribution follows from the Binomial distribution in the limit that $n_{\text{try}} \to \infty$. Recall that the mean value of the Poisson distribution is $\bar{m} = \epsilon\, n_{\text{try}}$. If we kept the success rate ϵ as a parameter, than any finite value of ϵ would cause the mean of the distribution to diverge to infinity. If instead we hold the new parameter λ constant, and set:

$$\epsilon = \frac{\lambda}{n_{\text{try}}}$$

we see that $\epsilon \to 0$ as $n_{\text{try}} \to \infty$ and the mean of the Poisson distribution remains at the fixed value λ.

We'll explore this limit numerically simply by taking $n_{\rm try}$ to the large (but finite) value of 1000. Re-run your Monte Carlo simulation using the parameters $n_{\rm try} = 1000$, $n_{\rm exp} = 1000$, and $\epsilon = \lambda/n_{\rm try}$. For now, take $\lambda = 2.0$. Instead of the Binomial distribution, compare your simulation to the Poisson distribution PDF using the `poisson.pmf` function:

```
from scipy.stats import poisson
xpred = edges[:-1]
ypred = nexp * poisson.pmf(xpred, lamb)
```

Note that the first argument of the `pmf` function is the array of positions to evaluate the function at, while the second is the Poisson parameter λ. Also note that sadly `lambda` is a reserved word in python, and so you cannot use it as a variable name.

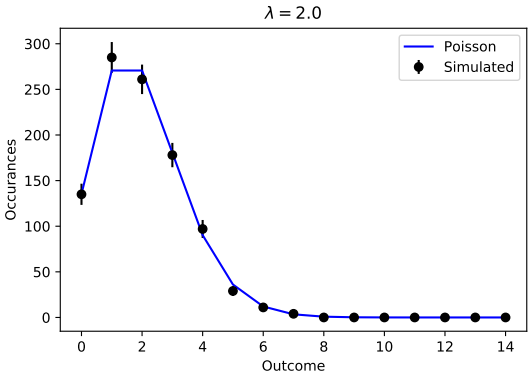


Figure 7.1: The expected result for **Plot 4**.

Plots 4-5: In the Poisson limit, compare the output of your Monte Carlo simulation to the Poisson PDF for two different values of λ: 2.0, 5.2. Plot the histogram with 15 bins for the outcomes: 0,1,2,3,...,14. An example is shown in Fig. 7.1.

This is a `sign-off` point for this lab. Make certain you can explain your code, and that it is as neat and organized as you can manage.

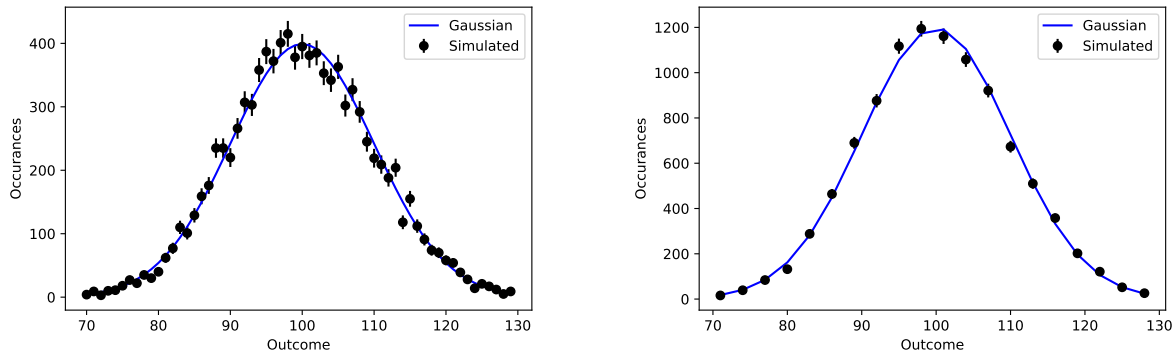
7.7 The Gaussian Limit

As the mean value λ of the Poisson distribution gets larger, the Poisson distribution resembles the Gaussian distribution. We will simulate this numerically by taking our Monte Carlo simulation in the Poisson limit, as above, with $\lambda = 100$. Initially use integer bins inclusively in the range 70 to 130, e.g.:

```
counts,edges = np.histogram(m,bins=60,range=(70,130))
```

and compare with the Gaussian distribution PDF (also called normal distribution), e.g.: `scipy.stats.norm.pd` function:

```
from scipy.stats import norm
xpred = edges[:-1]
ypred = nexp * norm.pdf(xpred, loc=lamb, scale=lamb**0.5).
```



(a) (b)

Figure 7.2: The expected result for **Plot 4**.

Set the parameter `loc` to the mean value, and the parameter `scale` to σ. Recall that in the Poisson limit $\sigma^2 = \lambda$, which is why we set `scale=lamb**0.5` in the example. This should reproduce the plot in Fig. reffig:gaussa.

You should see that 60 bins is rather unwieldy. We'll reduce the number of bins, but that's actually a bit more complicated than you might expect: non-integer bins with discrete data is about the most challenging binning you can tackle. You'll have to do the following:

- While keeping the range of 70 to 130, set the number bins to `bins=20` when filling your histogram.
- Our trick to use edges[:-1] will no longer work, since now the data for each bin is associated with a range of values in the bin. If we live this as is, the data will be plotted with an observable bias. For continuous data, we often simply use the middle of the bin:

  ```
  cbins = (edges[:-1] + edges[1:])/2
  ```

  but that is slightly biased in this case of discrete data which doesn't extend all the way to the right edge. To be precise, the x position we should use for plotting the contents of each bin is:

  ```
  cbins = (edges[:-1] + edges[1:] -1 )/2
  ```

- The normalization of the PDF to the data now requires an additional scale factor to account for the wider bins (which integrate more probability). You need to scale by an additional factor of 3 to account for the fact that each bin is now 3 integers wide.

Plots 6 Using the techniques described above, reproduce Fig 7.2b, which shows that the Binomial distributions becomes a Gaussian distribution as the mean value of the distribution becomes large.