Physics 80 Lab Manual

Michael Mulhearn and Emilija Pantic

February 11, 2020

Chapter 1

Introduction to Plotting

1.1 Introduction

This lab will introduce calculations and plotting techniques using numpy arrays within Scientific Python. See the course syllabus for details on producing and submitting your Jupyter Notebook. There are no logbook entries for this lab.

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.ylabel`.
- 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 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>
```

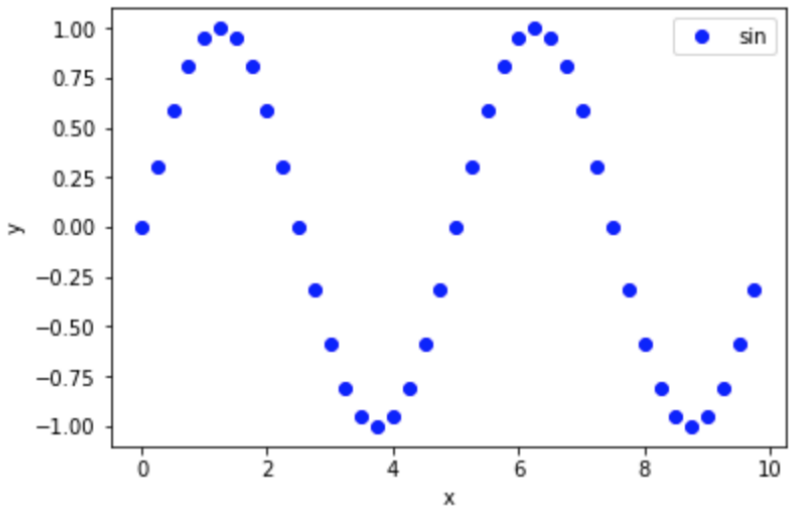


Figure 1.1: Sine function sampled at discrete values.

Jupyter Notebook 1.1.

Plot the sinc function with the following requirements:

- Plot in the range $x = (-5, 5)$.
- Plot discrete samples with a step size of 0.5 using blue circles.
- On the same axis, plot the corresponding smooth function using a red solid line. The smooth function should use much finer binning to approximate a continuous curve.
- Add appropriate axis labels.
- Add a legend for the "discrete" and the "smooth" function.

Use the numpy sinc function, e.g. `y = np.sinc(x)`.

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])
```

Jupyter Notebook 1.2.

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.**

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:

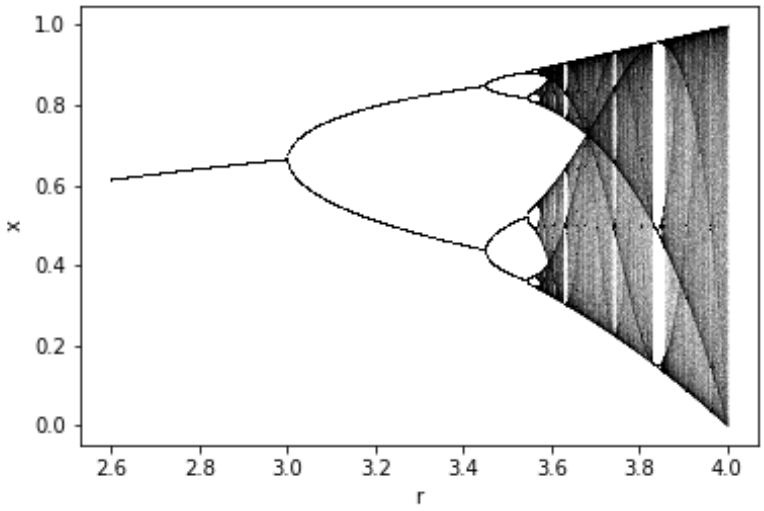


Figure 1.3: Long term behavior of the logistics map.

- An array of r values is defined.
- 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.

Jupyter Notebook 1.3.

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')
```

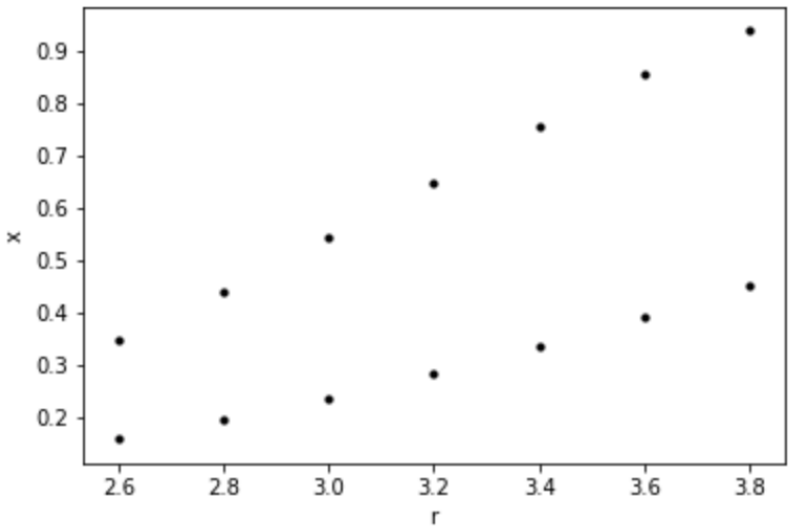


Figure 1.4: Modeling the logistics map.

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. This lab requires both logbook and Jupyter Notebook entries. Please read the syllabus for details on creating and submitting Jupyter notebooks, and maintaining your logbook.

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 further until the limit is increased. Turn the current limit knob just a bit past the point where 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. This procedure sets the current limit just past the current needed by your circuit. If you get in the habit of doing this every time you use your bench-top supply, you will save many components from 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.

The accuracy of your DMM for the DC Voltage reading is listed in Table 2.1. When making a measurement, you'll want to also record enough information to reproduce the accuracy later if needed. Recording the range scale you used, or recording the least significant digit (LSB), or calculating the accuracy using the table are all valid approaches. You'll also need to know which model of DMM.

Logbook Entry 2.1. With the alligator clip probes connected together, your DMM should read

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

about 0.00 V. Record your reading. 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 about 10 V. Record your reading. Next touch red probe to black terminal and black probe to red terminal, and record your reading. 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.

Instrument	**Range**	**Accuracy**
Triplet	200 mV- 200.0 V (DC)	$\pm(0.5\% \times \text{Reading} + 2 \text{ LSDs})$.
Mastech	200 mV- 200.0 V (DC)	$\pm(0.5\% \times \text{Reading} + 1 \text{ LSDs})$.
Triplet	200.0 Ω	$\pm(1\% \times (\text{Reading-Residual Resistance}) + 4 \text{ LSDs})$.
Triplet	2.000 kΩ	$\pm(1\% \times \text{Reading} + 2 \text{ LSDs})$
Triplet	20.00 kΩ -2.000 MΩ	$\pm(1.2\% \times \text{Reading} + 2 \text{ LSDs})$
Triplet	20.00 MΩ	$\pm(2\% \times \text{Reading} + 5 \text{ LSDs})$
Triplet	20.00 nF	$\pm(4\% \times \text{Reading} + 3 \text{ LSDs})$

Table 2.1: The accuracy of the Triplett 9007 and Mastech MS8264 multimeters. LSDs indicates the least significant digit for that range, as observed from the display, e.g. a readinog of 200.3 on the 200 V DC scale would have an uncertainty of $0.005 \times 200.3 + 2 * 0.1$ V

2.4 Resistance Measurement

Logbook Entry 2.2. Pick one resistor from the collection of resistors at the front of the lab. Connect the alligator clip across one of the resistors from the pile, and turn the dial to the resistance

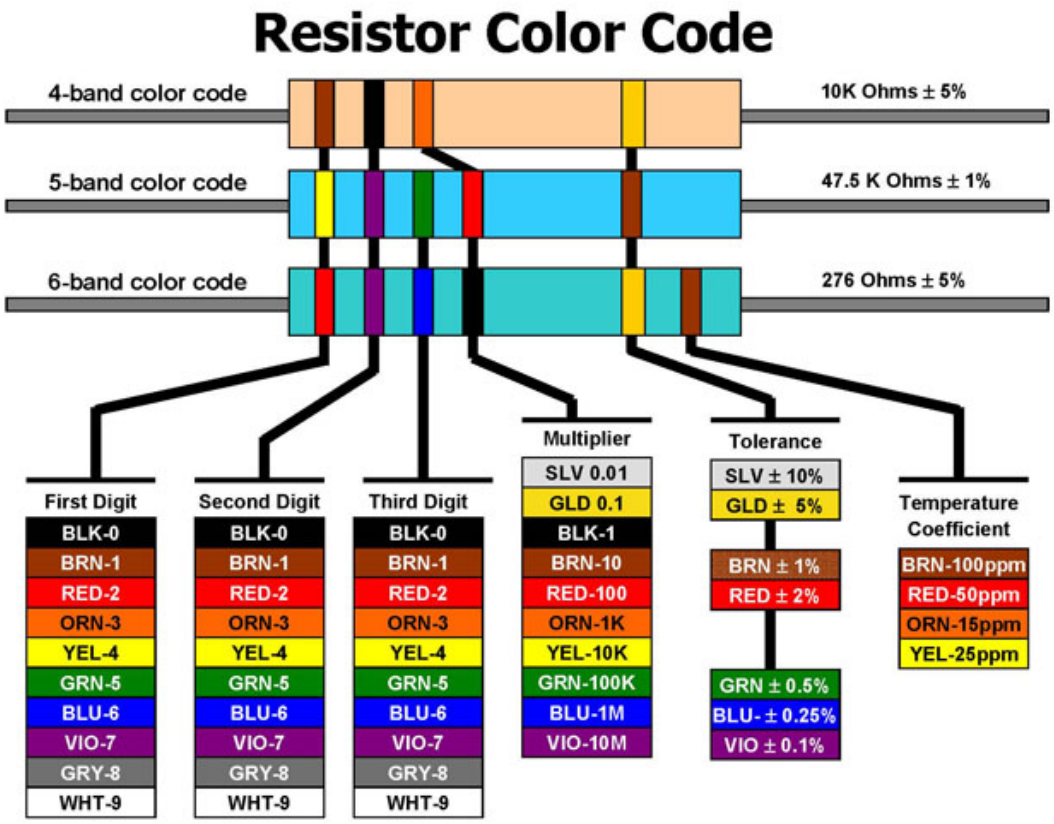


Figure 2.3: The resistor color code.

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. Use Table 2.1 to assess the accuracy of your measurement.

Resistors have a nominal value and a tolerance. The tolerance indicates the maximum percentage by which actual values of resistors provided by the manufacturer will vary from the nominal value. Using the resistor color coding guide in Fig. 2.3, determine the nominal resistance and tolerance of your selected resistor. In your logbook, record your resistor color pattern, the nominal value you determined, and the maximum and minimum value within the specified tolerance. Compare this to your measured value. Is your measurement consistent with the tolerance range for your resistor?

2.5 Current Measurement

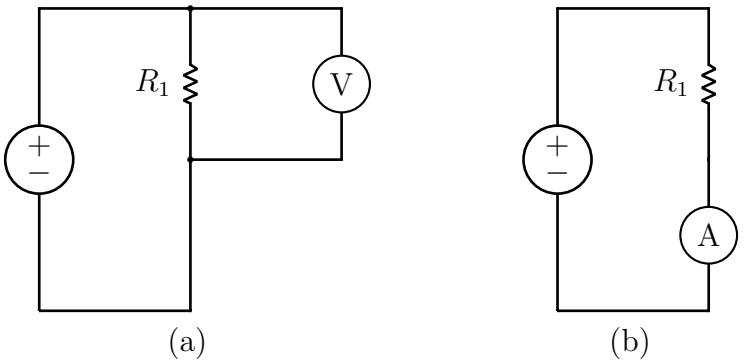


Figure 2.4: Appropriate connections for measuring (a) the voltage across resistor R_1 and (b) the current through the resistor R_1.

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 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 DMM which features a resettable fuse for making current measurements.

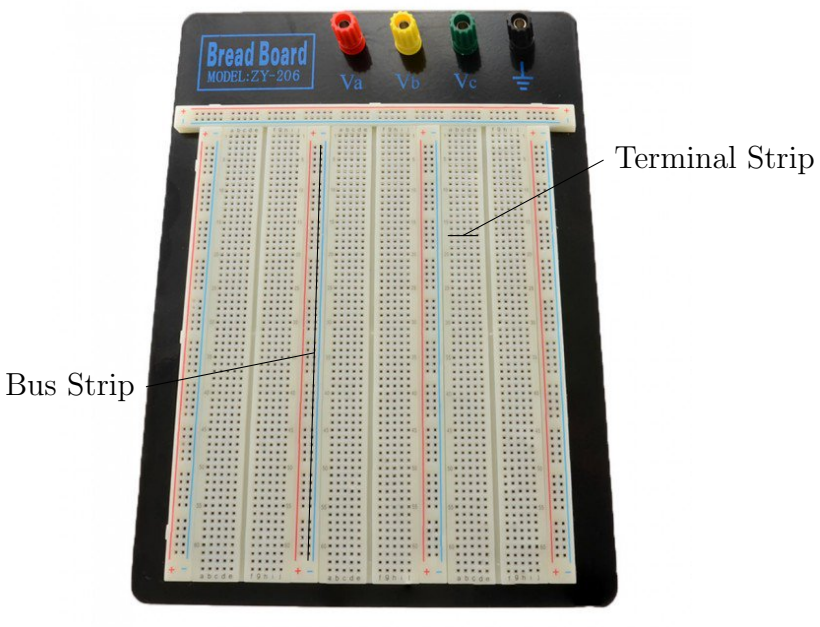


Figure 2.5: A typical breadboard.

2.6 Breadboard

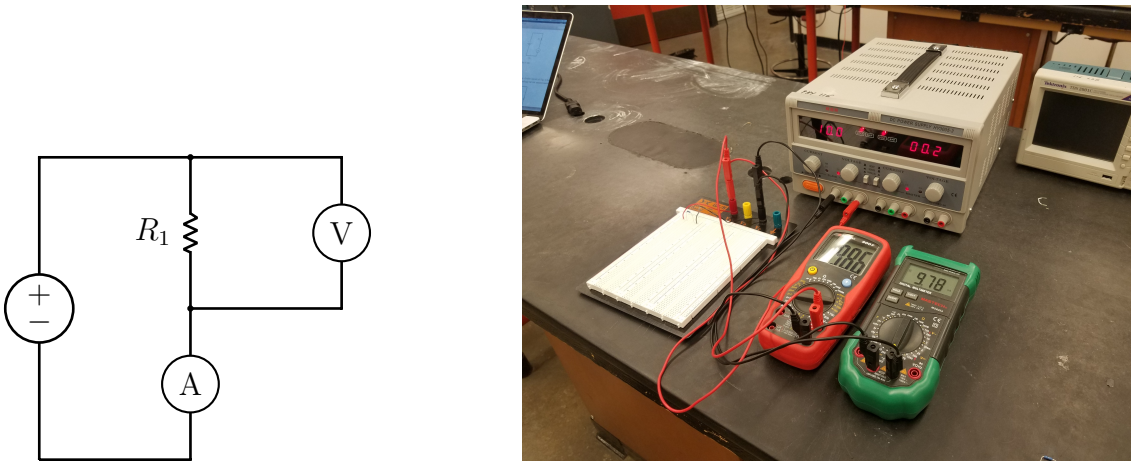
Breadboards are a convenient way to prototype circuits without having to solder. When the leads of discrete components are inserted in the holes in the breadboard, they make electrical contact with metal strips inside the breadboard that connect with additional holes. The short terminal strips are used 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. Take time to familiarize yourself with the breadboard.

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 (which you can take at the front desk). 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.

Logbook Entry 2.3. : 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. Record these values and the sketch of the circuit in the logbook.

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!



(a) (b)

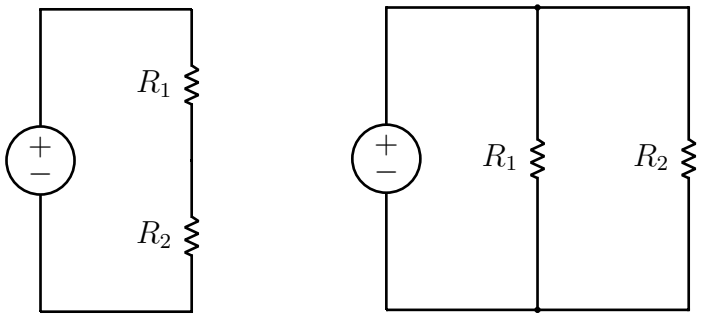
Figure 2.6: Circuit for verifying Ohm's law as a (a) circuit diagram, and (b) implemented using your lab equipment.

Jupyter Notebook 2.1. 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").

Logbook Entry 2.4. : After taking your last measurement, leave all the connections in place and the power-supply at 10 V. Record in your logbook the resistance of the resistor R_1 reported by your DMM. Is this a reasonable measurement? Record your comments in the logbook.

Logbook Entry 2.5. : Turn off the DC supply and record the resistance reported by the DMM. Is this accurate? Record your comments in the logbook.

Logbook Entry 2.6. : Remove the resistor from your circuit and measure the resistance with your DMM. Is this accurate? Record your comments in the logbook.



(a) (b)

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

2.8 Voltage Divider and resistors in parallel

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.

Logbook Entry 2.7. : Before installing it in your circuit, record the actual value of your resistor R_2 in your logbook. 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. Record a sketch of your circuit in the logbook. Compare these measured values to your expectation.

This is a **sign-off point** for this lab. (If the TA is not readily available, you can proceed and sign-off with the configuration for the next measurement instead.)

Logbook Entry 2.8. : Now adjust your circuit so that R_1 and R_2 are in parallel and set the supply to 10 V Record the voltage across the resistors R_1 and R_2 and the current through each resistor. Record the sketch of your circuit in the logbook. Compare the measured currents to your expectation.

Please return all the components you took and cables to their place. Leave you workstation clean.

Chapter 3

Equivalent Circuits

In this lab, you will explore Thevenin equivalent circuits. You will also solder two resistor circuits to explore the Δ-Y transformation for three terminal networks. For this lab, both logbook and Jupyter Notebook entries are required. You can find the resistors inside storage cabinet at the table to your left and cables hanging at the wall to your left. Please return them at the end of the lab.

3.1 Calculations

During lecture we determined 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.

Logbook Entry 3.1. Repeat this calculation in terms of V_1, V_2, R_1, R_2, R_3 in your logbook together with the sketch of the circuit.

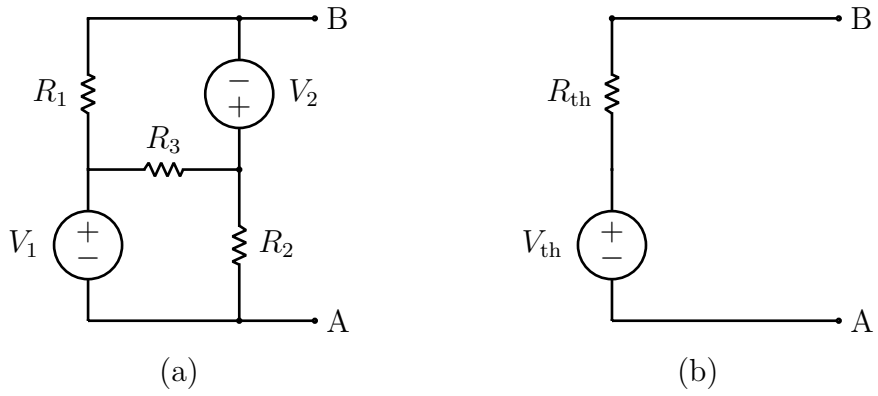


Figure 3.1: The circuit (a) you will be building in lab and it's (b) Thevenin Equilvalent.

3.2 Thevenin Equivalent Circuit

Build the circuit (on a breadboard) 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 (A) whereas the negative (black) output of V_2 is connected to point (B).Use your Triplett 9007 as a voltmeter and the Mastech MS8624 as a current meter.

Logbook Entry 3.2. 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. Record these values in your logbook.

Logbook Entry 3.3. First measure the open circuit voltage V_{ab}. Next short the points (a) and (b) through your current meter. Record these values in your logbook. 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. Record your comments in the logbook.

Logbook Entry 3.4. 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 $R = 470\ \Omega, 1.2\ \text{k}\Omega, 4.7\ \text{k}\Omega$. Record these values in your logbook.

3.3 Analysis

Jupyter Notebook 3.1. To present your analysis you should produce a plot similar to 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 should include data points for the five current and voltage measurements which you have made.

3.4 Impact of the Multimeter on the Measured Circuit

A perfect DMM would not affect the circuit you are measuring at all, but this is not the case in practice. An ideal voltmeter has infinite resistance so that it draws zero current when measuring voltage across two points. An ideal ammeter has zero resistance so that there is no voltage change as the current passes through it. Your DMM has a small non-zero resistance when used as an ammeter, and a large resistance when used as a voltmeter.

Logbook Entry 3.5. Build the voltage divider of Fig. 2.7a but with $V_1 = 10$ V and $R_1 = R_2 = 10\ \text{M}\Omega$. Record the voltage across one of the resistors. Compare the value your measure with what you predict for a perfect DMM. Sketch the circuit with a realistic DMM and calculate the resistance of your DMM when used as a voltmeter.

This is a **sign-off point** for this lab.

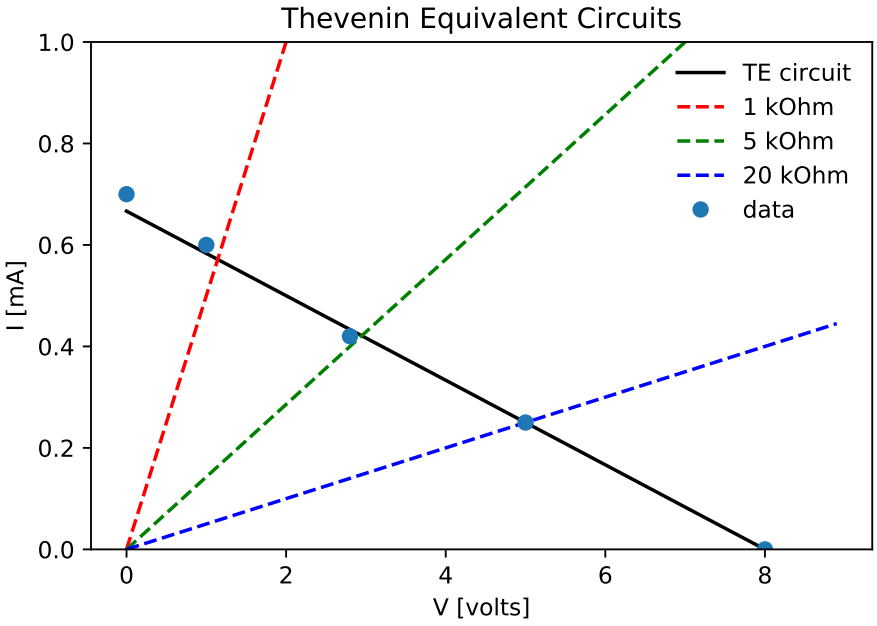


Figure 3.2: Example of IV curves for Thevenin equivalent source circuit with various load resistors.

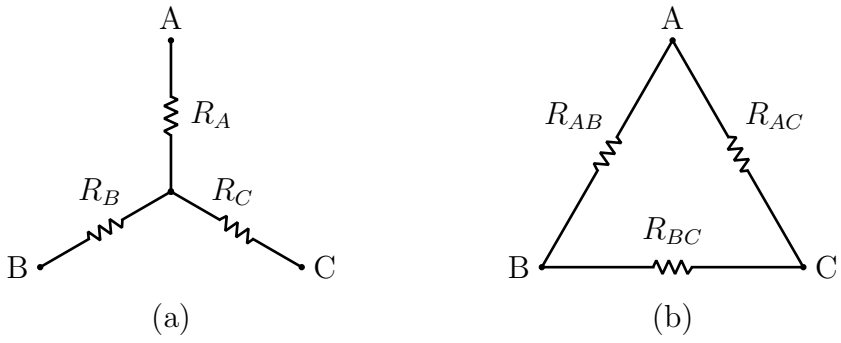


Figure 3.3: Equivalent three-node circuits.

3.5 $\Delta - Y$ Transformation

Not everyone will complete this portion of the lab. Do your best to finish as much as you can.

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

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

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. If 3 kΩ resistors are not available, 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. There is one soldering iron per two workstation. Find the squeeze bottle around the lab. The clamps (possibly various styles) are on the shelves to your left. 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.

Logbook Entry 3.6. Check the resistance between pairs of terminals on your creations, and compare with your expectation. Record those values in your logbook. You can bring your creations home if you like or bring them to the front desk.

This is an additional **sign-off point** for this lab.
Please return all the components you took and cables to their place. Leave you workstation clean.

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, how to trigger on and view time-dependent wave forms on your digital oscilloscope. You will produce Lissajous figures on your oscilloscope and reproduce them using Scientific Python. For this lab there are both logbook and Jupyter notebook entries.

4.2 Function Generator

This section introduces you to your Function Generator and illustrates some of the common used features and pitfalls. It should not take more than about one-half hour to complete.

Connect the output of Channel 1 directly to the Voltage measurement input of your Triplett 9007 DMM, using a coaxial cable with BNC connectors, and a BNC to banana plug adapter as shown in Fig. 4.1. BNC is a type of quick connector often used for coaxial cable. Coaxial cable is a type of electrical cable that can transmit high frequency signals with low losses. It has an inner conductor surrounded by an insulating layer, surrounded by a conducting shield. Coaxial cable has a characteristic impedance, which specifies the resistance that should be placed at the receiving end of the cable to minimize reflections, which can degrade high frequency signals. Your coaxial cable has a characteristic impedance of 50 Ω.

Turn on power to the function generator. Then set your function generator to the factory default:

Utility Button $\rightarrow$ System $\rightarrow$ Set to Default $\rightarrow$ 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.

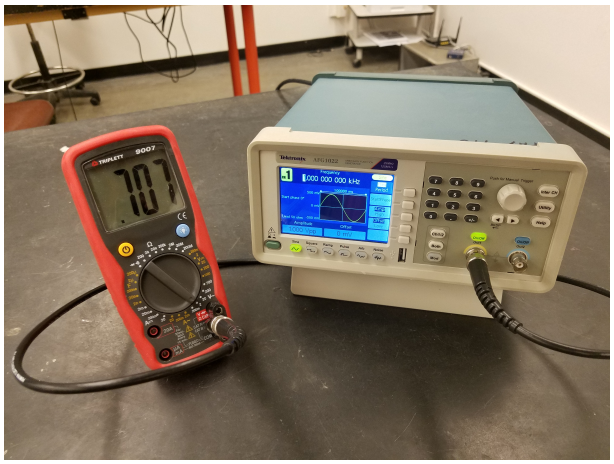


Figure 4.1: Connect the Channel 1 Output of your function generator directly to your DMM.

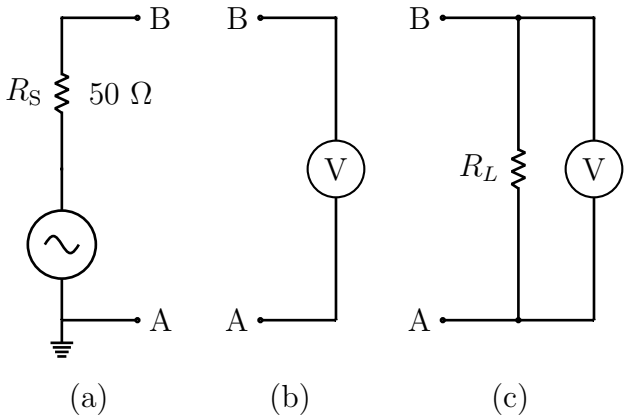


Figure 4.2: Equivalent circuit for (a) your function generator, which includes a 50 Ω source resistance, and two typical terminations for a coaxial signal: (b) infinite resistance voltmeter or scope, or (c) a terminating resistor in parallel.

Logbook Entry 4.1. Set your DMM to the 2 V (AC) scale (the V with a squiggly line). Measure the AC voltage, which should be close to:

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

Record the measured value in your logbook with a sketch of your circuit.

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

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

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

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

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 $R_{\text{L}} = 50\ \Omega$. In this case, the internal resistance and load resistance form a voltage divider, so that the output voltage V_{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.

Logbook Entry 4.2. To fix this discrepancy, we simply have to configure our generator to expect a high load resistance at both outputs:

$$\text{Utility Button} \rightarrow \text{OutputSetup} \rightarrow \text{CH1Load} \rightarrow \text{HighZ}$$
$$\text{CH2Load} \rightarrow \text{HighZ}$$

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:

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

Record the measured value in your logbook, and explain, in your own words, why this differs from the previous measurement.

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.

Logbook Entry 4.3. 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. Record this value and instrumental uncertainty in your logbook.

Logbook Entry 4.4. 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. Record your observation in your logbook. 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.

Logbook Entry 4.5. 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. Record the measured AC and DC voltages that you have measured and sketch the waveform ($V(t)$) for one period, which is 1 ms for this 1 kHz signal.

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.

4.3 Oscilloscope

This section introduces you to your Oscilloscope. It should not take more than about one-half hour to complete.

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.

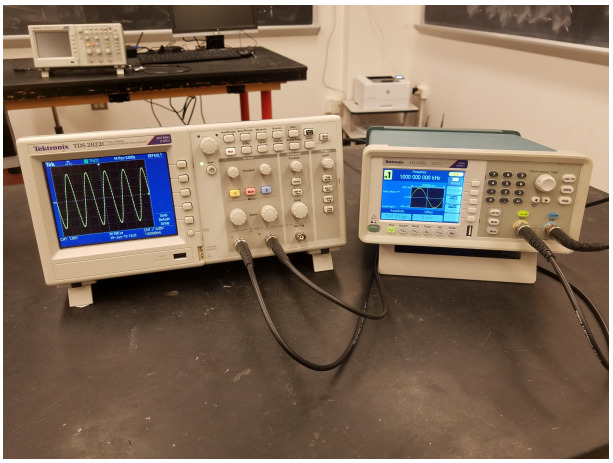


Figure 4.3: Connect the Channel 1 Output of your function generator directly to the Channel 1 input of your digital oscilloscope.

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!

Figure 4.4: An example scope probe.

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 affected 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 there 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

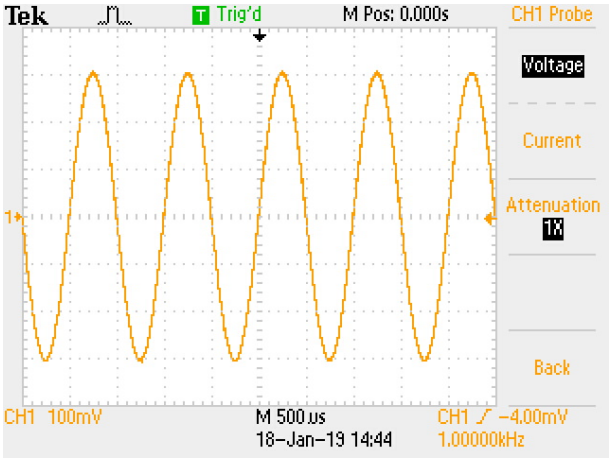


Figure 4.5: Correctly scaled scope output.

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. 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 scope display. By default, your scope includes the DC offset, but often

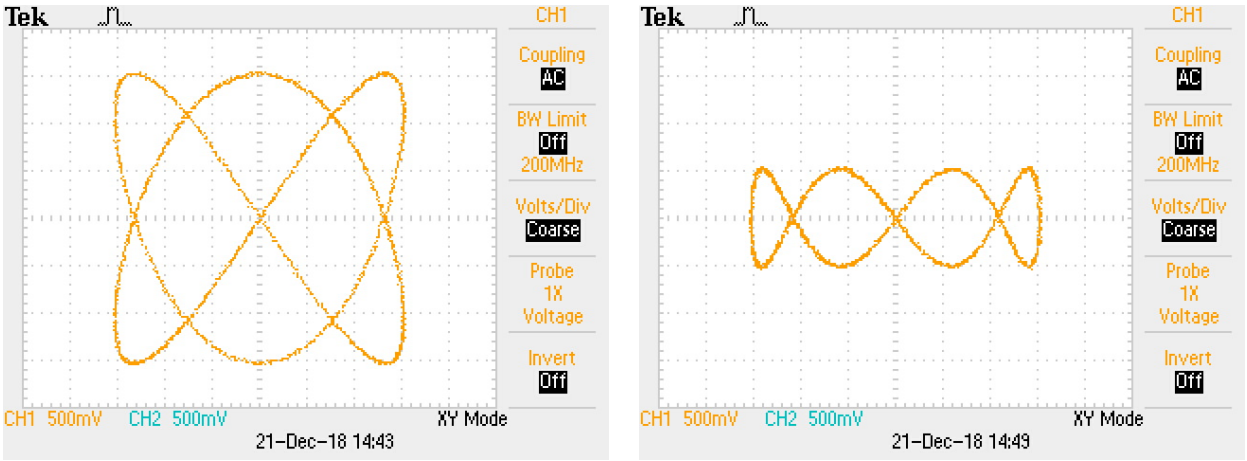
this is not what you want. On your scope, press the button labeled "Coupling DC", until the DC becomes AC. When AC coupled, the DC component of your waveform is removed. When AC coupled, observe that changing the DC offset on the function generator does not change the position of the waveform.

You can adjust the position of the waveform on your scope display 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.

Jupyter Notebook 4.1. Save a scope trace of a sine wave with a non-zero vertical and horizontal offset. Insert your USB drive into the scope and press the Save button (it takes few seconds to save). Whenever you save a scope trace in this class, make certain that there is a a date printed on the trace. This ensures that you have your own scope traces, as they can be confused with those from other sections.

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.

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

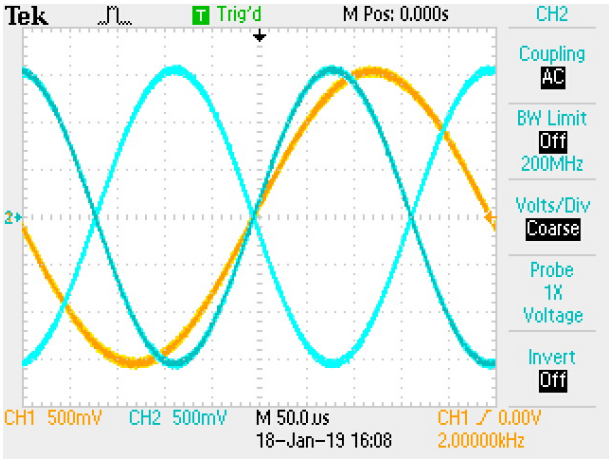


Figure 4.7: Correctly scaled scope output.

channel 2 to 3 kHz. Note that you can switch between the Channel 1 and Channel 2 parameter menus on the function generator with the button labeled "Ch1/2".

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 scope 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. You can check this but changing slightly the frequency of Channel 2 but return it to prescribe values to continue with the lab.

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

Jupyter Notebook 4.2. 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.

Jupyter Notebook 4.3. Produce the parabola pattern according to the settings in Table 4.1, saving a scope trace. Remember to align the phase each time you change the frequency.

Jupyter Notebook 4.4. Produce lace pattern and save a scope trace.

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

Jupyter Notebook 4.5. 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** for this lab.

Please return all the components you took and cables to their place. Leave you workstation clean.

4.5 Lissajous Figures Analysis

If you run out of time, you can finish the remaining portion of this lab, which is exclusively done in Jupyter notebook, at another time, such as during the Friday open lab period. Just make sure you have saved all of your oscilloscope traces.

Include your two scope traces in the python notebook. Make sure the date is clearly visible on each plot. Provide a full label for each plot which describes all the relevant information. You can display your scope traces in python using the Image library like this:

```
from IPython.display import Image

image = Image(filename='myscope.jpg')
display(image)
```

This assumes that you copied your scope traces inside the Jupyter notebook directory.

Jupyter Notebook 4.6. Reproduce fish pattern using scientific python to draw the parameterized shape. For example, see Fig. 4.8.

Jupyter Notebook 4.7. Reproduce parabola pattern.

Jupyter Notebook 4.8. Reproduce crown pattern.

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.

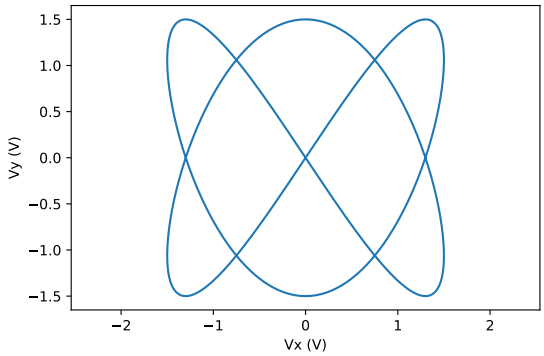


Figure 4.8: Lissajous curve constructed using Scientific Python corresponding to the scope trace in Fig. 4.6a.

Chapter 5

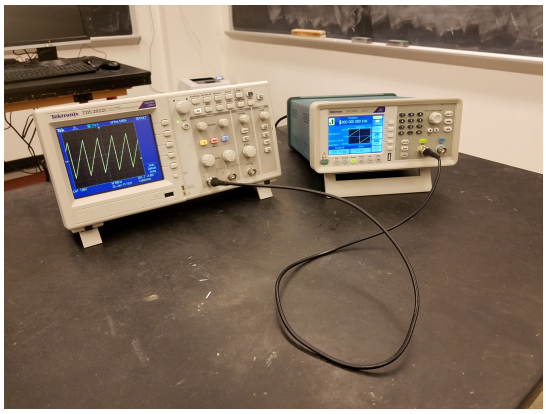
RC and RL Transient Signals

5.1 Introduction

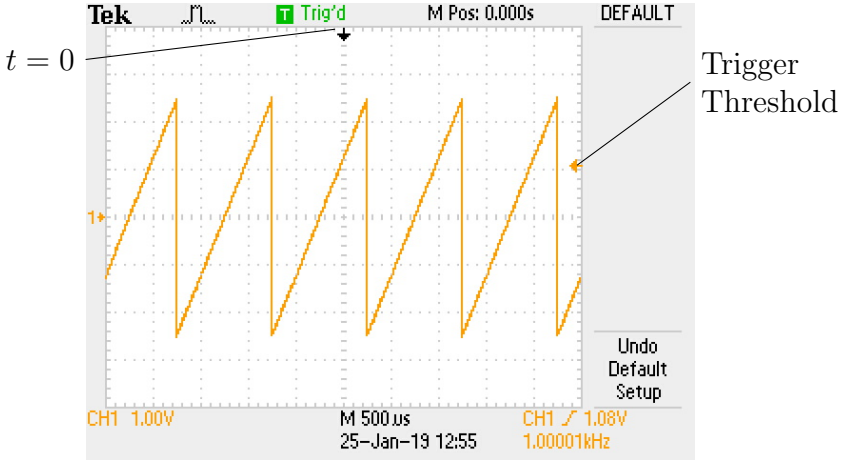
In this lab you will use explore transient behavior of *RC* and *RL* circuits. You will learn how to configure the trigger of your digital oscilloscope. You will learn how to use scope probes and measure properties of waveforms on your digital oscilloscope using cursors. For this lab there are both logbook and Jupyter notebook entries.

5.2 Oscilloscope Trigger

This section introduces the trigger feature of your digital oscilloscope. It should not take more than about one half hour.



(a)



(b)

Figure 5.1: (a) Setup for the scope trigger measurement, and (b) 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. 5.1a. 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%. Recall 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.)

Turn on your oscilloscope and press the "Default Setup" button. 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. Notice that the trigger value and trigger type is displayed, and that the trigger value changes as your turn the knob.

As shown in Fig. 5.1b, 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 updated 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.

Jupyter Notebook 5.1. Save a scope trace obtained while triggering on the falling edge of the ramp function.

5.3 Transient Response of an *RC* Circuit

Set 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. 5.2, use a BNC Tee adapter to split your signal into two identical outputs. Send output of the tee directly to Channel 1 of your scope. Send the other copy to a BNC-alligator-pair cable.

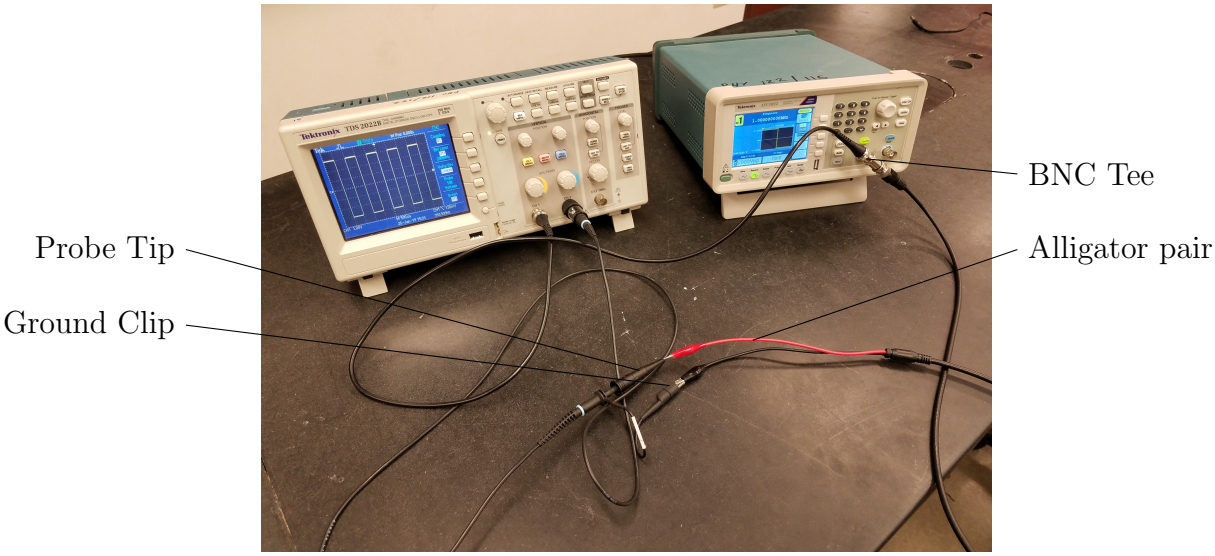


Figure 5.2: A setup for connecting the scope probe directly to the output of the function generator.

Install an oscilloscope probe at Channel 2 of your scope (see Fig 4.4). 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.

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 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. Adjust all the necessary settings for Channel 2 so that it produces 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.

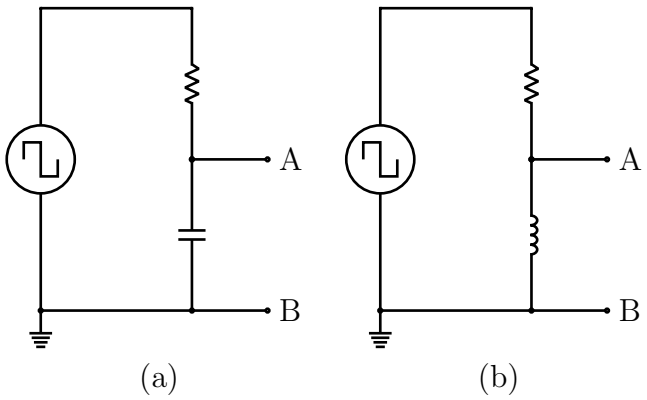
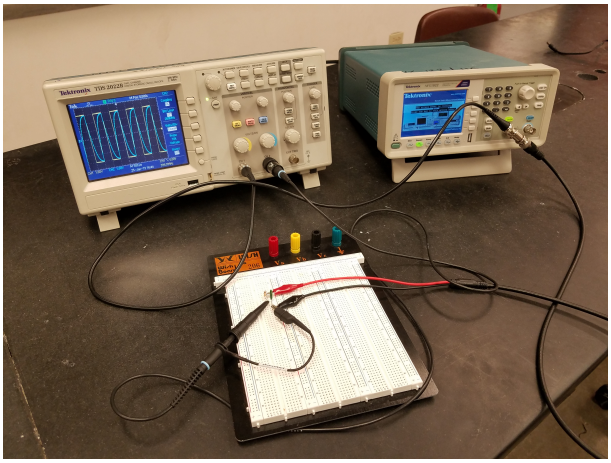


Figure 5.3: A function generator driving an (a) RC circuit, and (b) RL circuit.

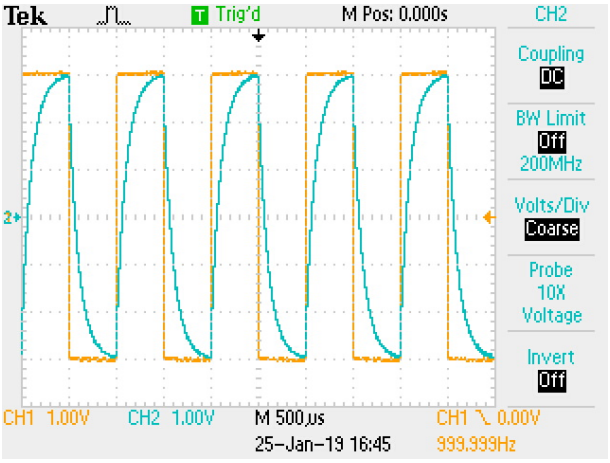
Select a $C = 10$ nF capacitor and an $R = 10$ kΩ.

Logbook Entry 5.1. Using the given values of resistor and capacitor, calculate and record in your logbook the time constant of the RC circuit. Using your DMM, measure and record the actual resistance and capacitance of your components before installing them.

Construct the circuit in Fig. 5.3a on your breadboard, as shown in Fig. 5.4a. Your function generator acts 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. 5.4b, each time the square wave changes polarity, the capacitor begins charging or discharging until it reaches equilibrium with the new voltage, revealing the characteristic exponential transient response.

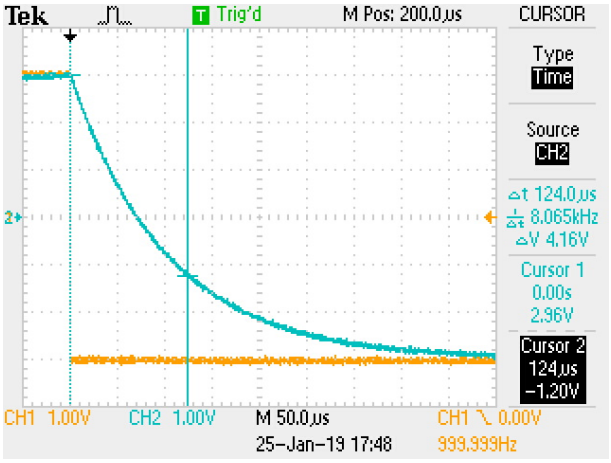


(a)

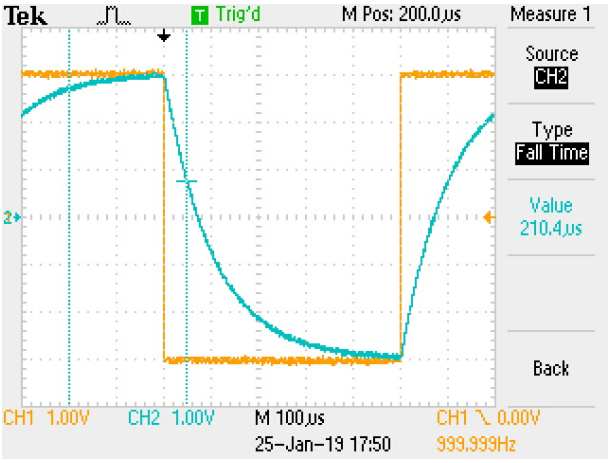


(b)

Figure 5.4: Setup for the (a) RC circuit measurement, and (b) example scope trace showing exponential curve.



(a)



(b)

Figure 5.5: 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.

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. 5.5a. 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.

Logbook Entry 5.2. Record one measurement every ~ 25 μs starting from $t = 0$ μs to 400 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.) Record a rough sketch of your waveform.

For an exponential decay with time constant τ, the rise-time (or fall-time), when defined as the time interval between 10% and 90% values, is given by:

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

Logbook Entry 5.3. Your scope can also directly measure the fall time of a waveform. For an accurate measurement, setup the scope so that one complete falling edge is on screen, as shown in Fig. 5.5b. Press Measure and then set source to CH2 and Type to "Fall Time". You might see this value fluctuating. In such circumstances, it's a good practice to record 3-5 different measurements to help intepret your results later.

The manufacturer does not specify the accuracy of the fall-time measurement, but from other specifications we can reasonably infer an accuracy of about 3% for the setting used in this lab. The values of R and C measured with your DMM are much more precisely known than 3%. Does your predicted time-constant based on the measured values of R and C agree with the measured fall time?

This is a **sign-off point** for this lab.

Jupyter Notebook 5.2. Plot your collected RC circuit voltage-versus-time data as discrete data points. On the same plot, compare the data to the expected exponential decay function as a continuous curve, using the RC time constant calculated from the measured values of R and C. Make sure to have appropriate axis labels and a legend indicating "Data" and "Prediction".

5.4 Transient response of an RL circuit

Not everyone will have time to finish this entire section, do your best with your available time.

Logbook Entry 5.4. 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.

Logbook Entry 5.5. Wrap an inductor around the provided wooden dowel. Estimate it's inductance by modifying your calculation above accordingly and record the value in your logbook.

Obtain a $R = 47\ \Omega$ resistor.

Logbook Entry 5.6. Using your DMM, measure and record the resistance of your resistor.

Turn down the supply to 2.5 V peak-to-peak. Build the circuit in Fig. 5.3b using your homemade inductor and your resistor.

Logbook Entry 5.7. Using your digital scope, measure the fall-time of your RL circuit and use it to determine a measured value for the inductance. Calculate the inductance of your coil homemade coil and compare to your theoretical estimate. Briefly explain your result in your logbook

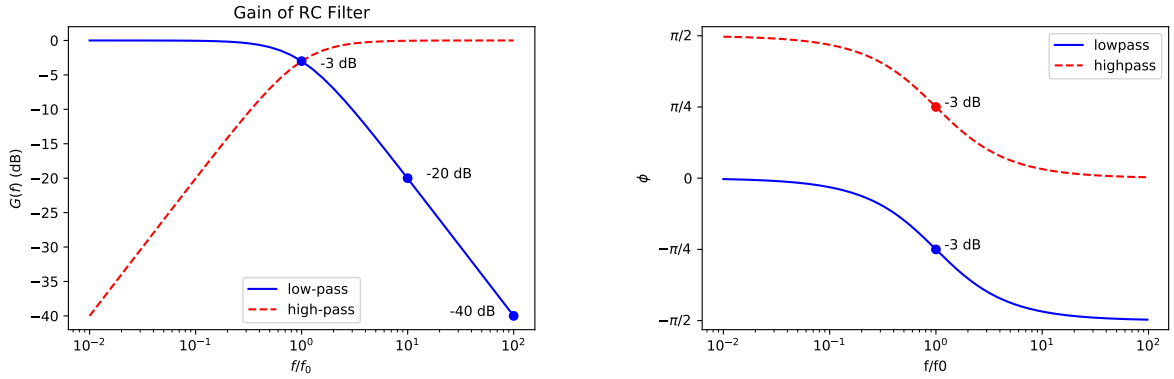
This is an additional **sign-off point** for this lab.

Chapter 6

Passive Filters

6.1 Introduction

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. 6.1. You will also build and measure the performance of an RLC band-pass filter and determine its resonant frequency and quality (Q) factor. For this lab there are both logbook and Jupyter notebook entries.



(a) (b)

Figure 6.1: 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.

6.2 Low-pass Filters

Logbook Entry 6.1. Calculate the corner frequency f_0 (aka the frequency of -3 dB point) for the RC filters shown in Fig. 6.2 for $R_1 = 1.5$ kΩ and $C_1 = 10$ nF.

Set your function generator to produce a sine function with a peak-to-peak voltage of 4 V and set the frequency to the calculated value of the corner frequency. Build the circuit shown in Fig. 6.2a using $R = 1.5$ kΩ and $C = 10$ nF.

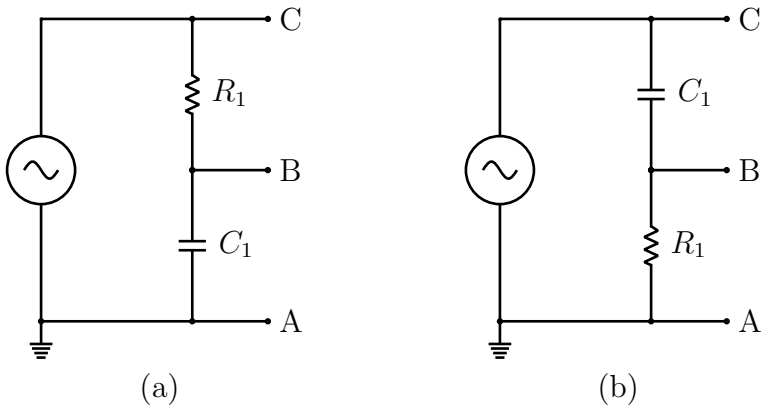


Figure 6.2: Circuit diagrams for RC (a) low-pass filter and (b) high-pass filter. The input voltage is $V_{\text{in}} = V_{\text{AC}}$ while the output voltage is $V_{\text{out}} = V_{\text{AB}}$.

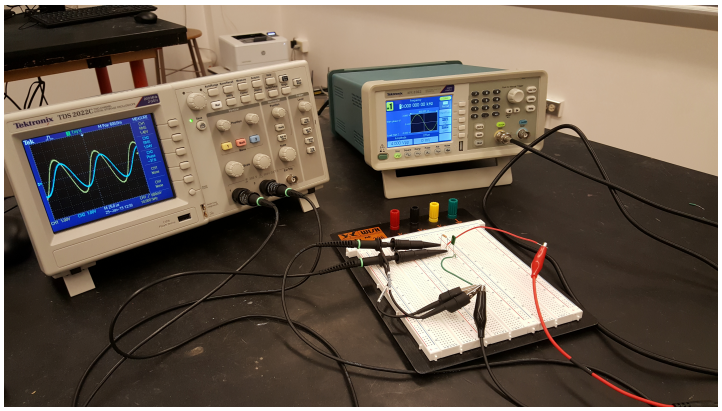


Figure 6.3: Setup for RC low-pass filter.

Logbook Entry 6.2. 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 breadboard, 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.

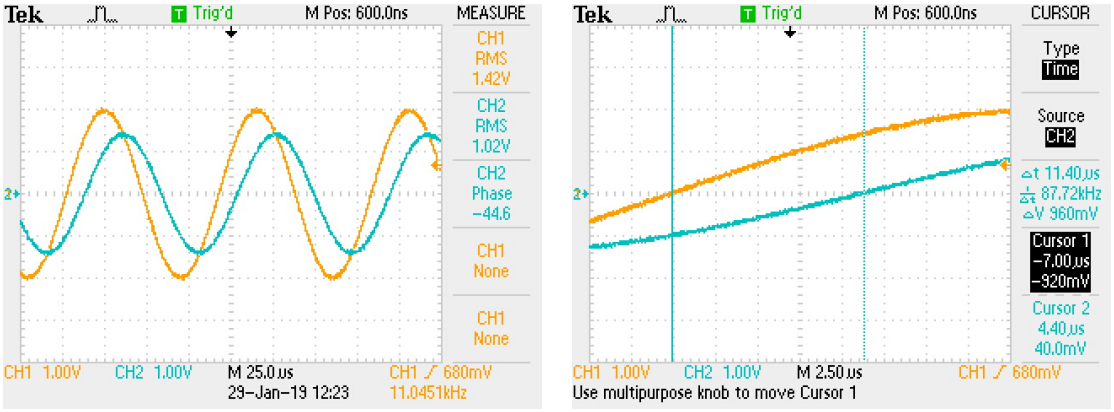
You will be using two scope probes in this lab, and as always some care must be taken with respect to grounding when using your scope. Your function generator has already set the earth ground point in your circuit at the black alligator clip, which is point A in the circuit diagram. The scope grounding 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.

The setup, including the scope, is illustrated in Fig. 6.3. 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.

Logbook Entry 6.3. 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 6.4: 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 6.4, the output has an amplitude of 1.4 divisions while the input has an amplitude of 2.0 divisions, so the gain is $G \sim 1.4/2.0 \sim 1/\sqrt{2}$. After adjusting the horizontal scale and using the cursors menu, as shown in Fig 6.4, the offset in time Δt between when each signals crosses zero is determined to be 11.4 μs. The offset in degrees is simply:

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

In this case, $\phi \sim 45°$. 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 $1/\sqrt{2}$ 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. 6.4. 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.

Logbook Entry 6.4. Sketch the setup in your logbook. 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.

6.3 High-pass Filter

Logbook Entry 6.5. Sketch the setup in your logbook. Using the same components as in the previous section, build the high-pass filter of Fig. 6.2b, and repeat your measurements at the nine different frequencies:

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

This is a **sign-off point** for this lab.

6.4 Analysis

Jupyter Notebook 6.1. Plot the measured gain as a function of frequency for your high-pass and low-pass filter, and compare to the expected response (using measured values of R and C). Make sure to have appropriate axis labels and a legend indicating "Data","Prediction",.... . Use a more descriptive wording instead of Data and Prediction (given here only as example). Describe (in your Jupyter notebook) how those predictions are matching your measured data?

Jupyter Notebook 6.2. Plot the measured phase shift as a function of frequency for your high-pass and low-pass filter, and compare to the expected response. Make sure to have appropriate axis labels and a legend indicating "Data","Prediction",.... . Use a more descriptive wording instead of Data and Prediction (given here only as example). Describe (in your Jupyter notebook) how those predictions are matching your measured data?

Jupyter Notebook 6.3. Plot the measured gain **in decibels** as a function of frequency for your high-pass and low-pass filter, and compare to the expected response (using measured values of R and C). Make sure to have appropriate axis labels and a legend indicating "Data","Prediction",.... . Use a more descriptive wording instead of Data and Prediction (given here only as example).

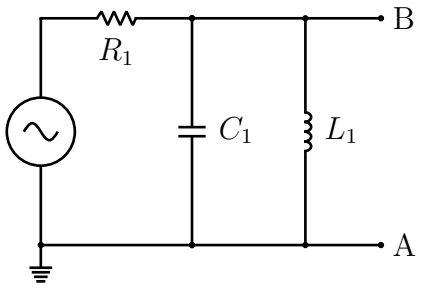


Figure 6.5: Circuit diagram for RLC band-pass filter.

6.5 Band-pass Filter

Build the circuit in Fig. 6.5 using $R = 1.5$ kΩ , $C = 10$ nF, and $L = 1$ mH.

Logbook Entry 6.6. Using your DMM, measure and record the actual resistance and capacitance of your components before installing them. Use an LCR meter (BK Precision 878B) to measure the actual inductance of the inductor and record it. Sketch the setup in your logbook. Calculate f_0 using the measured values of your components.

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

Set the frequency of the function generator to f_0 and the peak-to-peak voltage to 4 V.

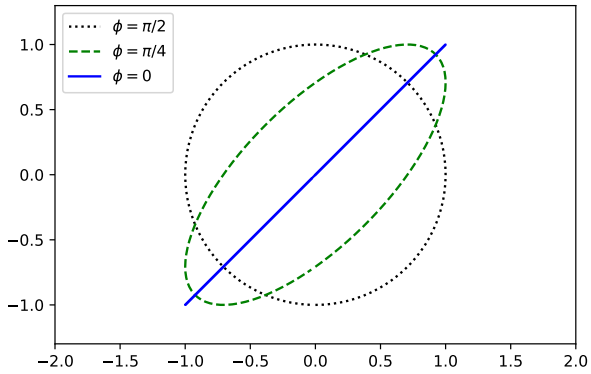


Figure 6.6: Expected scope traces in XY display mode for a relative phase $\phi = \pi/2$, $\phi = \pi/4$, and $\phi = 0$. It is easy, accurate, and somehow deeply satisfying to tune the frequency until the ellipse collapses into itself, forming a line.

We'll measure the actual resonant frequency using a trick: an XY mode of the scope. 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.

Logbook Entry 6.7. You can set you scope in XY mode and adjust the frequency until the ellipse collapses to quickly and accurately find the resonance frequency. Record the value in your logbook.

Add comment in your logbook about how this measurement compares to the calculated resonance frequency.

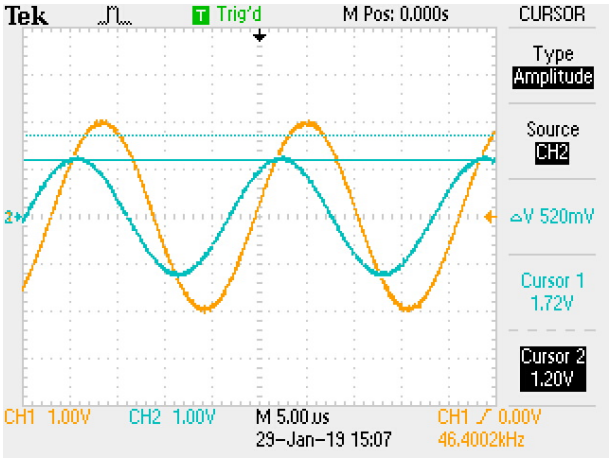


Figure 6.7: Set Cursor 1 to the peak of Signal 2 at the resonant frequency, then set Cursor 2 to this value times $1/\sqrt{2}$. The frequencies f_+ and f_- can be determined by adjusting the frequency until the output amplitude reaches the level of Cursor 2, as shown here for f_-.

Logbook Entry 6.8. 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 $1/\sqrt{2}$ 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 record at which frequencies the amplitude reaches this line. Calculate the bandwidth and record it in your logbook.

Logbook Entry 6.9. From your measurement of f_0, f_+, and f_- calculate the measured Q-factor of your circuit. Record this value in your logbook.

This is a **sign-off point** for this lab.

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. For this lab there are only Jupyter notebook entries.

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 (If documentation is unclear to you, check the examples). 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
```

Jupyter Notebook 7.1. 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 two 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%.

Jupyter Notebook 7.2. Increase the number of pseudo-experiments to $n_{\text{exp}} = 100000$. The agreement should improve to better than 1%. Be certain to print your test cases and the results in a clear way in your notebook.

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.

Jupyter Notebook 7.3. Construct and plot 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 $n_{\text{exp}} = 1000$.

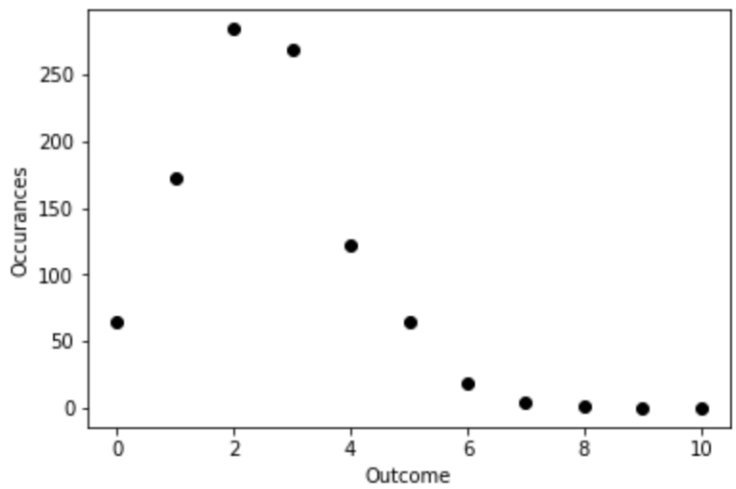
Histograms are most often used to handle continuous data, so you have to take a bit more care when using them to display discrete data (integers) as we are doing here. Consider the bin edges array `edges` that was returned in this example:

```
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 occurrences 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

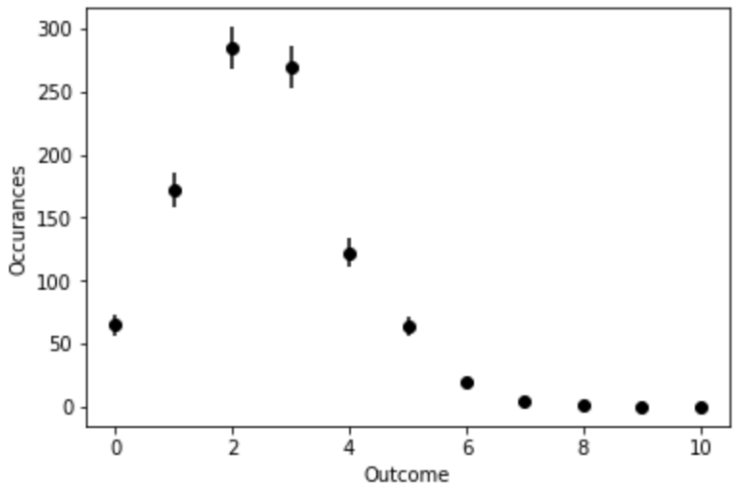
Jupyter Notebook 7.4. Add error bars to your histogram and make a new plot.

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')
```



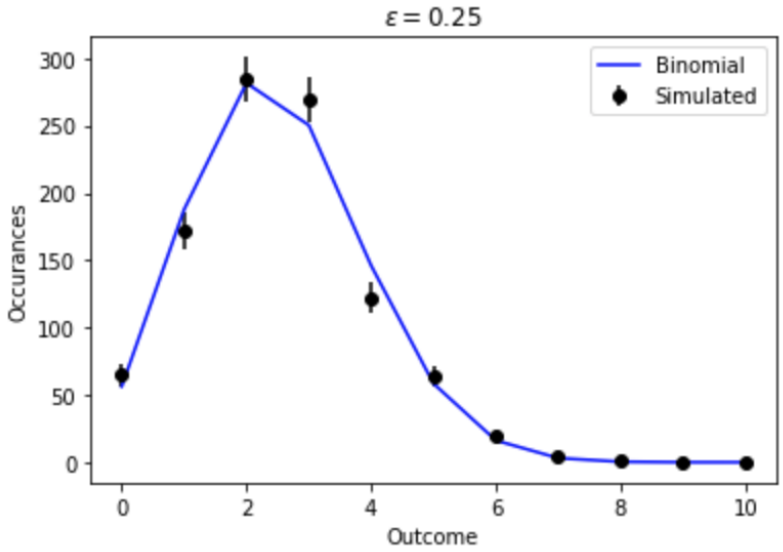
This provides an intuitive visual indication for the size of the statistical fluctuations associated with the counts reported by the histogram. Notice that the poorly named `errorbar` function plots *both* the data point and the error part, so it is a replacement for the `plot` function, not something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

Jupyter Notebook 7.5. Compare the output of your Monte Carlo simulation with the Binomial distribution PDF with $n_{\text{exp}} = 1000$ and $n_{\text{try}} = 10$, and $\epsilon = 0.75$. See example for $\epsilon = 0.25$ in 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:

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

we see that $\epsilon \to 0$ as $n_{\rm try} \to \infty$ 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.

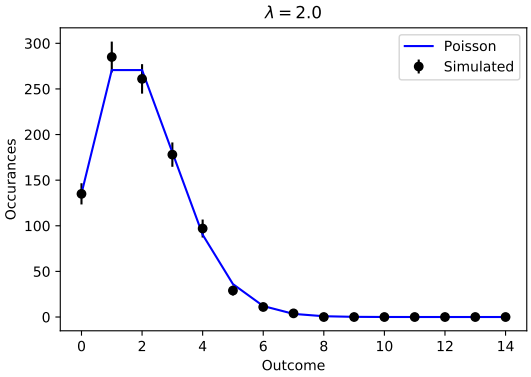


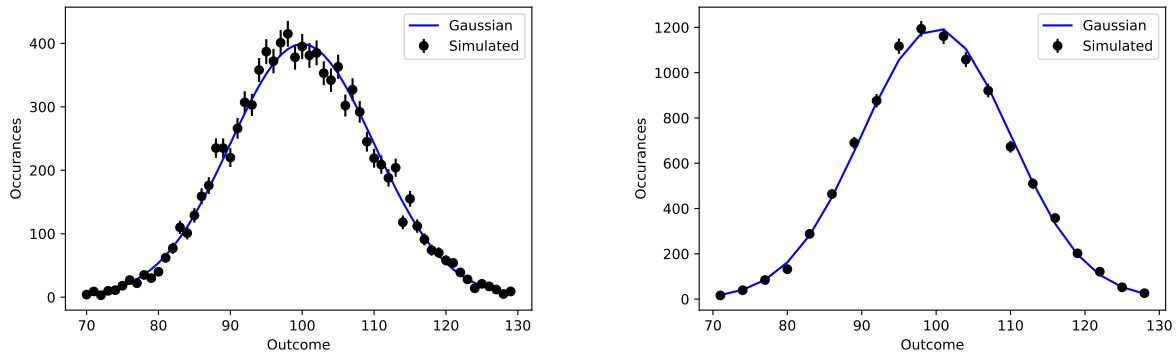
Figure 7.1: Example of the expected result for the Monte Carlo simulation data in comparison to the Poisson PDF for for $\lambda = 2$.

Jupyter Notebook 7.6. In the Poisson limit, compare the output of your Monte Carlo simulation to the Poisson PDF for $\lambda = 5.2$. Plot the histogram with 15 bins for the outcomes: 0,1,2,3,...,14. An example for $\lambda = 2$ 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.:



(a) (b)

Figure 7.2: Example of the expected result for the Monte Carlo simulation data in comparison to the Gaussian PDF for $\lambda = 100$.

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

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

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

Jupyter Notebook 7.7. 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. 7.2a.

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. Each bin is now 3 integers wide. If we live this as is, edges[:-1] will position the x value of the bin at its leading edge: 70, 73, 76 , ... and 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.
  ```

  This would position the x value of the bins at 71.5, 74.5. 77.5, ... and that seems more reasonable choice. In this specific case of discrete data which doesn't extend all the way to the right edge this is still slightly biased. The first bin in our example represents the count of all outcomes in the range from 70 (inclusive) to below 73 (exclusive). So its center should be at 71. To be precise for this specific case, 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 3 integers wide.

Jupyter Notebook 7.8. 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.

Chapter 8

Statistics of Radioactive Decays

8.1 Introduction

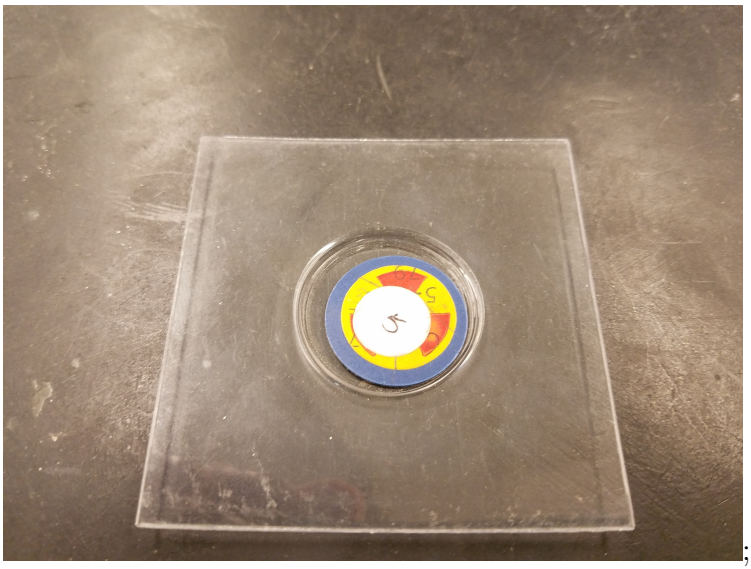
 ;

Figure 8.1: A sealed radioactive source. A small amount of Cs-137 is contained within the small button shaped piece of plastic. For your safety, the sources will be handled only by the TA.

In this lab, you will use a Geiger Counter to study the statistics of radioactive decays. For this lab there are both logbook and Jupyter notebook entries.

8.2 Precautions

Precautions with the Geiger counter:

- Leave the cable from the Geiger counter controller to the Geiger counter in place *at all times*. This carries voltages of approximately 1000 volts. If you leave the cable in place, nothing can be inadvertently plugged in (including fingers).
- Leave the Geiger tube in its holder. It has a thin front window which is easily broken.

- Do not set the high voltage higher than 1000 volts.

Precautions with the radioactive source:

- See Fig. 8.1 to familiarize yourself with what the sources look like.
- Don't touch the source.
- Leave the source in the tray at all times. The TA will provide the sources and handle moving them from place to place.
- Radiation falls off as $1/r^2$. So minimize your time near sources and maximize your distance from them.

8.3 The Geiger Counter

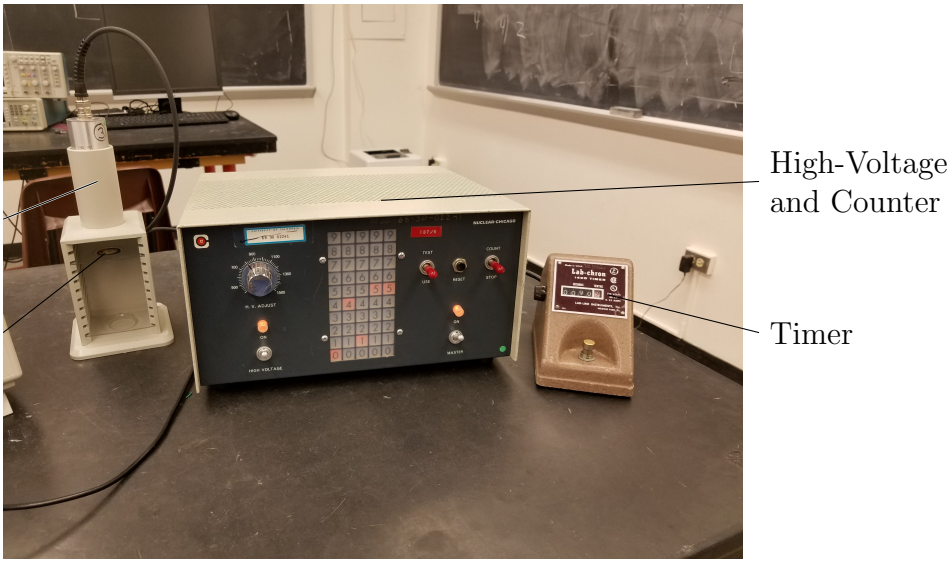


Figure 8.2: The Geiger Counter assembly.

To begin, familiarize yourself with the counter and timer features of your Geiger counter assembly using the built-in test mode. Your lab bench will already be prepared with a Geiger Counter assembly as shown in Fig. 8.2. Ensure that the high-voltage (HV) is off by turning the knob labeled "H.V. Adjust" counter-clockwise all the way to zero. Now put the Geiger counter into test mode by flipping the left red switch to "TEST". Flip the right red switch to "COUNT" and you should see the counter display begin incrementing. Push the button on the front of your timer and you should see the Timer turn on and off. Leave the timer incrementing. Now flip right red switch to "STOP", and observe that the both the counter and the timer stop simultaneously. The knob on left side of the old-school lab timer can be used to reset the time. Keep turning the knob clockwise until the time reads 0. Use the black button on the Counter to reset the count to zero.

Flip the right switch to "COUNT" and then back to "STOP" when 10 seconds have passed. During this time, the 60 Hz test signal should increment the counter close to 600 times. Try this a few times and make sure you can reliably count close to 600 test pulses in a 10 second interval.

You should reset the count each time, but there is no need to reset the timer. Simply stop when the timer reaches the next factor of ten. Due to your reaction time, you may well stop at one-to-two tenths of a second later. This is OK, and will only add less than a few percent error to your measurements over 10 second intervals.

8.4 High-Voltage Calibration

When you are confident that you know how to operate the timer, switch the left red switch to "USE" mode. Unless a sealed radioactive source is already in place, ask the TA to provide you with a source in the second shelf from the top of your Geiger tube holder. Switch the right switch to "COUNT" mode. With the HV off, you should not see any pulses. Turn the HV up until you begin to see counts increment on the display, and continue to the next interval of 50 volts (e.g. if it first starts incrementing at 730 volts, set the dial to 750 volts). Count the number of events in a ten second interval.

Repeat this measurement in 50 volt steps up to 1000 volts. Do not exceed 1000 volts.

Jupyter Notebook 8.1. Plot the rate (in Hz) as a function of high voltage. You should see a plateau region (a leveling off) which indicates the onset of the Geiger mode within the Geiger tube. The Geiger tube has some resistance even in Geiger mode, so do not expect a perfectly flat plateau. From your plot, chose a high-voltage near the beginning of the Geiger mode, and set the high-voltage to this calibrated value. If you are struggling with Python, you can make a rough plot by hand in your logbook to determine the plateau region, and leave the fancy plot for later.

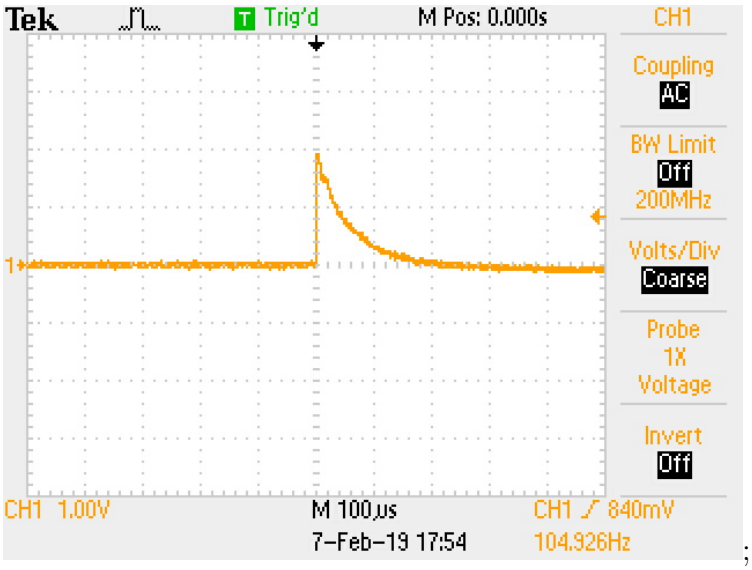


Figure 8.3: An example Geiger counter pulse.

Connect an oscilloscope to the output of the counter assembly (on the back, labeled "SCOPE"). Adjust your scope to view the Geiger pulses like that of Fig. 8.3. Note that the Geiger counter output contains a DC component in addition to the AC pulse, so you will want to use your scope in AC coupling mode which will remove the DC component and allow you to see the pulse. You will also want to see the attenuation to 1X because you are not using an attenuating probe.

Logbook Entry 8.1. Sketch a typical Geiger counter pulse in your logbook and indicates the pulse height and time duration.

8.5 Data Collection

Even in today's world of digital automation, it is still useful to know how to collect a small amount (up to a few hundred data points) of data manually. Often in the lab, you have one-off measurements that you would like to make without investing in automation.

In this section, you will collect data manually for about one hour. Practice a routine with your lab partner that allows you to take and record the data as fast as possible. For instance, person A should operate the counter, and person B should use the PC. Person A turns the counter on for ten seconds, turns it off, and says (quietly) "OK". Person B records the value on the PC and says "Go". Person A resets the counter and continues. Remember that there is no need to reset the Timer each time, which would take too long, and which would actually be counterproductive (if you consider the effect of a roughly constant reaction time.)

Practice your routine a few times, and make sure your count is near 1000 events in a ten second interval. Then record 120 data points.

When you have finished recording your data with the radioactive source, ask your TA to remove the source and return it to the radioactive locker.

Now record an additional 120 data points with no source, to measure the background radiation rate. You should record around 3 background counts per 10 second interval.

8.6 Analysis

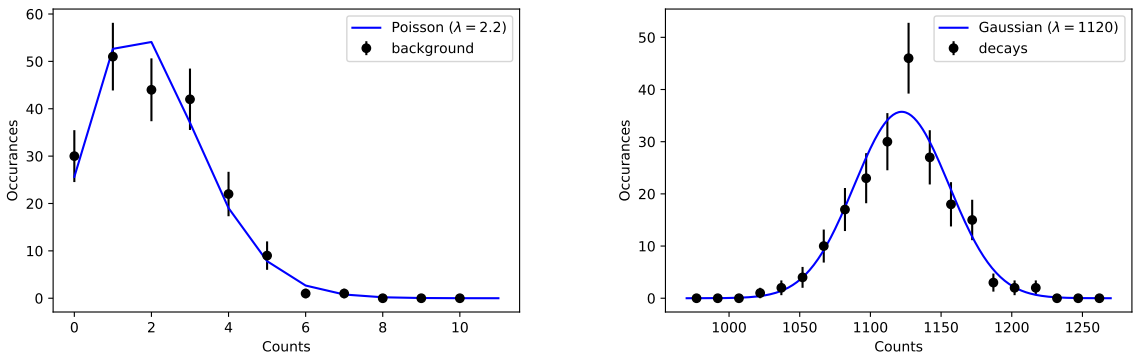


Figure 8.4: Numerical simulation of the experiment for (a) background radiation only, and (b) radioactive source present.

Using Scientific Python, measure the mean and variance of your collected background and source data. Then produce histograms to display your data as in Fig. 8.4. For the background data, plot the histogram for eleven bins: 0,1,2,...,10. For the source data, plot about 20 bins covering a few hundred counts around the mean value.

Jupyter Notebook 8.2. Compare your collected background data to a Poisson distribution, appropriately normalized, with a mean set to the mean of your data.

Jupyter Notebook 8.3. Compare your collected source data to a Gaussian distribution, appropriately normalized, with a mean set to the mean value of your data, and sigma set to the square root of your mean.

This is a **sign-off point** for this lab.

Chapter 9

The Central Limit Theorem and Experimental Uncertainties

9.1 Introduction

In this lab, you will produce a numerical demonstration of the central limit theorem. You will also model the propagation of uncertainties and compare with the calculated uncertainties. For this lab there are only Jupyter notebook entries.

9.2 Sampling Distributions

Scientific python provides functions to draw random samples according to various distributions. In today's lab, we will draw samples uniformly in the interval $[-1, 1]$, as demonstrated in Fig. 9.1. The line

```
r = np.random.uniform(low=-1.0, high=1.0, size=NEXP)
```

creates a NumPy array `r` which contains `NEXP` entries, with each entry chosen uniformly and randomly in the range from -1 to 1. In the example, these events are displayed in a histogram. When plotting histograms with plenty of statistics (one million entries here) and fine binning (60 bins here) it is usually preferable to use lines instead of points with error bars for plotting the histograms, as is done in this example. Notice, however, that even with one million events, there are still statistical fluctuations which prevent the curve from being perfectly smooth.

In Fig. 9.2, entries are instead drawn from the Gaussian distribution with the line:

```
r = np.random.normal(loc=5.0,scale=1.5,size=NEXP).
```

The histogram is plotted with a logarithmic y scale:

```
plt.semilogy()
```

which results in the Gaussian distribution appearing as a parabola. The histogram is compared to the Gaussian PDF appropriately normalized:

```
x = np.linspace(MIN,MAX,100)
y = NEXP*binsize*stats.norm.pdf(x,loc=MEAN,scale=SIGMA)
```

```
NEXP  = 1000000
NBINS = 60
MAX   = 3
r = np.random.uniform(low=-1.0, high=1.0, size=NEXP)
h,edges = np.histogram(r,bins=NBINS,range=(-MAX,MAX))
cbins = (edges[:-1] + edges[1:])/2.0
plt.plot(cbins,h,"k-")
plt.xlabel("$x$")
plt.ylabel("Entries")
```

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

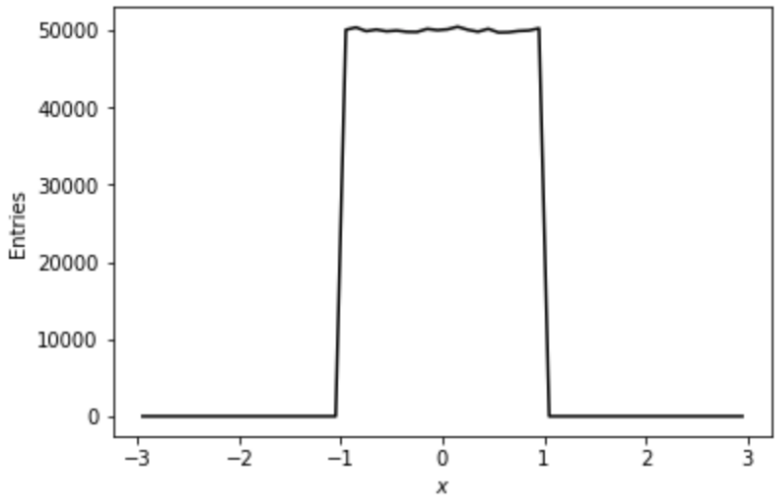


Figure 9.1: Sampling from the uniform distribution.

```
from scipy import stats
NEXP  = 200000
NBINS = 60
MIN   = 0
MAX   = 10
MEAN  = 5.0
SIGMA = 1.5
r = np.random.normal(loc=5.0,scale=1.5,size=NEXP)
h,edges = np.histogram(r,bins=NBINS,range=(MIN,MAX))
cbins = (edges[:-1] + edges[1:])/2.0
plt.plot(cbins,h,"k-",label="Monte-Carlo")
binsize = float(MAX-MIN)/NBINS
x = np.linspace(MIN,MAX,100)
y = NEXP*binsize*stats.norm.pdf(x,loc=MEAN,scale=SIGMA)
plt.plot(x,y,"r--",label="PDF")
plt.xlabel("$x$")
plt.ylabel("Entries")
plt.semilogy()
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1a26a4ce10>
```

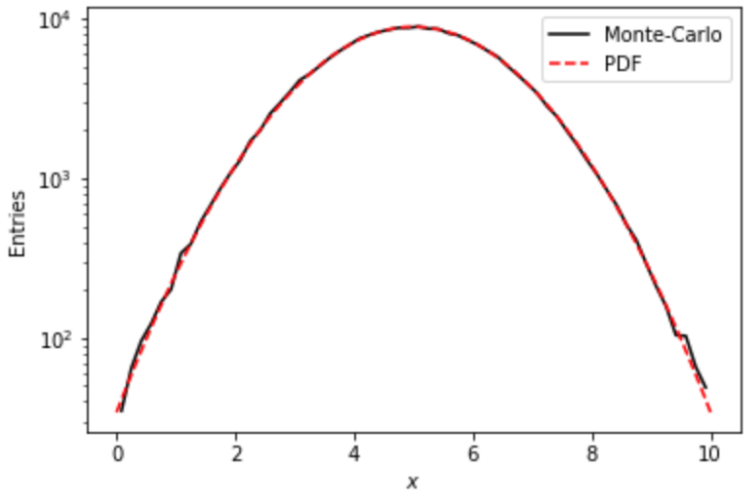


Figure 9.2: Sampling from the Gaussian distribution and comparison with the Gaussian PDF.

9.3 Demonstration of the Central Limit Theorem

In this section, you'll show that average value of random variables chosen uniformly from -1 to 1 approaches a Gaussian distribution, consistent with the central limit theorem. First create a 2-D array of size `NEXP` by `NAVG` filled with uniform random values in the interval from -1 to 1, as follows:

```
r = np.random.uniform(low=-1.0, high=1.0, size=(NEXP,NAVG))
```

Then calculate averages values from `NAVG` entries:

```
x = np.sum(r, axis=1)/float(NAVG)
```

From the Central Limit Theorem, we expect the entries in x to approach a Gaussian distribution.

Jupyter Notebook 9.1. Set `NEXP` to 1000000 for plenty of statistics. Produce three different histograms with 40 bins covering the range from -1.2 to 1.2 for three values `NAVG`: 1,2, and 3. Plot all three histogram in the same graph with appropriate legend.

Your plot will show that already for three contributions to the average, the result looks quite Gaussian on a linear scale. For more precise comparison, we will use a log scale and compare to the PDF.

Jupyter Notebook 9.2. Calculate `NEXP`= 1000000 average values `x` for `NAVG`= 10. Calculate the mean value of the entries in `x` using the `np.mean` function. Calculate σ for the entries in x by taking the square root of the output from the `np.var` (variance) function. Produce a histograms with 20 bins covering the range from -0.5 to 0.5 for the average values. Compare with a Gaussian distribution, appropriately normalized, using your calculated values from the mean and sigma. Plot both the histogram and PDF on the same graph, including an appropriate legend. Use a logarithmic y axis.

9.4 Propagation of Uncertainties

Consider two measured values $a \pm \sigma_a$ and $b \pm \sigma_b$. If we calculate the quantity $c = a + b$ or $c = a - b$, the uncertainty on the calculated value c is given by:

$$\sigma_c = \sqrt{\sigma_a^2 + \sigma_b^2}.$$

If instead, we calculate $c = a * b$ or $c = a/b$ the fractional uncertainty on c is given by:

$$\frac{\sigma_c}{c} = \sqrt{\left(\frac{\sigma_a}{a}\right)^2 + \left(\frac{\sigma_b}{b}\right)^2}.$$

In this section, you'll develop a numerical simulation for the propagation of uncertainties under addition, subtraction, multiplication, and division. An example, for $c = a + b$ is shown in Fig. 9.3.

Pick values for a, b, σ_a and σ_b for simulating subtraction: $c = a - b$. Record them out in your logbook. Choose values that are different from what is plotted in Fig. 9.3.

Jupyter Notebook 9.3. Simulate the measurement a by drawing 100,000 random samples from a Gaussian distribution with mean a and sigma σ_a, and do likewise for b. Calculate the values $c = a - b$ from the a and b values. Plot the distribution of all three variables (as in Fig. 9.3) as histograms with 50 bins and an appropriate range. Calculate the mean and variance of the simulated c values and compare to your expectations from the standard propagation of uncertainties.

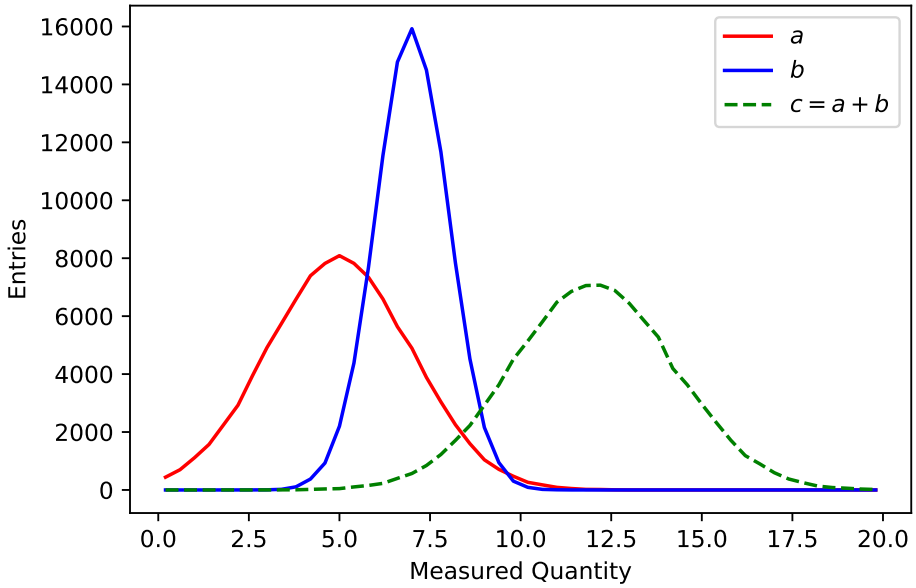


Figure 9.3: Simulation of many measurements of the quantity $c = a + b$.

This is a **sign-off point** for this lab.

Pick values for a, b, σ_a and σ_b for simulating division: $c = a/b$. Record them in your logbook. Choose values that are will allow you to see the distributions of a, b, and c all on the same plot.

Jupyter Notebook 9.4. Simulate the measurements a and b using the same procedure as in the previous plots, but with your new values. Calculate the values of $c = a/b$ from the a and b values. Plot the distribution of all three variables (as in Fig. 9.3) as histograms with 50 bins and an appropriate range. Calculate the mean and variance of the simulated c values and compare to your expectations from the standard propagation of uncertainties.

Chapter 10

The Diode

10.1 Introduction

In this lab, you will measure the IV curve of a diode, use it to predict the operating point of a circuit, and use rectification to provide a DC current source with low ripple voltage. For this lab there are only logbook entries.

10.2 Measuring the I-V Curve of a Diode

In this section you will measure the I-V curve of a 1N914 diode, and compare your results to the curves available from the device data sheet. To avoid taking a bunch of measurements by hand, we will use a trick to plot the curve directly on your oscilloscope using the XY mode.

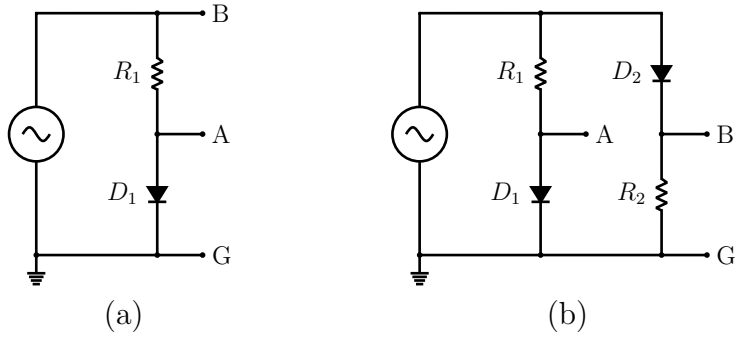


Figure 10.1: Diode circuits for (a) demonstrating rectification and (b) plotting the diode IV curve on your oscilloscope.

Consider (but don't build!) the circuit in Fig. 10.1a. The voltage between points A and B is proportional to the current passing through the resistor, and the voltage between points A and G is the voltage across the diode. So if we could display V_{AB} versus V_{GA} on your scope we could use this circuit. Unfortunately, this is not possible on your scope, because (1) the only valid place to put the scope probe ground shield clips is at the point G and (2) you can only display Channel 1 versus Channel 2 in XY mode.

One solution is to drive two copies of the diode in series with the resistor, but with the component order reversed in the copy, as in Fig. 10.1b. This way, we can connect both of the probe ground

shields as required at point G, put the voltage across the diode on scope Channel 1 by connecting the probe tip at A, and put the voltage across the resistor (proportional to current through the diode) on scope Channel 2 by connecting the probe tip at B.

Build the circuit in Fig. 10.1b using a 1N914 fast switching diode for D_1 and D_2 and $R_1 = R_2 = 10$ kΩ. Set your function generator for high-impedance output, providing AC with peak to peak voltage of 20 V at a frequency of 100 Hz. Before switching to XY mode, make certain that your Channel 1 has no voltage offset (that is, zero voltage is located at the origin) or else your diode output voltage won't be calibrated properly in your output plot. To minimize noise, set the bandwidth limit "On" for both channels (this is available in the menu for each input channel as "BW Limit").

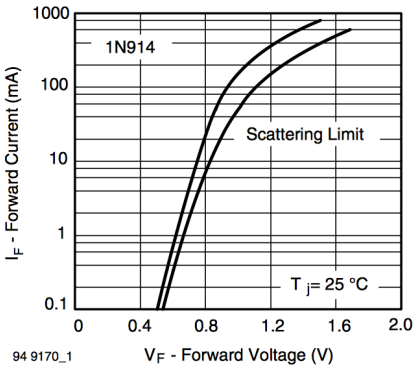
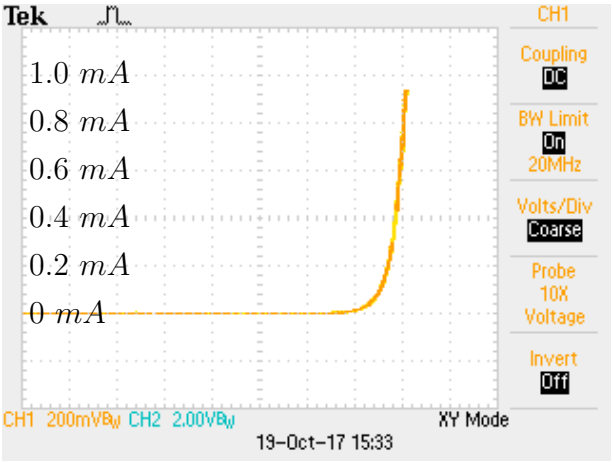


Fig. 1 - Forward Current vs. Forward Voltage

(a)



(b)

Figure 10.2: IV curves for the1N914 from (a) data sheet, and (b) as you will measure in this lab. In the scope trace, the Channel 2 (Y) with scale set to 2 V measures the voltage across a 10 kΩ resistor, so each division corresponds to 200 μA as indicated.

Logbook Entry 10.1. Set the scope to XY mode, and reproduce the diode IV curve in Fig 10.2b. Sketch the curve in your logbook. Make sure to include enough details so that one can read approximate values from it. From your IV curve, estimate the voltage you expect across the diode for a current of 1 mA. Record it in your logbook together with an estimated uncertainty.

Logbook Entry 10.2. The component data sheet in Fig. 10.2a shows the Scattering Limit IV curves for the 1N914. Typical 1N914 diodes will have an IV curve that falls somewhere between the two curvers. Estimate the voltage you expect across the diode for a current of 1 mA and the uncertainty, by considering the range of values consistent with the Scattering Limit. Does your measured value agrees with this expected value?

10.3 Rectifying an AC Signal

Set your function generator to provide an AC source with frequency 100 Hz and peak-to-peak voltage $V_{\rm pp} = 5$ V. Build the circuit in Fig. 10.3 using a 1N914 diode for D and $R = 1.8$ kΩ. With your scope probe ground shield clips both properly connected to the ground at G, monitor the voltage at points A and B.

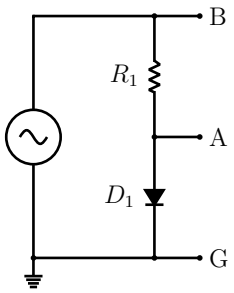


Figure 10.3: A diode rectification circuit.

Logbook Entry 10.3. Sketch the voltage across the resistor R and the voltage supplied by the function generator versus time on the same plot in your lab book. Make sure to include enough details so that one can read approximate values from it.

Logbook Entry 10.4. Using your scopes amplitude measurement feature, measure precisely (i.e. to within 50 mV precision) the voltage drop across the diode at the peak current value, by measuring the difference between Channel 1 and Channel 2 of your scope at the peak. Is this operating point consistent with your results from the previous section and the calculation above?

10.4 Building a DC voltage source

Now build the DC source circuit in Fig. 10.4 using a 1N914 diode for $D_1 = D_2 = D_3 = D_4$ and $R_{\mathrm{L}} = 18$ kΩ. Adjust your function generator to provide a peak-to-peak voltage $V_{\mathrm{pp}} = 20$ V.

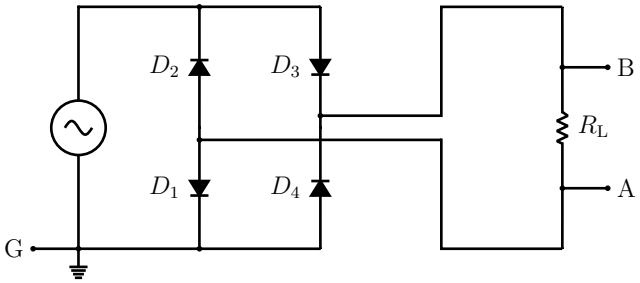


Figure 10.4: A full-wave rectifier.

To measure the performance of our DC source, we would like to measure the voltage across the resistor R_L on the scope. However, notice that the ground for the circuit is located at point G, so you cannot measure the voltage between A and B using a single probe. To make the measurement, connect both probe ground shield clips to the point G as required, and connect the probe tips to points A and B. Next, use your scope's Math mode to subtract Channel 1 to from Channel 2. The result of this operation is the voltage across the resistor R_{L}.

Logbook Entry 10.5. In your logbook, sketch the current as a function of time for a few cycles, and measure the amplitude. Make sure to include enough details that one can read approximate values from it. Explain the shape and the amplitude.

10.5 Controlling the Ripple

The ripple voltage (the residual AC voltage after rectification) for a full-wave rectifier with a capacitance C is given by:

$$\Delta V = \frac{I_{\max}}{2fC}$$

You might notice that this differs from the equation for a half-wave rectifier by a factor of two: $f \to 2f$. This is because the full-wave rectifier uses both the positive and negative halves of the AC signal, so the rectified AC signal has twice the frequency of the original.

Logbook Entry 10.6. Add a capacitor with $C = 1\ \mu$F to your circuit, as in Fig. 10.5 and sketch the resulting waveform for the voltage across the load resistor as measured with your scope. Estimate the ripple voltage.

Logbook Entry 10.7. As your DMM is a handheld device that is not DC coupled, you may use it to measure the voltage across R_L directly. Using your DMM, measure the voltage across R_L in both AC and DC mode. Record those values in your log book.

This is a **sign-off point** for this lab.

For the last tweak, you are going to use a large electrolytic capacitor. **These capacitors are polarized, and will likely "let the smoke out" if you install them the wrong way.**

Logbook Entry 10.8. Making sure the negative terminal is connected as indicated in Fig. 10.5, install a $C = 100\ \mu$F electrolytic capacitor in your circuit and measure the ripple voltage. Record the value in your log book. Note down your observation when comparing this value to the measurements in Meas. 10.6 and 10.7.

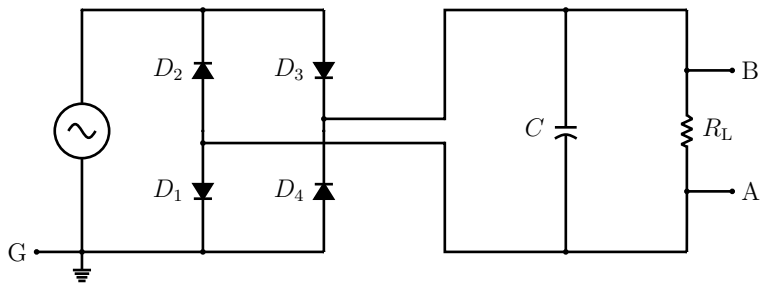


Figure 10.5: A full-wave rectifier with ripple voltage limiting capacitor. When using a polarized electrolytic capacitor, make certain that the negative terminal is connected to the lower half of the figure, as indicated. In other words, the curved sign of the symbol for a polarized electrolytic capacitor is always negative.

Chapter 11

Curve Fitting

11.1 Introduction

In this lab, you will learn about curve fitting with Scientific Python function `curve_fit`. For this lab there are only Jupyter notebook entries.

Given a function to fit $f(x;p)$, with p representing any number of parameters, and a set of measurements y_i and points x_i, the `curve_fit` function determines the best fit parameters by minimizing:

$$\chi^2 = \sum_i \frac{(f(x_i;p) - y_i)^2}{\sigma_i^2}. \tag{11.1}$$

If the uncertainties σ_i are not specified, the function assumes $\sigma_i = 1$, which still finds the correct minimum if the uncertainties are identical to one another.

11.2 Fitting a Straight Line

An example using Scientific Pythons `curve_fit` function to fit a straight line to data is shown in Fig. 11.1. A block of code defining the function we wish to fit, in this case, a straight line, is defined as a function:

```
def line_func(x, a, b):
    return a*x + b
```

In this case, the function requires three parameters (in the computer science sense, not the mathematical sense) the x data in a numpy array as function parameter x, the slope as function parameter a, and the intercept as function parameter b. When called, the function returns the x data multiplied by the value a, with the value b added. We don't directly call this function, but in principle, it could be called like:

```
y_data = line_func(x_data, 2.0, 0.0)
```

to create a numpy array `y_data` constructed from `x_data` with slope 2 and intercept 0.

The next section filling numpy arrays containing the data, and plotting it with error bars should be familiar by now. The fit itself is performed by the line:

```
par, cov = optimize.curve_fit(line_func, x_data, y_data, p0=[guess_a, guess_b])
```

```
from scipy import optimize

# define the fitting function, in this case, a straight line:
# return y = a*x + b for parameters a and b
def line_func(x, a, b):
    return x*a+b

# fill np arrays with the data to be fit:
x_data = np.array([1.0, 2.0, 3.0, 4.0, 5.0, 6.0])
y_data = np.array([21.5, 24.5, 30.1, 40.2, 37.4, 57.2])
y_unc  = np.array([3.0, 3.0, 3.0, 3.0, 3.0, 3.0])

# plot the raw data
plt.errorbar(x_data, y_data,yerr=y_unc,fmt="ko",label="data")

# calculate best fit curve (line_func) for the x_data and y_data
# guess_a and guess_b are initial guesses for the parameter values
guess_a = 1.0
guess_b = 0.0
par, cov = optimize.curve_fit(line_func, x_data, y_data,
                              p0=[guess_a, guess_b])
# retrieve and print the fitted values of a and b:
fit_a = par[0]
fit_b = par[1]
print("best fit value of a:  ", fit_a)
print("best fit value of b:  ", fit_b)

# plot the best fit line:
xf    = np.linspace(0.0,6.0,100)
yf    = fit_b + fit_a * xf
plt.plot(xf,yf,"b--",label="line fit")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
```

```
best fit value of a:   6.494285714297698
best fit value of b:   12.420000000027086
```

```
<matplotlib.legend.Legend at 0x101e725f60>
```

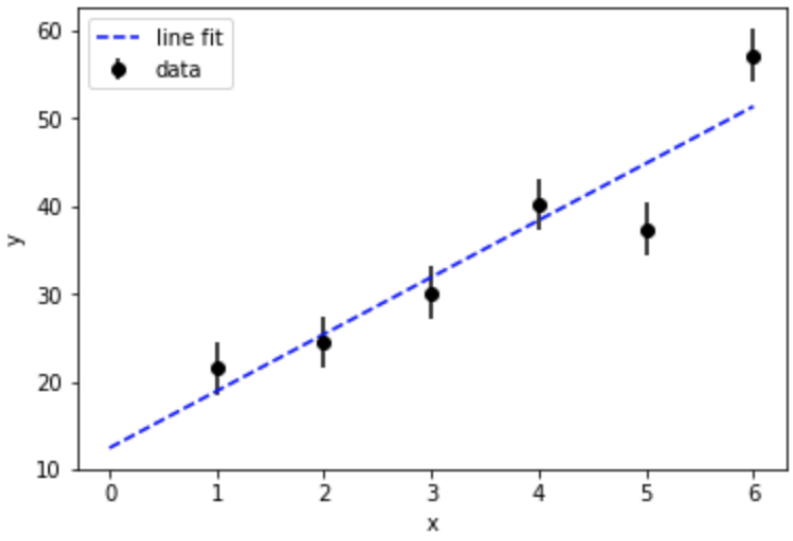


Figure 11.1: Example fitting data to straight line.

Table 11.1: Sample data for straight line fit.

x	$y \pm \sigma_y$
1.0	15.9 ± 3.0
2.0	23.6 ± 3.0
3.0	33.9 ± 3.0
4.0	39.7 ± 3.0
5.0	45.0 ± 10.0
6.0	32.4 ± 20.0

This performs a fit of the function `line_func` defined above to the x and y data contained in the arrays `x_data` and `y_data`. Numerical fits generally find the local minimum, which is not necessarily the global minimum of interest. It is important therefore, especially for complicated fits, to provide an initial guess near the expected fit values. These are provided to the optional, named, function parameter `p0`, which is set to the python list `[guess_a, guess_b]` which contains our initial guesses for the fit parameters a and b. The function performs a least-squares fit to find the best values of a and b which are returned as the numpy array `par`. The function also returns the covariance matrix as the numpy array `cov`.

The remaining code simply uses the best fit values to plot the fitted function as a dashed line. Numerical fits are fickle. Even if you are only interested in the fitted value, you should always plot the best-fit function and compare the results to your data as in important check for your work.

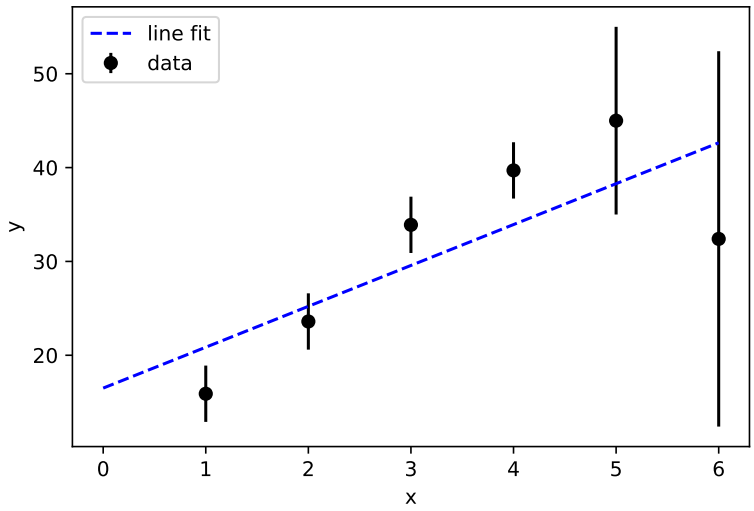


Figure 11.2: This linear fit is biased by the failure to properly account for uncertainties. An unbiased fit would track the well constrained points more closely.

Jupyter Notebook 11.1. Apply code like that of Fig. 11.1 to the data in Table 11.1. Plot the data including error bars and the best-fit function to obtain a result like that of Fig. 11.2.

Notice that the last two data points in Table 11.1 have larger uncertainties than the other data points. However, the call to the curve_fit function does not provide the parameter uncertainties,

and so the function assumes that they all have the value 1. In this case, since the uncertainties are not in fact all the same, the function does not find the correct minimum. The answer is clearly biased (as in Fig. 11.2) toward the poorly measured points, because the function gives these points the same weight as all of the other points.

Jupyter Notebook 11.2. Look-up the `curve_fit` function and the optional parameter `sigma`. Provide the correct uncertainties to the fit and make a new plot with the data and the fit. You should observe that the fit results is no longer biased, and more closely tracks the well constrained left side of the plot.

11.3 Parameter Uncertainties

```
# will fit y_data to a constant value:
def constant_func(x, a):
    return a

# choose y_data indepdent of x values, from
#   Gaussian with a mean of 50 and sigma of 10
x_data = np.arange(100)
y_data = np.random.normal(50.0, 10.0, size=100)

# fit for the best fit constant value, should find the mean:
guess_a = 0.0
par, cov = optimize.curve_fit(constant_func, x_data, y_data, p0=[guess_a])

# determine the best fit parameter and it's uncertainty:
unc = np.sqrt(np.diag(cov))
fit_a = par[0]
unc_a = unc[0]

#print the results
print("mean of y data:  ", np.mean(y_data))
print("fitted constant: ", fit_a)
print("uncertainty:     ", unc_a)
```

```
mean of y data:   48.426944799228266
fitted constant:  48.42694479930867
uncertainty:      0.9711303089930362
```

Figure 11.3: Example obtaining parameter uncertainties.

We will show in lecture that the uncertainty σ_{p_i} on the ith parameter p_i can be determined from the second derivative of the χ^2 function:

$$\frac{d^2\chi^2}{dp_i^2} = \frac{2}{\sigma_{p_i}^2}.$$

In general, for M parameters, the $M \times M$ covariance matrix is calculated as:

$$C(i,j) = 2 \cdot \left(\frac{d^2\chi^2}{dp_i dp_j} \right)^{-1}$$

from which we can see that the diagonals are simply the parameter uncertainties squared:

$$C(i,i) = 2 \cdot \left(\frac{d^2\chi^2}{dp_i^2} \right)^{-1} = \sigma_{p_i}^2.$$

The off-diagonal elements contain information about how parameters are correlated, and for a well designed fit function they should be close to zero.

The `curve_fit` function returns both the best fit parameter values and the covariance matrix:

```
par, cov = optimize.curve_fit(...)
```

For a fit with M parameters, we can obtain an array containing the M parameter uncertainties from the square root of the diagonals of the $M \times M$ covariance matrix:

```
unc = np.sqrt(np.diag(cov))
```

An example is shown in Fig. 11.3, for a single parameter. In this case, the y-values are simply 100 random numbers drawn from a Gaussian distribution with mean $m = 50$ and a width $\sigma_y = 10$. The best fit constant value is simply the mean of the y-values. The uncertainty on the mean value should be:

$$\sigma_m = \sigma_y/\sqrt{N} = 10/\sqrt{100} = 1.0$$

Indeed we obtain a best fit constant value and it's uncertainty close to these expected values.

It's surprising actually, that the fit returns the correct uncertainty. Look through the code carefully and notice that nowhere has the uncertainty on the y parameters σ_y been provided to the fit. So how can it possibly deduce the correct uncertainty $\sigma_m = \sigma_y/\sqrt{N}$?

The answer is that behind the scenes, the `curve_fit` function is being really quite clever (too clever, in my opinion, for a default behavior!) By default, the covariance matrix returned by the `curve_fit` function is scaled by the factor:

$$\alpha = \frac{\chi^2_{\rm min}}{\rm NDF}$$

the minimum value of the χ^2 divided by the number of degrees of freedom (number of data points minus number of parameters). We'll show in lecture that α is around 1 for a least-squares fit with an appropriate model and correct uncertainties. So nominally this factor is one, and has no effect. But consider what happens if the actual uncertainties are σ while the χ^2 used in the fit assumes they are, for example, "1". In this case, the calculated χ^2 is:

$$\chi^2 = \sum_i \frac{(f(x_i;p) - y_i)^2}{1}$$

which differs from the correct χ^2:

$$\chi^2 = \sum_i \frac{(f(x_i;p) - y_i)^2}{\sigma^2}$$

by a factor of σ^2. This means that while the correct value for α is nearly one, the calculated value of alpha will be σ^2. This is precisely the factor needed to scale the squared parameter uncertainties to account for the fact that the initial uncertainty was σ but we assumed 1.

This behavior is controlled by the parameter `absolute_sigma`. By default, the function sets `absolute_sigma = False` and scales the covariance matrix as just described. On the other hand, if you want to simply use the provided uncertainties without re-scaling the covariance matrix, you must remember to set `absolute_sigma=False`. I think this is a really poor choice of default behavior... it's really quite a fancy thing to do implicitly. In cases when you know the uncertainty on your data points, this re-scaling actually results in less correct estimate for the uncertainties. This is because for a good model with proper uncertainties, the factor α is near one, but not exactly one.

Jupyter Notebook 11.3. Repeat the fit in Fig. 11.3, but set the uncertainties on the y values to 10 and also set `absolute_sigma = True` in the fit. Record the uncertainty.

Jupyter Notebook 11.4. Leave the uncertainties on the y values unspecified and set `absolute_sigma = True` in the fit. You should obtain an uncertainty of 0.1. Record your value and explain why you obtained this value.

As a rule of thumb, when using `curve_fit`, if you provide explicit uncertainties, you should remember to set `absolute_sigma=True`. And really, for precision work, you should almost always be providing explicit uncertainties.

11.4 Fitting a Sine Curve

In this section, you will fit the sample data to a sine function:

$$y = A \sin(kx).$$

Use the following sample data:

x	y	x	y	x	y
0	5.3	4	-9.7	8	15.7
1	15.0	5	-17.4	9	18.5
2	19.2	6	-20.5	10	8.6
3	6.8	7	2.1		

Assume the uncertainty is the same for each y value: $\sigma_i = 2$.

Jupyter Notebook 11.5. Plot the sample data including error bars and the best-fit sine wave. Print the best fit parameter values and their uncertainties. This fit is highly sensitive to the initial guess, so make sure you provide good starting values close the correct answer. Remember to set `absolute_sigma=True`.

This is a **sign-off point** for this lab.

Chapter 12

Measurement of Planck's Constant

12.1 Introduction

In this lab, we will measure Planck's constant by measuring the V-I curves of three different colored light emitting diodes (LEDs). This lab has both logbook and Jupyter notebook entries.

An LED is a particular type of diode for which the recombination of electrons and holes produces photons, typically in the visible light spectrum. These diodes have an activation voltage given by:

$$V_{\mathrm{A}} = \phi + \frac{hc}{e}\frac{1}{\lambda} \tag{12.1}$$

where λ is the wave-length of the light produced by the diode, and ϕ is the contribution to the voltage drop due to other effects in the $p-n$ junctions. The diodes we are using have been chosen to ensure that ϕ is approximately constant across all three diodes.

The quantity

$$\frac{hc}{e}$$

can therefore be determine from the slope of the activation voltage as a function of $1/\lambda$.

The 2018 redefinition of the SI is means that the quantity

$$hc = 1.23984193 \text{ eV}\mu\text{m}$$

is technically now exactly known, because the values h, c, and e are now taken as exact values which define the corresponding SI units. Of course, it is still useful and fun to measure this quantity ourselves in the lab. Since we will also be measuring the speed of light, we'll interpret this measurement as our determination of Planck's constant.

12.2 LED Model

For the purpose of this experiment, we will model the LED as an ideal diode with voltage drop equal to the activation voltage V_{A} of Equation 12.1 plus a series resistance R_{LED}. As shown in Fig. 12.1, the effect of this resistance is to replace the vertical line at the activation voltage V_A with a line of slope $1/R_{\mathrm{LED}}$.

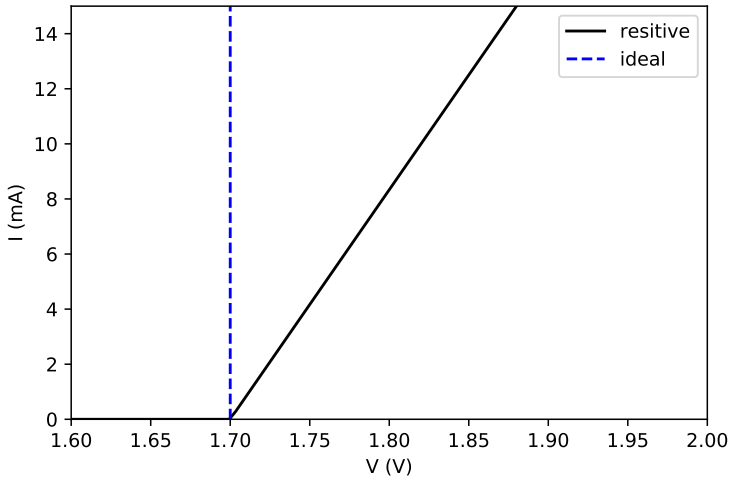


Figure 12.1: The LED model for $V_A = 1.7$ V and $R_{\text{LED}} = 12\ \Omega$

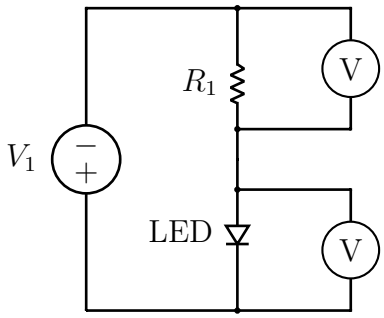


Figure 12.2: experimental setup.

Table 12.1: Instructor data from a red LED not used.

target I (mA)	I (mA)	$V_{\rm D}$ (V)
0.5	0.44	1.62
1.0	0.95	1.66
2.0	1.99	1.71
4.0	4.15	1.77
6.0	6.05	1.81
8.0	8.09	1.85
10.0	10.04	1.88
12.0	12.02	1.91

12.3 $V - I$ Curve from LED.

Construct the experimental setup shown in Fig. 12.2 using a 1 kΩ precision 1% resistor for R_1, a green LED, and using your bench-top DC supply to provide V_1, initially set to 0 V. Connect one voltmeter across the resistor R_1 and another voltmeter across the LED. When constructing your circuit, make sure the LED can be easily removed and replaced. Also keep in mind that the longer lead is the positive terminal of the LED, i.e., the upper terminal of the LED as drawn in the Fig. 12.2. Set both voltmeters to the 20 V setting.

Test your circuit first by turning up the voltage on your supply to about 5 V and checking that the LED lights up. The voltmeter across the resistor $R_1 = 1$ kΩ is effectively measuring the current in mA as 1 V/1 kΩ = 1 mA. Do not misunderstand this statement to mean you should set the multi-meter to the current measurement: we measure the voltage, but from $Ohm's Law$ we know the current in resistor. The voltmeter across the diode is measuring the diode drop $V_{\rm D}$.

Logbook Entry 12.1. With the green LED, take a series of measurements of I and V_D near target values of $I = 0.5, 1.0, 2.0, 4.0, 6.0, 8.0, 10.0, 12.0$ mA by adjusting the voltage provided by your DC supply until the current I, as measured with the voltmeter across R_1, is near the target value. Remember not to waste time fussing to make the measurement at exactly the target value. For instance, measuring at $I = 2.16$ mA instead of the target $I = 2.0$ mA is perfectly acceptable. Simply record the actual measured value of I next to target value, in addition to measurement of $V_{\rm D}$.

Logbook Entry 12.2. Repeat the previous measurement using the yellow LED.

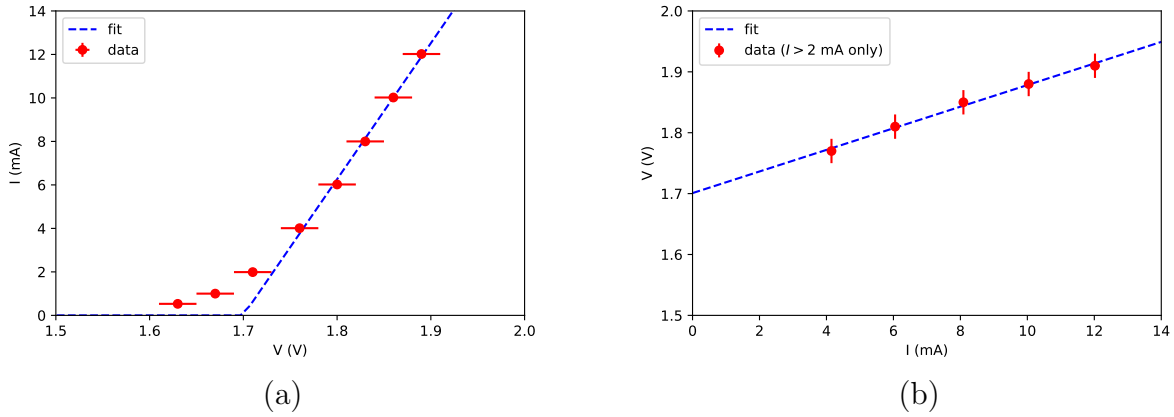
Logbook Entry 12.3. Repeat the previous measurement using the red LED.

This is a **sign-off point** for this lab.

Table 12.2: LEDs used in this experiment.

color	part no.	λ (nm)	max current
green	WP710A10GT	565	25 mA
yellow	TLHY4405	581-594	30 mA
red	TLHR4405	612-625	30 mA

12.4 Analysis



(a) (b)

Figure 12.3: Instructor fit for red LED. The (a) real diode response departs from the simple linear model for $I \leq 2$ mA, so the (b) fit for the linear response is performed for $I > 2$ mA. Note that V is taken as the y-axis for the purpose of fit, because the voltage uncertainties are the dominant uncertainty in the fit. Notice also that the 0.02 V uncertainty reported by the DMM specifications appears to be predominantly systematic.

The V-I response for a red diode as measured by the instructor is plotted in Fig 12.3a. Notice that at large values for the current ($I > 2$ mA) the VI response is linear, indicating that it is dominated by internal resistance of the diode. As lower current ($I \leq 2$ mA) the VI response is exponential, as expected for a real diode. We will fit our simple linear model for the diode response only in the region where this approximation is valid, for $I > 2$ mA. As shown in Fig. 12.3b, perform a linear fit only for the data with $I > 2$ mA.

With the DMM set to the 20 V scale, the uncertainty on your measured values for V and I is approximately 0.02 V and 0.02 mA respectively. Because the measured voltage range is less than a volt, but the measured current range is around 10 mA, it is the uncertainties on the voltage that dominate the uncertainty on both the slope and intercept of the linear function. We therefore choose to use V as the y-axis and and I as the x-axis for the purposes of fitting, because the χ^2 analysis of the curve_fit function only considers uncertainties in the x-axis. There are straightforward ways to handle uncertainties in both x and y, but they add complexity which is best avoided if possible.

Jupyter Notebook 12.1. Using your green LED data, fit the V versus I data to a linear function, and determine the best fit resistance and activation voltage, and the uncertainties. Assume the uncertainty on the measured voltages is 0.02 V for each data point and only consider $I > 2$ mA in the fit. Remember to set `absolute_sigma=True` in your fit so that the fit uncertainties are based on the absolute uncertainties without re-scaling.

Jupyter Notebook 12.2. Repeat the previous fit using yellow LED data.

Jupyter Notebook 12.3. Repeat the previous fit using red LED data.

Jupyter Notebook 12.4. Using the best-fit values of V_A determined from the previous plots, plot V_A of each diode versus $1/\lambda$, and determine the slope (hc/e) and its uncertainty from a linear fit.

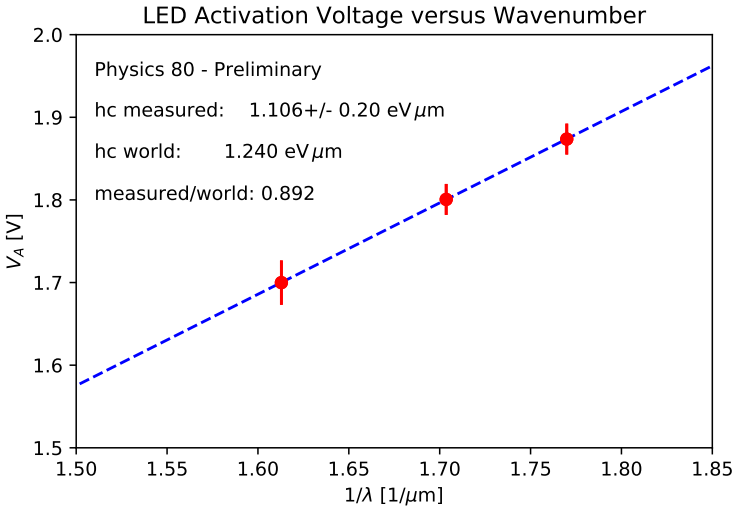


Figure 12.4: Instructor plot for determination of hc.

The instructor plot is shown in Fig. 12.4. Recall that 1 eV is the change in potential energy of one electron passing through 1 V of potential energy, allowing you to conveniently convert the slope in Vμm to eVμm for comparison with the established value $hc = 1.240$ eVμm.

This is an additional **sign-off point** for this lab.

12.5 Systematic Uncertainties

In addition to the statistical uncertainties reported by the fit, their are a number of systematic uncertainties. For this lab, we'll consider one obvious source of systematic uncertainty, which arises from our treatment of the real diode response as a simple linear function.

To determine the size of this systematic uncertainty, we measure the effect of this assumption on our measured valued. One simple way to estimate this effect is to remove the requirement $I > 2$ mA from our analysis. The difference between the measured value for hc as determined with and without the cut on $I > 2$ mA can be interpreted as a systematic uncertainty due to our simplistic model.

Jupyter Notebook 12.5. Repeat your analysis without the requirement $I > 2$ mA and take this as the overall systematic uncertainty on your measurement.

Chapter 13

The Speed of Light

13.1 Safety

The laser used in this lab is of low power. Even so, avoid pointing the laser directly into anyone's eye.

13.2 Introduction

In this lab, you will measure the speed of light in air by measuring the time between sending and receiving a flash of light over a known distance. This lab includes both logbook and Jupyter notebook entries.

The light signal for this measurement is provided by a laser diode. Like an LED, the photons in the laser diode are the result of electrons and holes recombining. The laser diode produces simulated emission of photons from population inversion of holes and electrons injected from p-type and n-type semiconductors into an intermediate layer of un-doped intrinsic semiconductor.

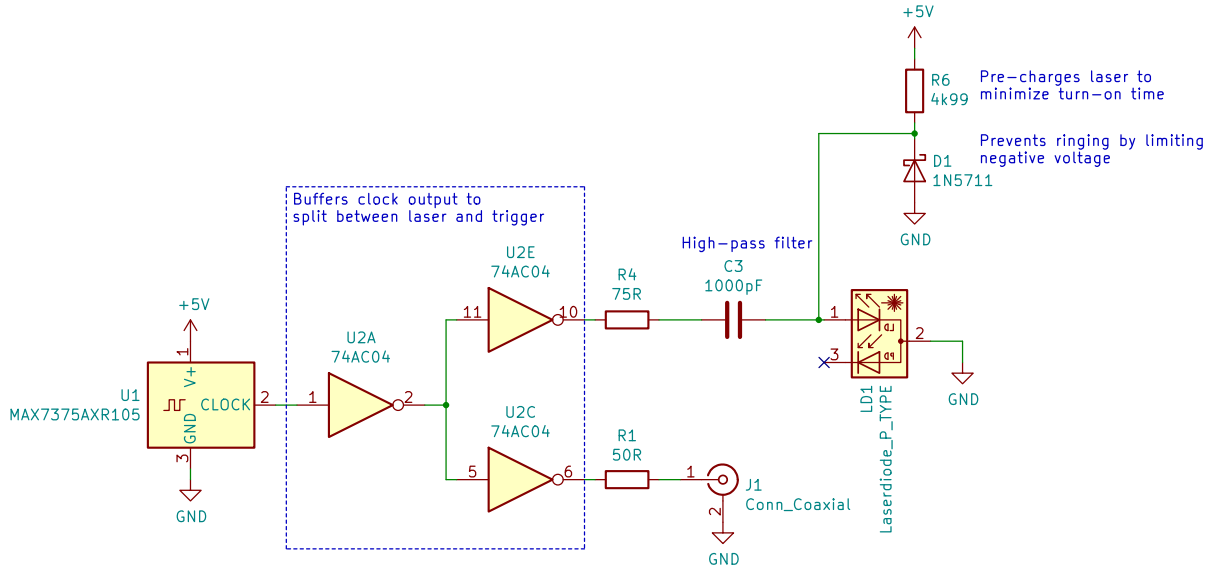


Figure 13.1: Circuit diagram for the pulsed laser diode.

If laser light is produced continuously, we'll have no way to measure a time difference between sending and receiving the pulse. Instead, we'll produce a brief pulse of laser light and a reference signal to indicate the time at which the pulse was sent. The circuit for pulsed laser diode assembly we will be using is shown in Fig. 13.1. You will recognize the passive components consisting of resistors, diodes, and capacitors. The MAX7375 (square symbol) is a silicon oscillator which produces a 1 MHz square wave function. The 74C04 (triangle symbol) is technically an inverter, but here they are used to simply produce two independent copies of the square wave function output. One copy of the square wave function is sent to a BNC connector, as the time reference for sending the laser pulse. The other copy is send through a capacitor, which acts as a high-pass filter, converting the step function into very narrow positive and negative pulses. The diode D1 rectifies this AC signal, so that only the positive signal is used to drive the laser diode, causing a brief pulse of laser light, in sync with the square wave signal which will be available on the scope.

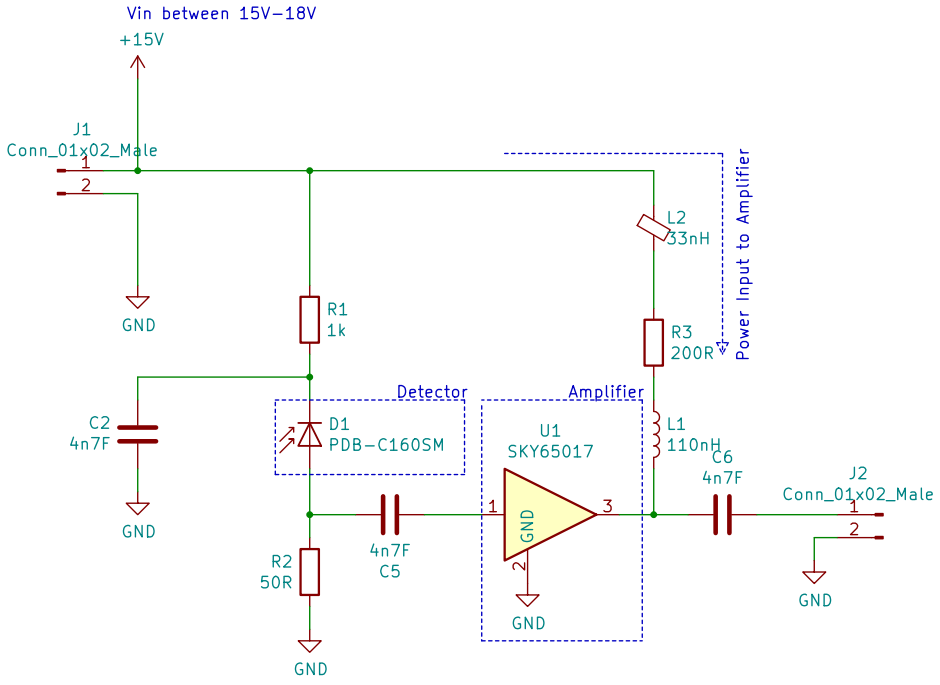


Figure 13.2: Circuit diagram for photodiode detector.

We'll detect the flash of light using a photo-diode. The photo-diode is placed in reverse bias, creating a depletion zone. When photons strike the depletion zone, they excite electrons to create electron-hole pairs, which allows a current to flow. The receiver circuit is shown in Fig. 13.2. The photo-diode D1 is held under reverse bias by the externally supplied DC voltage. The current pulse created when the laser light reaches the photo-diode is amplified by the SKYC5017 broadband amplifier. All amplifiers have limits to their bandwidth, but this amplifier is fast enough to handle the brief laser pulse that we are sending. The input connector for this device has an internally generated DC voltage, so we use the capacitor C5 to isolate this DC voltage from our circuit. The high-frequency AC signal pulse we wish to amplify will see this capacitor as effectively a short-circuit. All amplifiers require external DC power, but this one is a bit peculiar in that the DC power is supplied at the output pin. This explains the use of inductors L1 and L2 and capacitor C6. Remember inductors are a short-circuit to DC and an open circuit to AC, where as capacitors are an open-circuit to DC and a short-circuit to AC. The DC supply is provided to the output pin

through the inductors, but the amplified AC output signal passes through the capacitor.

13.3 Experimental Setup

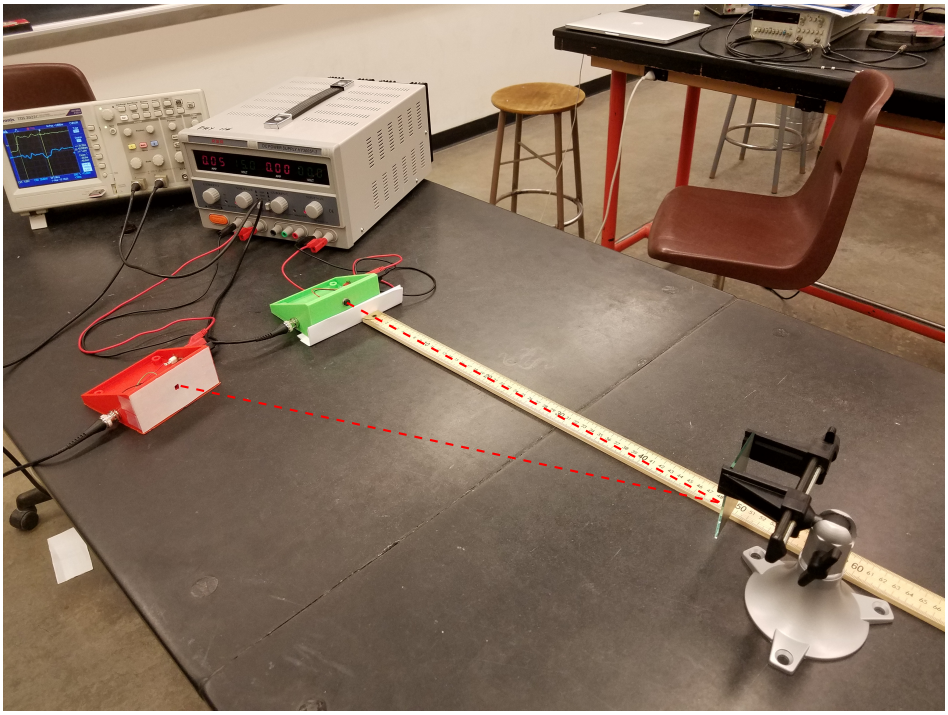


Figure 13.3: Setup.

The setup is shown in Fig. 13.3. You will use your bench-top DC power supply to power the pulsed laser diode and the photo-diode receiver assemblies. The right most pair of output from your supply provides a fixed 5 V DC output, which you will use to power the pulsed laser diode assembly (transmitter). The transmitter is housed in the green box. The reference signal for the transmitter is output on the BNC connector, and should be connected to your channel 1 of your scope, **using a** 50 Ω **terminator.**

The photo-diode receiver circuit is housed in the red box. It should be powered at 10 V from your bench-top DC power supply. The amplified signal output on the BNC connector should be connected to channel 2 of your scope, **using a** 50 Ω **terminator.**

To test your setup and the equipment, point the laser directly into the receiver as shown in Fig. 13.4. Use folded printer paper to adjust the height and orientation of the boxes as needed.

Make certain that your digital oscilloscope has a bandwidth of 100 MHz. The receiver signal is the most intermittent, so to avoid seeing empty wave-forms, you should always trigger on the receiver signal. Set both channels to AC coupling and make certain that the bandwidth limit is off. Set the time-scale to 5 nanoseconds.

An example of the timing measurements you will be making is shown in Fig. 13.5. Ideally, when making a timing measurement, you should use a sharp edge. This is why the reference signal is measured on the rising edge. The receiver signal, however, is quite noisy, so the edge is easily distorted. A slightly more reliable measurement comes from the using the minimum of the pulse. There is significant variation of the pulse height, so it helps to set the trigger level to only select the

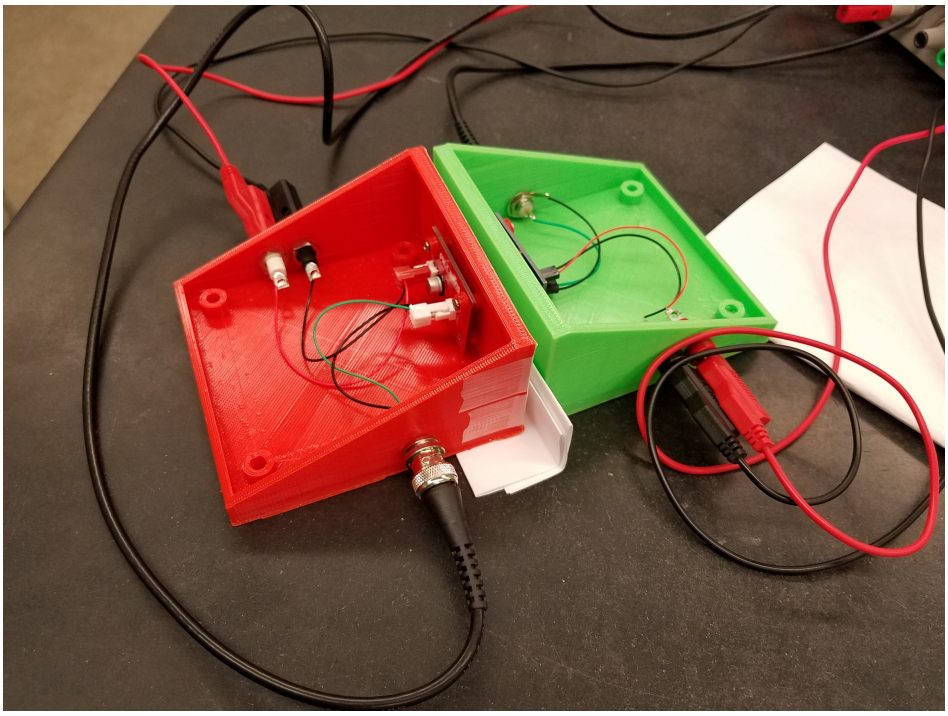


Figure 13.4: Initially, point the laser directly into the receiver.

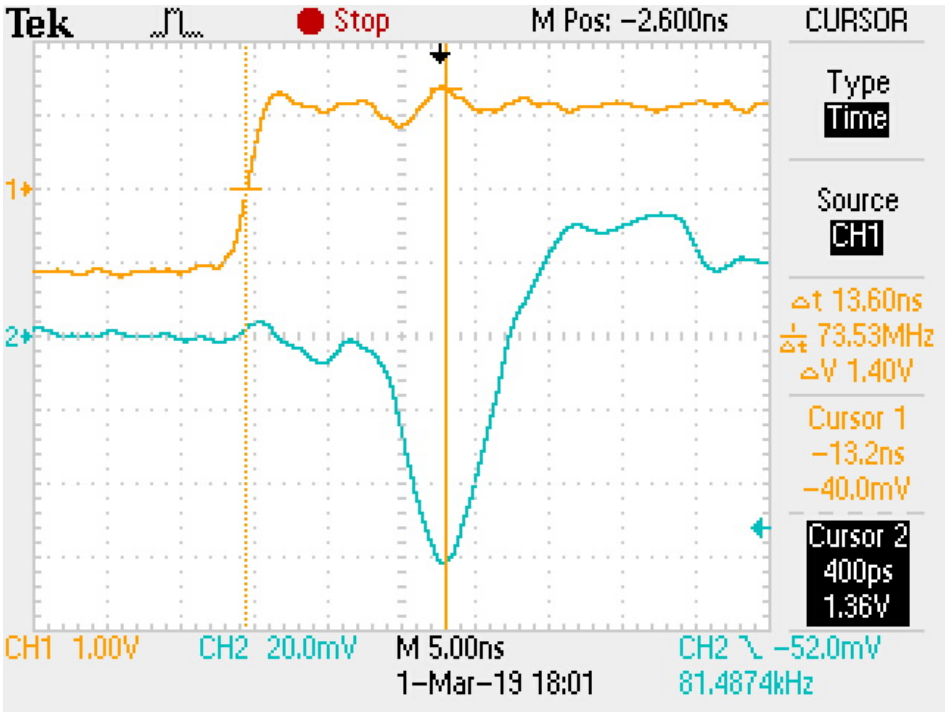


Figure 13.5: Measure the time interval from the receiver reference signal (yellow, offset by two divisions) crosses zero to the lowest point in the signal pulse (blue).

largest pulses (by moving the trigger threshold as far toward the bottom of the screen as possible). Because of timing jitter, you will acquire a single waveform to make each time measurement, using the scopes run/stop or the single button.

Logbook Entry 13.1. Measure the time offset (Δt at zero distance). Each student should make at least three measurements, from three different waveforms. Despite the jitter, you should see that the time interval measurements are fairly stable, varying by about 0.2 nanoseconds, the cursor resolution at the five nanosecond scale.

13.4 Speed of Light Measurement

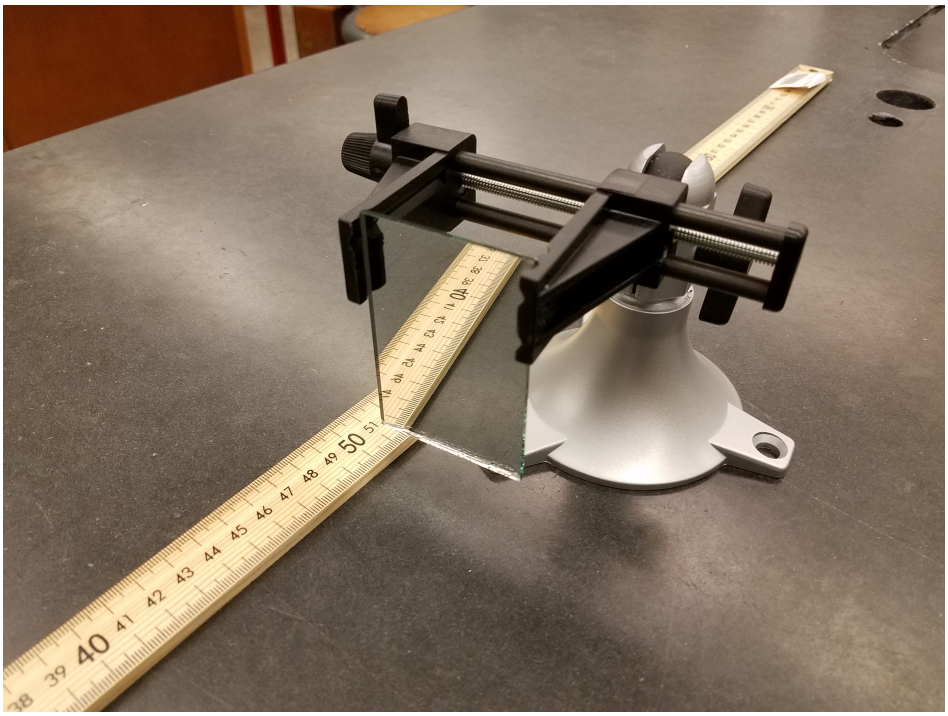


Figure 13.6: Mirror held in a swivel-mount vise

Install a mirror in your swivel mount vise as shown in Fig. 13.6 Move the transmitter to point the beam down the long access of your lab bench. Place the mirror approximately 25 cm away from the transmitter as measured with a meter stick. Using a piece of white paper to track the beam spot, adjust the transmitter until the beam is pointed at the mirror. Place the receiver next to the transmitter and pointed at the mirror. Adjust the mirror until the beam is directed back to the receiver.

Once you have the beam pointed toward the receiver, orient your scope display so that you can see it clearly from the mirror. Set the trigger to just below the noise level, and adjust the mirror until the beam points into the receiver and a clear signal appears on your scope. Tighten the swivel mount to hold the mirror in place. You may find that the mirror moves slightly after you remove your hands and the signal is lost. If this is happening, try gradually tightening and adjusting the mirror, or lightly tapping it while the swivel-mount is tight. Continue adjusting until you have a clear signal.

Logbook Entry 13.2. Each lab partner should make their own measurement of the time difference between the transmitter and receiver signals, and record all measurements in your logbooks. Then each lab partner should make a measurement of the total beam distance to the nearest 0.5 cm. Record all measurements in your logbooks. Estimate the uncertainty on your measurement. The resolution of the cursor time measurement at the 5 ns scale is 0.2 ns. The meter stick has a resolution of about 0.5 cm. If your measurements are consistent with one another, you can use these resolutions as your uncertainties. From these measurements and the timing offset measured previously, calculate the speed of light and an uncertainty.

Logbook Entry 13.3. Repeat the procedure for previous measurement (at about 25 cm) for a starting position of the mirror at 50 cm, 75 cm, and 100 cm.

13.5 Analysis

Jupyter Notebook 13.1. Plot the data with the x-values populated by the quantity with smaller uncertainty and y-values populated by the quantity with the larger uncertainty. Include x and y-uncertainties in the plot. Perform a straight line fit using `curve_fit` function. Include y-uncertainties in the fit and be sure to set `absolute_sigma=True`. From the fit values calculate the speed of light together with its uncertainty.

A major source of systematic uncertainty in this measurement comes from the timing measurement. The transmitter reference has a nice sharp edge, but the signal pulse is distorted by amplification. As the distance traveled by the light pulse increases, the signal becomes smaller, and the signal shape changes, which changes the measured time interval. This introduces a non-linearity into the relationship between the distance and your measured time.

Jupyter Notebook 13.2. To estimate the size of this effect, repeat the analysis but with `absolute_sigma=False`. This setting instructs the curve_fit function to adjust the uncertainties until they are consistent with the linear relationship assumed in the fit. You can then interpret the parameter uncertainty as including both the statistical uncertainty and the systematic uncertainty due to this non-linearity.

Chapter 14

Muon Lifetime

14.1 Introduction

In this lab, you will analyze data to cacluate the muon lifetime. This lab has only Jupyter notebook entries.

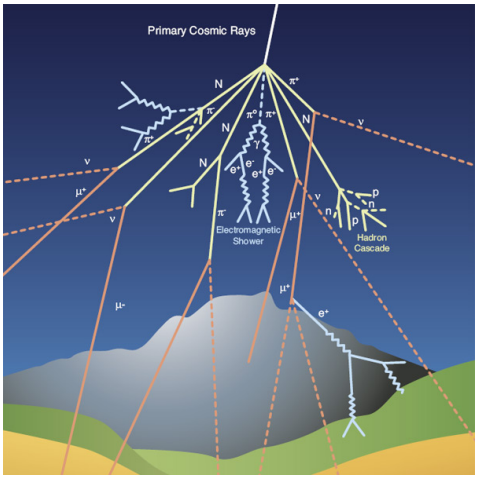


Figure 14.1: Cosmic ray shower induced by a primary cosmic ray (typically protons) striking an atom in the upper atmosphere.

The muon is a fundamental particle in the Standard Model of particle physics. It is essentially a heavier version of the electron. Like the electron, the muon has a corresponding anti-particle, the anti-muon (μ^+). Muons are readily available for study because they are produced as a result of showers that are induced by incident cosmic rays that constantly bombard the earth. Typical primary cosmic rays are protons, and there collisions with nuclei in the upper atmosphere produce mainly protons, neutrons, and pions. The subsequent decays of these particles produce electrons, neutrinos, and the muons we will be studying. The flux of muons at sea-level is about 1 per cm^2 per minute, and this population has a mean kinetic energy of about 4 GeV. Such muons are highly penetrating: they pass quite readily through buildings.

The muon decays via the weak interaction, its most probable decays being:

$$\mu^- \rightarrow e^- + \bar{\nu}_e \nu_\mu,$$
$$\mu^+ \rightarrow e^+ + \bar{\nu}_\mu \nu_e.$$

As fundamental particles, muons are indistinguishable from one another, and therefore the decay rate for a population of N muons must be simply proportional to N:

$$\frac{dN}{dt} = -\frac{N}{\tau}$$

The solution to this differential equation is:

$$N(t) = N(0)\exp(-t/\tau).$$

The muon lifetime has the value

$$\tau_\mu = 2.1969811(22)\mu s$$

in vacuum as reported by the Particle Data Group (PDG) with the uncertainty in parenthesis. Interactions with the scintillator material in this experiment lead to a slightly different expectation for the lifetime as discussed below.

Muons are produced at a typical height 15 km above sea-level, and so, in the earth frame, their transit time from the upper atmosphere to our lab is therefore about 50 μs or about 20 lifetimes. The fact that we see an appreciable number of them at sea level, given an upper limit on their production rate in the atmosphere, is clear evidence for the time dilation effects of special relativity.

14.2 Experimental Setup

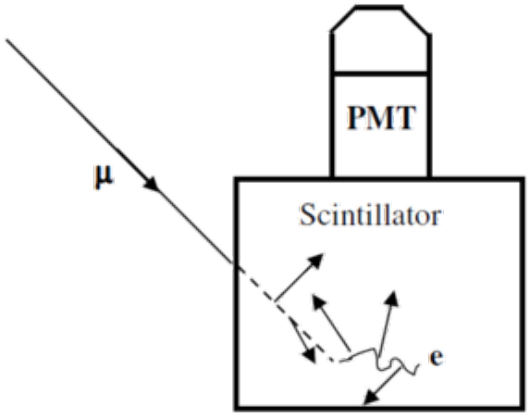


Figure 14.2: Experimental setup.

The active component of the particle detector used in this experiment is a polyvinyltoluene-based plastic scintillator in the shape of a cylinder with a 15 cm diameter and a 12.5 cm height. All materials absorb energy due to the passage of ionizing radiation. Scintillators are materials which re-emit a fraction of this energy as visible light, typically in the blue to near UV range. The light yield is relatively low, so a sensitive photomultiplier tube is used to amplify a modest number of photons into a large easily measurable voltage.

Muons are therefore constantly passing through the scintillator, depositing energy, and causing observable PMT pulses which are recorded by the data acquisition system (DAQ). Occasionally, however, a relatively low-energy muon enters the scintillator and deposits all of it's kinetic energy, coming to rest. As an unstable particle, it eventually decays into a highly-energetic electron and

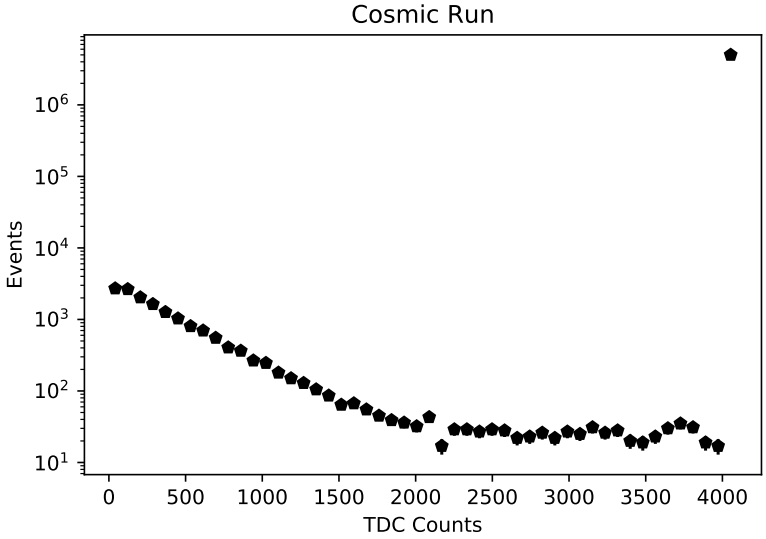


Figure 14.3: Three days of cosmic ray data, plotted as the distribution in digitized time measurement (TDC counts).

neutrinos. The electron deposits additional energy in the scintillator which is observed as a second pulse. The time interval between two consecutive pulses is digitized using a time-to-digital (TDC) converter, which converts analog time interval into a digital number. In this case, we use a 12-bit TDC, so the measured digital values are integers in the range from 0 to $2^{12} - 1 = 4095$. To interpret these raw TDC counts as a time value in microseconds, you will need to calibrate the TDC as described below.

Three days of data has already been collected for you, and is plotted in Fig. 14.3 . This data is available on the course web site in the file `cosmics.npy` available You load this data using the numpy `load` command, taking care to handle the directory path correctly. As can be seen from the plot, most of the recorded events have the maximum possible value 4095. This is how the DAQ records a single pulse, with no secondary pulse in the timing window of the TDC. This is what happens most of the time.

The remaining events, with a measured time below the maximum value, are from events were two pulses were detected within the timing window of the TDC. This time interval, the decay time of the muons in this population, will have a time-dependence given by:

$$-\frac{dN}{dt} = \frac{N(0)}{\tau}\exp(-t/\tau)$$

which you will use to extract the muon life time. This is possible because the muon decay rate (λ) and it's lifetime τ are simply recipocrals:

$$\lambda \equiv \frac{-\frac{dN}{dT}}{N(t)} = \frac{1}{\tau}.$$

This exponential decay feature is clearly visible in the raw data as the downward sloping line on the right side of this semilog plot.

There is an additional source of background for two pulse events, which arises from the possibility for two muons to arrive independently within the time window of the TDC. Sine the arrival time

of the two muons is independent, this contribution is flat in time, and is clearly visible in the raw data on the left-hand side, where the exponential contribution becomes small.

14.3 Calibration

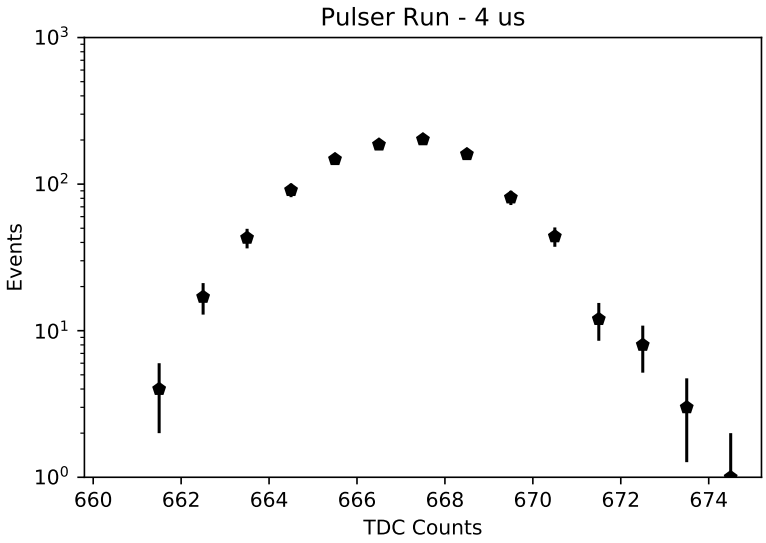


Figure 14.4: Calibration run with pulses spaced 4 μs apart.

To interpret the output of the TDC (digital TDC counts as integers in the range from 0 to 4095) as physical times in units of microseconds, a calibration is needed. We are using a good TDC which we can assume has a linear response, meaning that the time interval and TDC counts are related as:

$$TDC = a \cdot \Delta t + b$$

To determine the calibration parameters a and b we use an LED pulser feature built into the detector. The LED pulser feature flashes light into the PMT in a controlled manner, so that two pulses a distance δt in time apart are sent repeatedly to the TDC. An example of 1000 pulser events with $\Delta t = 4\mu s$ is shown in Fig. 14.4. A total of four pulser runs of 1000 events each with $\Delta t = 1, 2, 4, 8$ have already been collected. The data from these four calibration runs available on the course website with names like `pulser_4.npy` for the $\Delta t = 4$ μs run.

Jupyter Notebook 14.1. or each pulser run, compute the mean value of the recorded data and an uncertainty on this mean value. Plot these mean values, and their uncertainties, versus Δt. Fit the linear response to obtain the parameters a and b. Plot the best fit linear function alongside the data.

14.4 Muon Lifetime Analysis

The calibration constants a and b are used to convert the TDC counts of the recorded comsic ray data into physical time intervals.

Jupyter Notebook 14.2. Using your calibration contants, convert your comsic data TCD counts into time intervals, and plot the ditribution as a histogram. You should omit all events with RAW TDC count of 4095, as these are single pulse events. Fit the distribution to an exponential function plus a constant value for the flat (combinatoric) background. Plot your fit function against the recorded data. Report the statistical uncertainty on the fitted value for the muon lifetime. In addition, there are several systematic uncertainties associated with the experiment. The largest is the effect of material interactions on the lifetime of the muon, which is about 5%. Other systematic effects are smaller, and can be neglected.