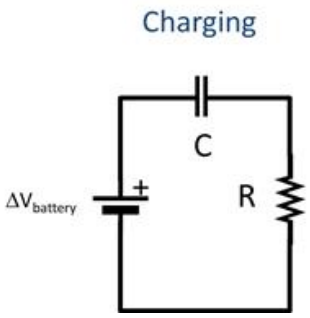
Chapter 6

Resistors and Capacitors

In this lab, we will not build a new instrument. We will use an instrument we built in a previous lab (or at least only a new version of that instrument adjusted for today's resistance values). You will notice a pattern in what we do from now on in PH250. We will build an instrument and then test a model with that instrument. The instrument must be designed so that it can take the data needed to test the model. In today's lab, we will test the model of how capacitors work in a circuit. If your PH220 class is moving along nicely, this model will be familiar.

6.1 The Model to Test



Let's start by thinking of hooking up a capacitor and a resistor in series with a battery. The capacitor will become charged. The voltage across the capacitor as a function of time will be given by

$$\Delta V_C(t) = \Delta V_{\max}\left(1 - e^{-\frac{t}{\tau}}\right) \qquad \text{charging}$$

where

$$\tau = RC \tag{6.1}$$

is the product of the resistance, R, and the capacitance, C. The current in the circuit as a function of time will be given by

$$I(t) = I_{\max} e^{-\frac{t}{\tau}} \qquad \text{charging}$$

while we charge up the capacitor. The quantity

$$\tau = RC$$

is called the time constant.

We should review what a time constant is. Think of a particular case, say,

$$\begin{aligned} \Delta V_{battery} &= 1.5\text{V} \\ R &= 2\Omega \\ C &= 10\text{F} \end{aligned}$$

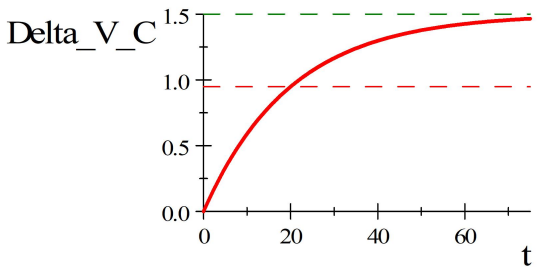
then

$$V_C(t) = (1.5\text{V})\left(1 - e^{-\frac{t}{(2\Omega)(10\text{F})}}\right)$$

and

$$\begin{aligned} \tau &= (2\Omega)(10\text{F}) \\ &= 20.0\text{s} \end{aligned}$$

We can plot thisNotice, that by about $t = 70$s we essentially have $\Delta V_C =$



$\Delta V_{battery}$. But up to that point, the voltage across the capacitor changes in a very non-linear way. The part of the equation that looks like

$$\left(1 - e^{-\frac{t}{RC}}\right)$$

is interesting. What is e^0?

$$e^0 = 1$$

So at $t = 0$ we do have $\Delta V_C = 0$ on the capacitor because

$$\left(1 - e^{-\frac{t}{RC}}\right) = (1 - 1)$$

For any positive time, $e^{-\frac{t}{RC}}$ will be less than 1. For large positive times $\frac{t}{RC}$ gets to be a big number. So $e^{-\frac{t}{RC}}$ gets very small. Then $\left(1 - e^{-\frac{t}{RC}}\right)$ gets very close to 1. That means that

$$\lim_{t\to\infty} \Delta V_C = \lim_{t\to\infty} \Delta V_{battery}\left(1 - e^{-\frac{t}{RC}}\right) = \Delta V_{battery}\left(1\right) = \Delta V_{battery}$$

just as we saw in the graph and as we know it must.

But what if $t = \tau = RC$? Then

$$\begin{aligned} \Delta V_C &= \Delta V_{battery}\left(1 - e^{-\frac{RC}{RC}}\right) = \\ &= \Delta V_{battery}\left(1 - e^{-1}\right) \\ &= 0.632\,12\Delta V_{battery} \\ &\approx 63\%\Delta V_{battery} \end{aligned}$$

The time τ is the time it takes for the capacitor to be 63% charged!

The quantity τ is called the *time constant* because it tells us something about how long it takes for ΔV_c to go from 0 to get to $\Delta V_{battery}$. The "t-looking-thing" is a Greek letter "t." It is pronounced "tau." This quantity will be useful in planning your experiment.

Notice what we have done. We have used our model to form an equation, and we have used part of that equation to understand how much time it will take to perform a test (experiment) of the model. This is typical, we have to get an idea of how to make the measurement from the model we are testing.

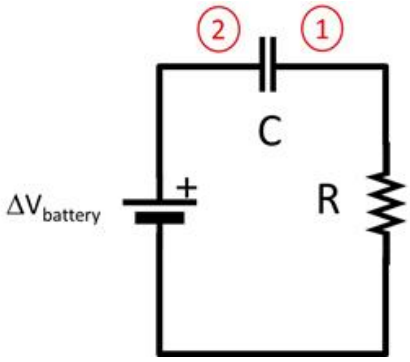
But! you say, I don't really remember where all of these equations came from. Or maybe your PH220 class hasn't gotten to allowing current to flow yet so you have not done this. If any of this is mysterious, please read the section of our PH220 book that covers RC circuits. But if it is vaguely familiar or seems to make sense, really we can test our model of how capacitors work just knowing a little about capacitors and the equations that came from the model.

6.2 The Instrument

To test our capacitor model we need to measure the voltage across the capacitor as a function of time. We could also measure the current in the circuit as a function of time. One of these is sufficient to test the model. I am going to describe measuring the voltage across the capacitor as a function of time. But you know from a previous lab how we might add current as a function of time.

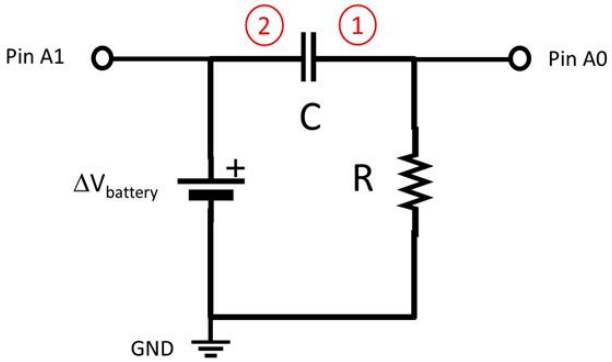
We need a device that measures voltage and how it changes as a function of time. But that is just what our Arduino's do! We already know how to build this device. Suppose we can live with a 0V to +5V range of $\Delta V_{battery}$. Then even our simple voltmeter will work. Since it is a function of time that we are testing, we need to output both voltage and time from our Arduino. We can't guarantee that either of our capacitor leads will be at ground, so we will have to be careful in wiring this voltmeter to give ΔV_C.

Remember that ΔV_C is the difference between two voltage measurements. so



$$\Delta V_C = V_2 - V_1$$

neither of which will be ground, so we really have to measure both with our Arduino. We also need a ground connection. The wiring diagram might look like this:and our sketch will be a little like the one from our last lab.



Download here

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
// very simple voltmeter
// will measure 0 to 5V only!
// Voltages outside 0 to 5V will destroy your Arduino!!!
// Delta_V_shunt must therefore be much less than  5V
////////////////////////////////////////////////////////////
// we want to have voltage vs time,
// so make a place to store a time value
  unsigned long time;
 // make some integer variables that identify the
```

```
 // analog input pins we will use:
   int AI0 = 0;
   int AI1 = 1;
 // you also need a place to put the analog to
 // digital converter values
 // from the Arduino
   int ADC0 = 0;
   int ADC1 = 0;
// Remember we will have to convert from Analog to
// digital converter(ADC) units to volts.
// We need our delta_V_min just like we did in lab 3
   float delta_v_min=0.0049;   // volts per A2D unit
// We need a place to put the calculated voltage
float voltage = 0.0;

//////////////////////////////////////////////////////////
 void setup() {
  // put your setup code here, to run once:
  //Initiate Serial Communication
  Serial.begin(9600);    //9600 baud rate
}

//////////////////////////////////////////////////////////
 void loop() {
  // Read in the voltages in A2D units form the
  // serial port
  ADC0 = analogRead(AI0);
  ADC1 = analogRead(AI1);

  // Convert the voltage across the
  // test resistor to voltage
  //   units using delta_v_min
  voltage = (ADC1-ADC0) * delta_v_min;

  // output the time since we started and the voltage
  Serial.print("␣time␣in␣sec,␣");
  // this next line uses millis() which gives time in
  // ms since we started
  time = millis();
  // convert to seconds and print.
  Serial.print(time/1000.0, 6);
  Serial.print(",␣voltage␣across␣C,␣");
  Serial.println(voltage,6);
}
//////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
```

This is just a voltmeter, but one with two A2D pins and a ground. This sketch also gives us time using the millis() function. This function gives the number of milliseconds since our experiment began. We can use our python code from a previous lab to save the data into a file. I modified the previous code just a bit, so here is an updated version.

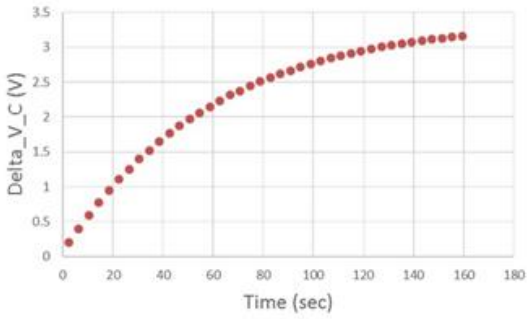
Download here

```
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
# Python Code to read a stream of data from the serial port
#   and save it to a file
#-------------------------------------------------------------------------
#   The idea is to read a series of voltages from an Arduino
#   connected to the serial port where the Ardunio is being
#   used as the Analog to Digital converter. Both the voltage
#   and the time the voltage was taken are sent to the serial port.
#
# We will use two libraries, serial and time
#   The serial library is used to read the serial port
#   The time library is used to space out our data collection by
#   adding a delay in between data points. The amount of time
#   to wait in between data points is called "timeBetween."
#
# We may have to install the serial library. If you have the
#   Anaconda Python for Windows, you can open an Anaconda
#   window and use the command 'conda install pyserial'
#   This must be done before the code can run.
#
# Debugging issues:  The Anaconda Python distribution tends to
#   hang on to the serial port even if the program does not run.
#   If this happens, try sending the python command ser.close()
#   at the command prompt. If this doesn't work, You may have to
#   restart Python.
#   In windows, closing (after saving) the IDE and reopening it
#   might be enough.
#-------------------------------------------------------------------------
# import libraries
import serial
import time


# define variables for the delay time we wait between data points
timeBetween=4 #seconds

# define the number of data points to collect
N=40

#the next line opens a file and creates a pointer or handle for that
                                        file
#  to use as a reference. You have to use double slashes in the path.
#  The pointer, "dataFile" takes the place of all the path and file
#  name so it is easier to use in the code below
dataFile=open("C:\\Users\\rtlines\\Documents\\RCdata.csv","w")

#the next line opens the serial port for communication
```

```
ser=serial.Serial('COM3', baudrate = 9600, timeout=1)
#the next line clears out the serial port so we get clean data.
ser.flushOutput()

#there will be a delay before the first data point comes from the
#  serial port, warn the user so they don't worry.
print('getting started...')

# set our index to zero
i=0


# Now for N points, collect data and write it to a file
while (i<N):    #Begin data collection loop
    #We will take data every "timeBetween" seconds. We need to know
    #   when we start waiting so we can tell if it is time to collect
    #   data yet. Use the time.time() to get the current time in
                                        seconds
    #   since Jan 1,1970. Yes that is a weird way to measure time, but
    #   computers do it this way.
    waitStart=time.time()

    #Data comes to the serial port fast. We will continually collect
    #  the data as fast as it comes, but only save it every timeBetween
    #  seconds. The next while loop keeps us reading in data, but only
    #  when the current time - waitStart >= timeBetweem will we use
    #  the data.
    while (time.time()-waitStart<timeBetween): #Begin Data read loop
         # Get data from the serial port
         # it should have a time and a voltage
         arduinoData=ser.readline().decode('ascii')
         # end of the Data read loop

    # the next line just prints the voltage point on the console so the
                                         user
    # feels like something is happening.
    print(arduinoData)
    # This next line writes combines the time since we started and the
                                        Arduino
    # value from the serial port into one string
    writeString=str(arduinoData) #+ " \n"
    # The file write command adds a new line character The Arduino
                                        added a
    #   new line character, so we have double spacing. Let's remove one
                                         of
    #   them before we write to the file.
    writeString = writeString.replace("\n", "")
    # The next line writes our time plus Arduino value to the file.
    dataFile.write(writeString)
    # and finely we increment the loop counter
    i=i+1      # end Data collection loop

# Print out a message saying we are done
print("done with data collection, closing the file and the serial port"
                                      )
# Close the file
dataFile.close()
```

```
# Close the serial port
ser.close()
#--------------------------------------------------------------------
#--------------------------------------------------------------------
```

The resulting data could be plotted in Excel. It might look something like this.I plotted this in Excel, and that is great. But Excel doesn't have the right



function built into it for a curve fit. So we will use a new analysis program named LoggerPro. It is not hard to use, and you can copy your data from a file, or from Excel into LoggerPro easily. The next section shows how to make this work. If you are a fantastic Python programmer, you could use Python for this part. If you are a die-hard Excel user, we can show you how to use Excel and get the same result. But LoggerPro will make this very easy.

6.2.1 LoggerPro Curve Fitting.

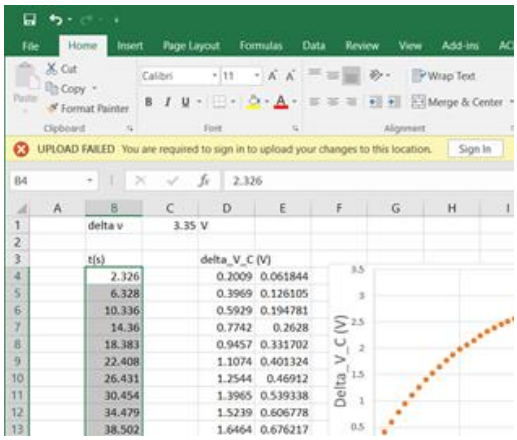
Let's start by noting that you can download LoggerPro to your own computer if you would like. You should have received a notification about this on I-Learn. If you wish to install LoggerPro, follow the announcement instructions.

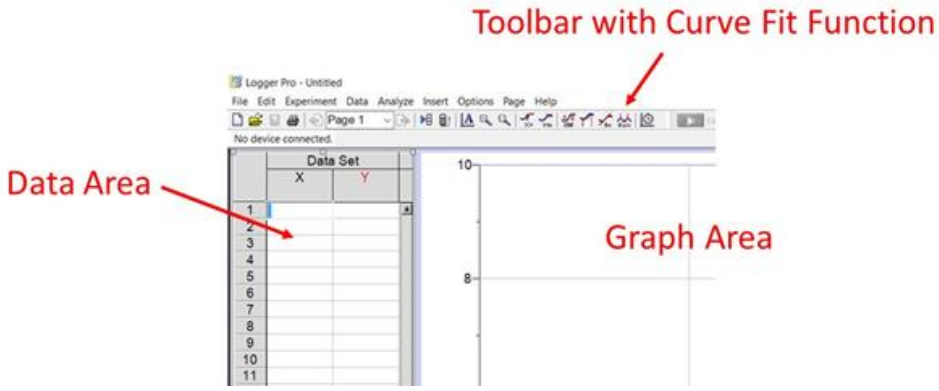
Once you have LoggerPro on your computer or on one of our lab computers, we will use it for fitting a curve to our data just like we did last time in Excel. Suppose you have already imported your data in Excel. It might look like this:

Now highlight a—— column (I highlighted the time column) and select copy to copy the data to the clipboard. Then open up LoggerPro. You will see a data area, a graph area, and the toolbar.We want to past the data into a column in LoggerPro. If you also selected the time data, paste it into the x-column in Logger Pro.

Do the same for the voltage data. Paste it into the y-column. Once you do this, the data will automatically be graphed. We now want to fit a curve to this data. The curve fit function can be accessed from the tool bar. It looks like this: Click on the icon and a new dialog will appear.

It is tempting to try just any function to see if it fits. But the goal is not to have a great fit. The goal is to see if our data fit the equation from our capacitor





model.

$$\Delta V_C\left(t\right) = \Delta V_{\max}\left(1 - e^{-\frac{t}{\tau}}\right)$$

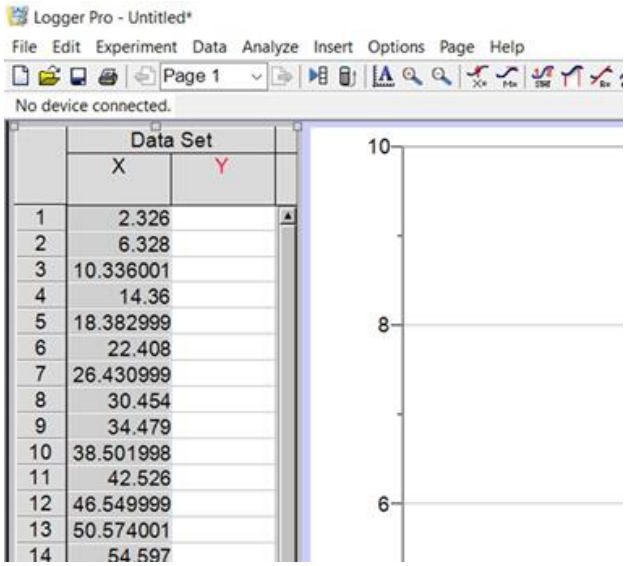
so we need to find this particular function. This is why I didn't suggest using Excel. Excel does not have the equation we need in it's list.

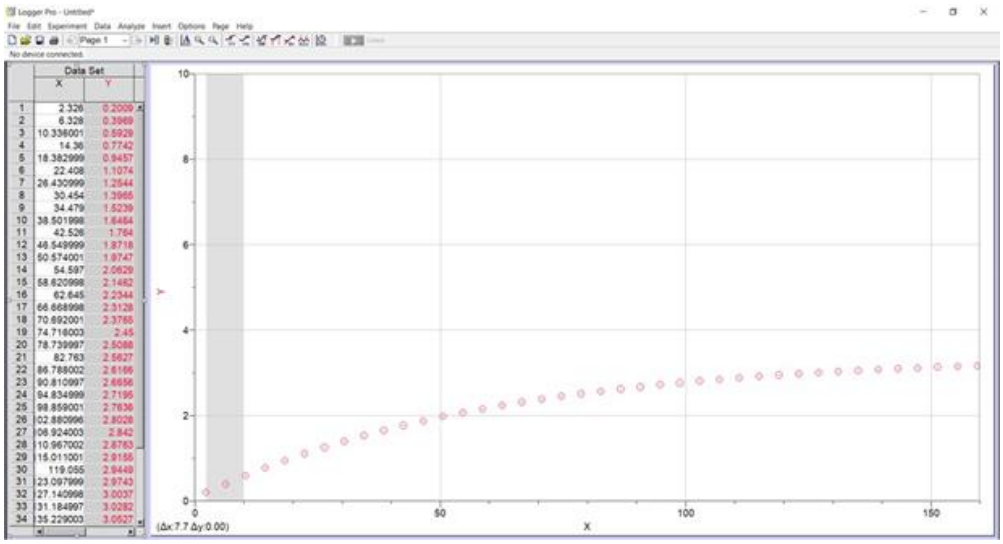
Of course we won't find a match with our exact notation. The one we want is given as

$$y = A * (1 - \exp(-C * x)) + B$$

and now we have to match our variables with theirs. Let's compare the equations.

$$\begin{aligned} y &= A * (1 - \exp(-C * x)) + B \\ \Delta V_C\left(t\right) &= \Delta V_{\max}\left(1 - e^{-\frac{t}{\tau}}\right) + 0 \end{aligned}$$



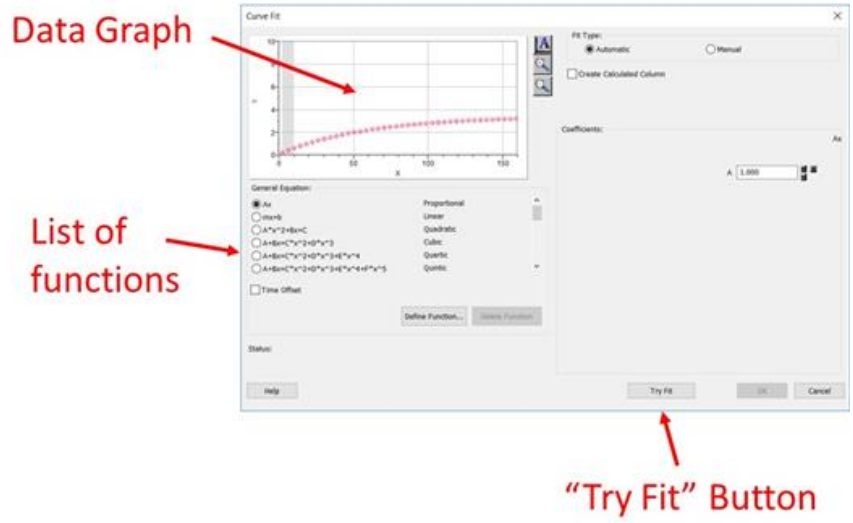


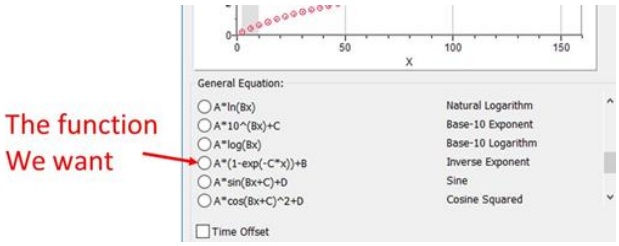
We can see that

Ours	Theirs
$\Delta V_{\max}$	A
0	B
τ	$\frac{1}{\mathrm{c}}$

We can see this

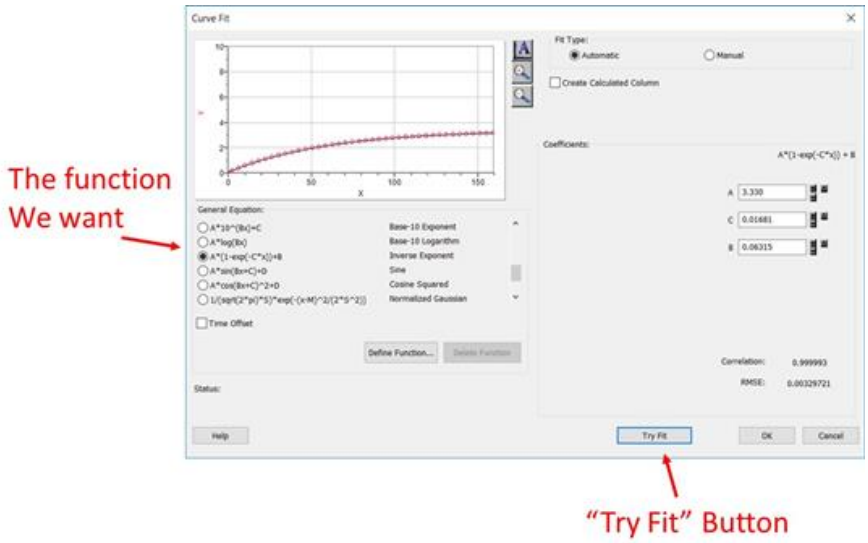
If the fit looks good, choose the "OK" button. You get the graph back with the curve fit and a new little box. The curve fit looks nice and that is

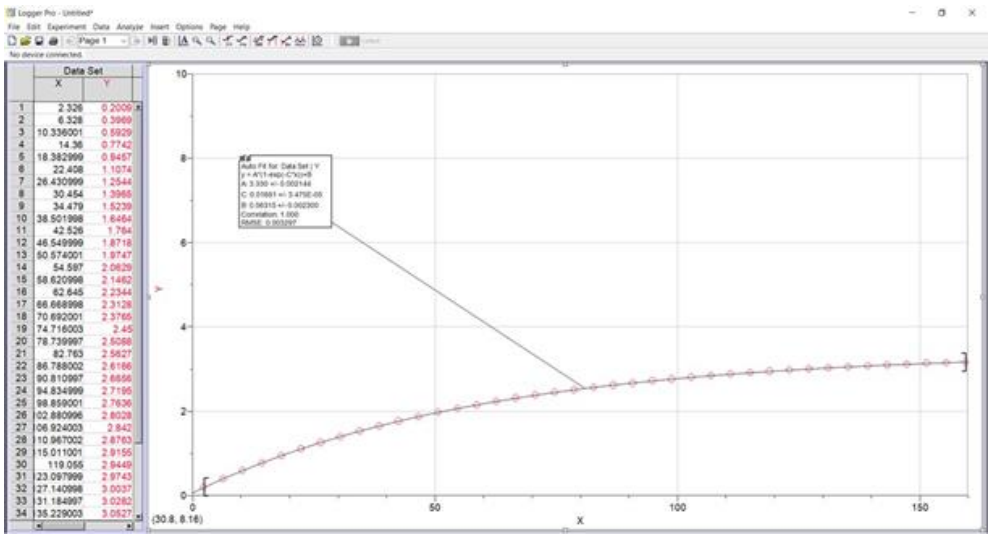




comforting. For my data, it looks like our capacitor model might be correct, but we can't be sure until we add in error bars. Before we do that, let's look at the new little box. The box has our fit equation that we chose and it has values for the fit parameters and their uncertainties. We will need those later!

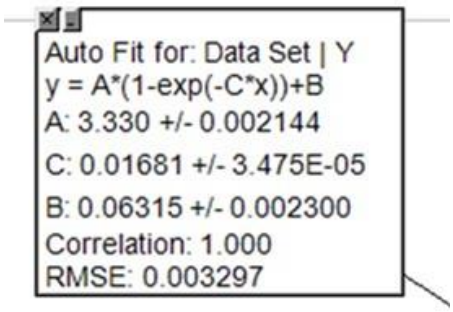
Let's add on the error bars now. Right click on the graph if you have a PC or do the Mac equivalent if you have a Mac. A new dialog appears and in this case choose "Graph Options." On the "Graph Options" dialog make sure both x and y-error bars are checked.

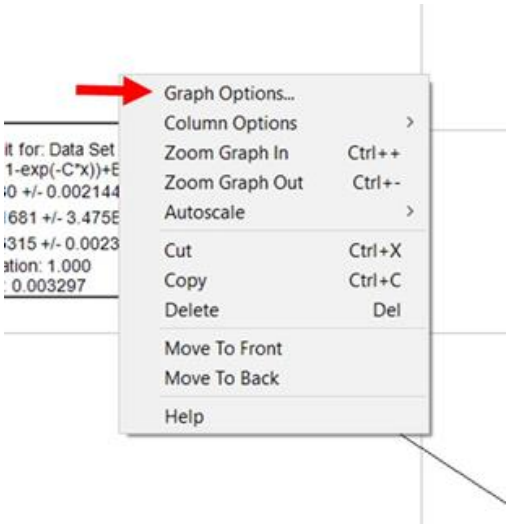




Chose "Done" and right click on the graph again. This time choose "Column Options." and choose the y-data set. Another dialog appears with two tabs. Choose the "Options" tab.

You will see a place to choose how error bars are calculated. If you used the simple voltmeter sketch as your basis, you know the quantization error is about 4.9mV. That will be true for every voltage measurement so we can input this as a constant value. If you used a voltage divider, you will have to use your calculated error value here. When you have your error value in place, choose "Done."





The voltage error was so small that it is hard to see the error bars! That is fine. What this tells us is that our fit line must go right through the center of each data point. But it does. So we are really doing fine. The data supports the model.

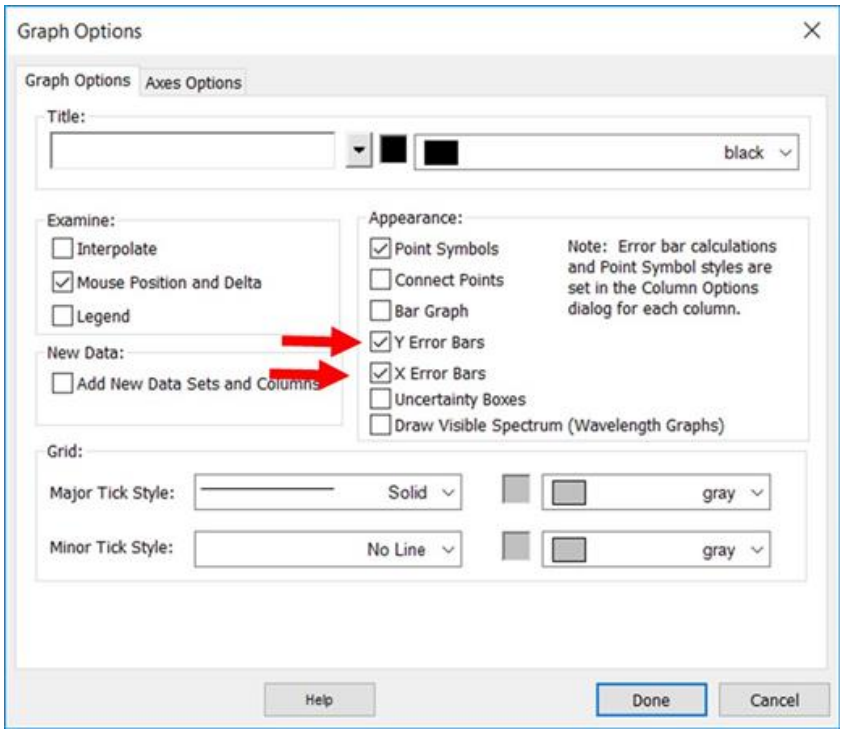
But now let's go back to our little box of curve fit parameters. We identified

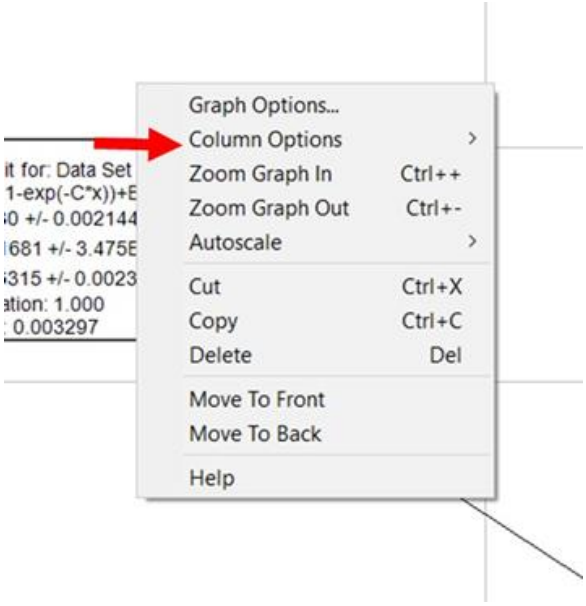
$$\tau = \frac{1}{\mathtt{C}}$$

and for my data I have

$$\mathtt{C} = 0.01681 \pm 3.475 \times 10^{-5}$$

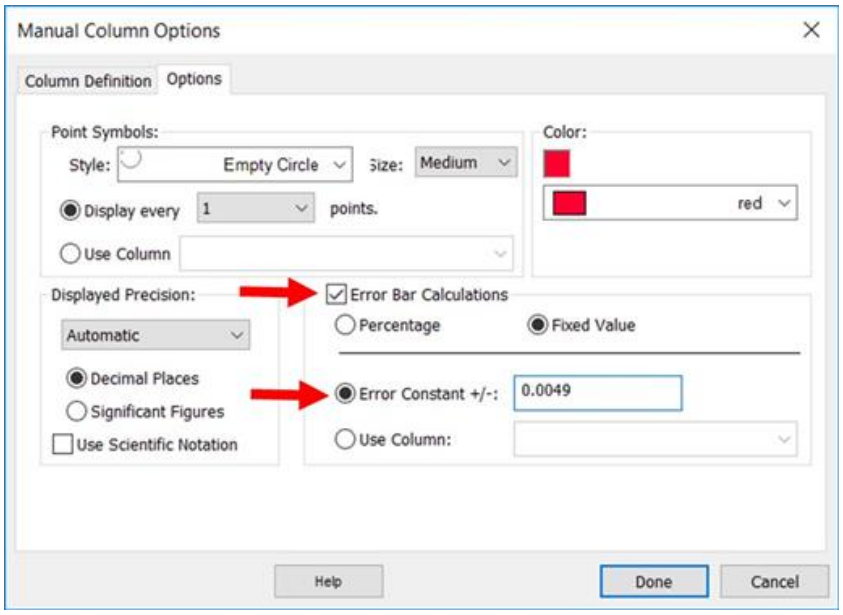
We need units, and looking at the equation we know τ has units of seconds, so

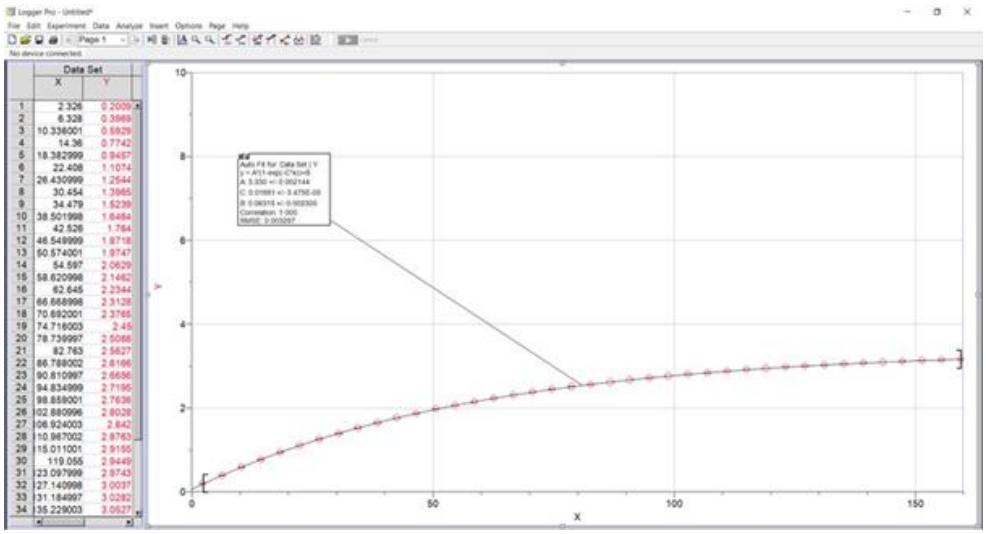




`C` must have units of inverse seconds.

$$\texttt{C} = \left(0.01681 \pm 3.475 \times 10^{-5}\right) \frac{1}{\text{s}}$$





so we can find a value for τ. For my data, I have

$$\begin{aligned} \tau_{measured} &= \frac{1}{0.01681\frac{1}{\text{s}}} \\ &= 59.488\text{s} \end{aligned}$$

Note that we will have to calculate the uncertainty in τ. I will leave that for an exercise. But I can compare this $\tau_{measured}$ to the $\tau = RC$ value I started

with. If they are within each other's error range, this is a powerful confirmation of our capacitor model.

6.3 Lab Assignment

We have two equations for charge as a function of time for a RC circuit. They are

$$\Delta V_C(t) = \Delta V_{\max}\left(1 - e^{-\frac{t}{\tau}}\right) \qquad \text{charging}$$

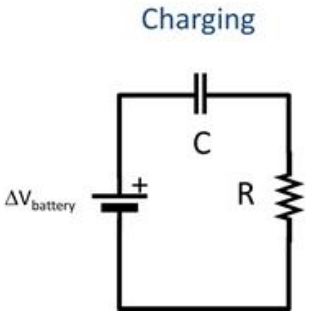
$$I(t) = I_{\max} e^{-\frac{t}{\tau}} \qquad \text{charging}$$

where

$$\tau = RC \tag{6.2}$$

is the time constant.

1. Using a capacitor with a capacitance of about 20μF and a resistor of about 1MΩ, create a circuit as shown. This will be our system that we will use to test our capacitor model.You can use one of our power supplies, but be



   careful to either stay in the 0V to +5V range, or to use a voltage divider to achieve this range at the Arduino input. Follow good lab notebook procedures by recording the model you are testing and your test setup in your lab notebook. Just a note, we will be using directional capacitors. You haven't studied directional capacitors in class. For today's lab, the only difference is that these capacitors only work one direction. This is a little like our diodes. If the circuit doesn't work, try turning your capacitor around.

2. Build your instrument. and write the sketch and Python collection codes. Test every part of the instrument before you start collecting capacitor data. Don't forget to find your uncertainties. Follow good lab notebook procedures by recording your instrument design in your lab notebook.

3. Now get ready to collect data for a capacitor charge. Work with a lab partner from your group to achieve the data collection. Compare your data among your group to make sure things went well. Follow good lab notebook procedures by recording your data or giving a location of the stored data in your lab notebook.

4. Take the data from your file and graph it. LoggerPro is fine for both graphing and the curve fit (next item). You should include this graph in your lab notebook (but might also include the curve fit described in the next item on the same graph).

5. Perform the curve fit. As in the last lab, having the proper curve fit the data is a validation of our model! So if the thoretical curve fits the data, it makes sense that something about the model is right. Include the graph of the curve fit and the data in your lab notebook as well as the fit equation and fit parameters (don't forget their uncertainties).

6. Find the time constant and compare to your predicted value. If these compare within their uncertainties, we have a further validation of the model. Record the time constants and their uncertainties in your lab notebook.

7. Draw a conclusion, is our capacitor model good?