Daniel Opdahl

Photoelectric effect experiment

Introduction

When light hits a metal, if the light as a certain frequency, electrons can be ejected from the metal. This is because, as Einstein hypothesized and then discovered, the energy in light is quantized and carried by quanta of energy called photons. When an incoming photon strikes an electron that is part of the metal plate, the photon transfers its energy to the electron. If the energy of the photon was great enough, the electron can gain enough energy to escape from the metal. The threshold energy for electrons to escape from the surface from a given material is called the work function of the material. If two plates are set up such that one has light shown on it, and the other, obscured from that light, is positioned in a direct line to the other metal plate, ejected electrons from the first plate will reach the second plate. If the two plates are connected electrically, a small current can be carried by the ejection of electrons from one plate to the other. If a voltage is applied between the two plates, it is possible to limit the amount of electrons that make up the current flowing, and even stop any ejected electron from reaching the other plate. The applied voltage at which this occurs is called the stopping voltage.

Setup

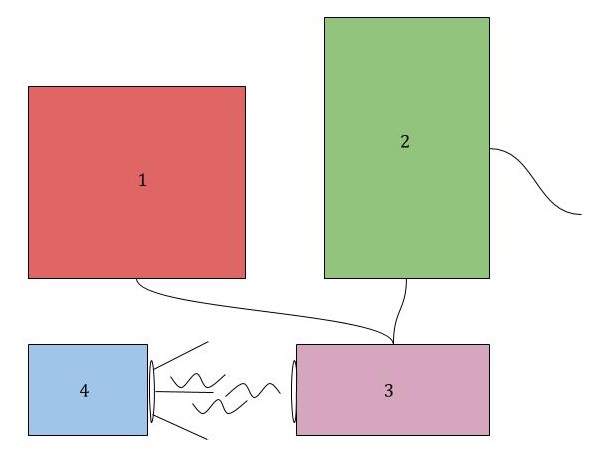


Figure 1: A depiction of the experimental setup. (1) Power supply controlling the voltage between the plates in (3). (2) A picoammeter reading the current created by ejected electrons, connected to a laptop (not shown) that stores the data. (3) The phototube that houses the two plates and has attached to it several filters for filtering incoming light. (4) A mercury lamp that provides the light to eject electrons from (3).

The setup for this experiment consisted of the elements shown in Figure 1. A power supply was used to control the potential between the two plates inside the phototube while a picoammeter was connected to the phototube to measure the current at each incremented voltage. These measurements were sent to a computer running a MATLAB script, found in Appendix A, that automated the data collection process. Additionally, a mercury lamp provided the light source for the ejection of electrons inside the phototube. The light from the mercury lamp passed through a selected filter before entering the phototube.

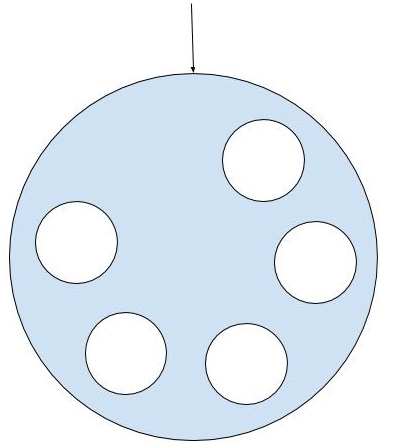


Figure 2: A close up of the lens disk that housed 5 different lenses used to filter light from the mercury lamp, only allowing a certain band of wavelength through.

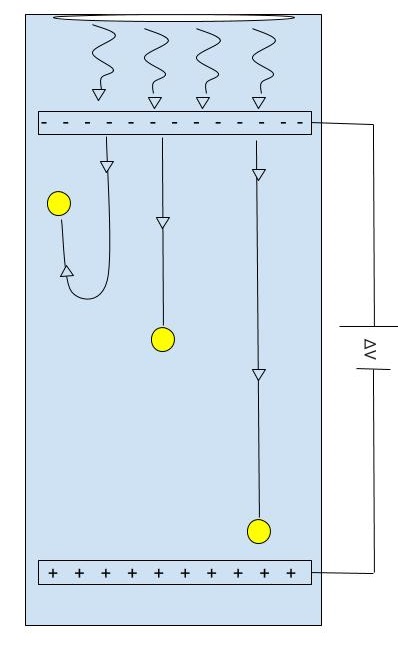


Figure 3: The phototube consists on an applied voltage across two metal plates. As incoming light of a certain frequency hits the top plate, photoelectrons are ejected towards the bottom plate. The phototube shown is currently in a “forward bias” state because the bottom plate is attracting the negatively charged electrons rather than repelling them.

Methods

The first thing we did when collecting data was ensure that the room we were collecting data in could be as dark as we could make it to prevent any stray light from interfering with our light source, the mercury lamp. The next thing we did was power up the mercury lamp. This requires a bit of time to warm up to get to a sustained brightness. Next, we rotated the filter disk to place a desired filter over the opening to the phototube, being careful not to touch the filter lest we dirty it. Next, we connected the power supply to the phototube such that the voltage will accelerate the photoelectrons towards the receiving plate. This will be referred to as “forward bias”. Using the MATLAB script found in Appendix A, we configured the power supply to increase the potential from 0 V to 30 V, incrementing the voltage by 0.1 V. At each step, the picoammeter would measure a value for the current at that voltage. A pause of one second was allowed between each increment and reading. After we had taken measurements from 0 V to 30 V in this way three times, we switched the leads on the power supply in order to reverse the voltage. This position will be referred to as “reverse bias”. Again, we used the MATLAB script found in Appendix A to configure the power supply and the picoammeter to take data automatically. However, this time the setup was configured to take data from 0 V to -2 V, with increments of 0.05V. We repeated taking data in the “reverse bias” position three times. After this, we were done with the first filter, and we repeated the process for “forward bias” and “reverse bias” three times each for each remaining filter.

Results and Interpretation

Each of the figures shown below are grouped by filter. There are six graphs per filter, and six graphs in every figure. The first graph in each figure shows all our gathered data averaged over all three runs with error bars representing the propagated uncertainty. The second and third graphs show only the forward and reverse biases, respectively. The fourth graph shows a fitted exponential curve to the forward bias. This curve was used to determine the saturation current by finding the curve’s horizontal asymptote. The fifth graph shows the first method we employed for finding the stopping potential. We averaged points on the horizontal plateau of the reverse bias graph along with their uncertainties to produce a threshold value. The data point that had a value and uncertainty closest to, but not intersecting this threshold value was said to be the stopping potential. The sixth graph shows our second method for finding the stopping potential. We fitted linear slopes to the horizontal plateau and the almost linear slope leading down to the knee and found the intersection of those slopes. The voltage value at that point was said to be the stopping potential. (Note to reader, if the graphs are not clearly visible, in Microsoft Word, hold control and scroll up on the mouse wheel. The document will zoom in so that the graphs are more visible.)

The saturation current was determined by fitting an exponential function to the forward bias data collected and finding the horizontal asymptote of that fitted curve. The collected data is shown in Table 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Filter (nm) | 577 | 546 | 436 | 405 | 365 |
| Saturation Current () | 0.642 | 2.159 | 4.30 | 2.86 | 5.53 |

The values for the saturation current at different wavelengths tell us the relative intensities of light at each frequency emitted by the mercury lamp. For example, the mercury lamp emits light of wavelength 355 nm more intensely than light of wavelength 577 nm. It follows that not all saturation currents are the same because the mercury lamp does not emit light of all wavelengths at equal intensities.

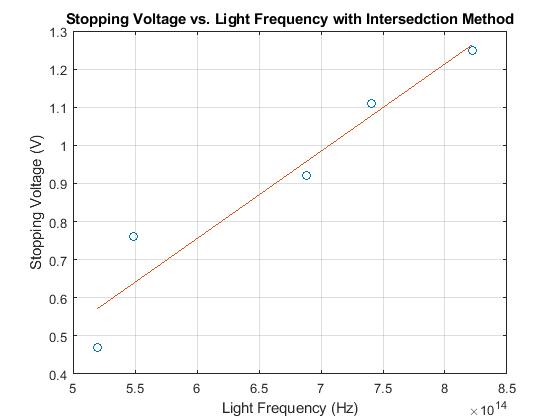
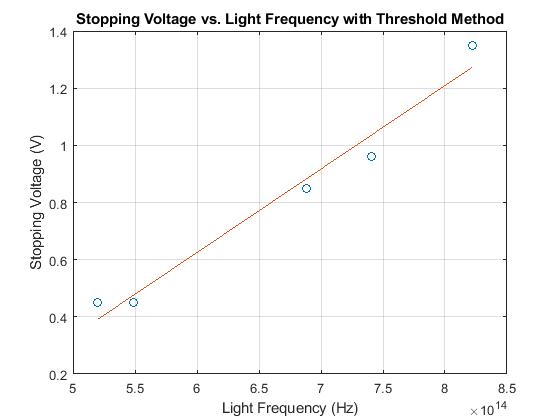
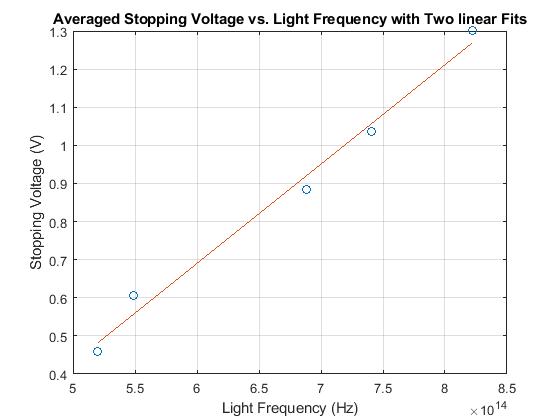


Figure 9: Stopping voltage and Frequency data shown for both methods and averaged values along with fitted linear slopes to all data.

The stopping potential of each wavelength of light was determined with two methods. The first method involved averaging points on the horizontal plateau of the reverse bias graph along with their uncertainties to produce a threshold value. The data points that had a value and uncertainty closest to, but not intersecting this threshold value was said to be the first value of the stopping potential. The second method for finding the stopping potential involved fitting linear slopes to the horizontal plateau and the almost linear slope leading down to the knee and finding the intersection of those slopes. The voltage value at that point was said to be the second value of the stopping potential. These values and their averages are shown for each frequency of light in Figure 9.

The averaged slope using the two methods yields our experimentally determined value of . We determined the slope of our averaged linear fit to be Vs. The expected value of is 3.4±0.3×10−15 Vs.

We were able to determine the value of the work function of the material used for the pates inside the phototube. Our experimentally determined value of the work function is 0.87 eV. Unfortunately, there is no metal with this value as its work function. The best estimate we can give for the metal is Rubidium, because it has the smallest work function of 2.26 eV, but even that value is over two times greater than our experimentally determined value. More analysis is needed.

Summary

The photoelectric experiment was performed by shining light of a particular wavelength on a plate, ejecting electrons from that plate. Inside the phototube, another plate was positioned on the line formed by the light source and the first plate. An electric potential was created between the two plates and was varied by incremental amounts for both forward and reverse bias. Electrons crossing the gap between the two plates for any given applied voltage would create a current and that current was measured and recorded. Using out gathered data for current for a particular voltage for each filter, we were able to determine the saturation current for each wavelength of light tested by fitting a exponential cure to our forward bias data and finding the horizontal asymptote of that curve. Additionally, we were able to determine the stopping potential for each wavelength tested in two ways. First, by finding the threshold value of the horizontal plateau of the data for reverse bias and finding the data point that was closest to that value. Second, by fitting lines to the relatively constant slopes present in the reverse bias graphs and finding the intersection of those two fitted lines. Averaging those values resulted in a measured value for the stopping potential of every wavelength tested, and by plotting the stopping potentials and frequencies (found from the wavelengths of light), we were able to estimate values for and the work function of the material used in the plates inside the phototube. Unfortunately, experimentally determined values of and the work function are inconsistent with known values and expected values.

Appendix A: MATLAB script used for data collection

%This script automates the data collection capabilities for the

%PASCO Photoelectric effect apparatus.

%Both the Keithley Picoammeter and the AMREL powersupply should be

%connected in series to the computer using the Agilent GPIB-to-USB cable, and the

%GPIB-to-GPIB cable.

%% Code exported from tmtool for Keithley Ammeter

% Find a VISA-GPIB object.

obj1 = instrfind('Type', 'visa-gpib', 'RsrcName', 'GPIB0::14::INSTR', 'Tag', '');

% Create the VISA-GPIB object if it does not exist

% otherwise use the object that was found.

if isempty(obj1)

obj1 = visa('AGILENT', 'GPIB0::14::INSTR');

else

fclose(obj1);

obj1 = obj1(1);

end

% Connect to instrument object, obj1.

fopen(obj1);

%% Code exported from tmtool for AMREL powersupply

% Find a VISA-GPIB object.

obj2 = instrfind('Type', 'visa-gpib', 'RsrcName', 'GPIB0::12::0::INSTR', 'Tag', '');

% Create the VISA-GPIB object if it does not exist

% otherwise use the object that was found.

if isempty(obj2)

obj2 = visa('AGILENT', 'GPIB0::12::0::INSTR');

else

fclose(obj2);

obj2 = obj2(1);

end

% Connect to instrument object, obj2.

fopen(obj2);

%% Automating Photoelectric effect data acquistion

%Prompts for the user

vstart = input('Input your desired initial value for the powersupply voltage (in V): ');

vend = input('Input your desired final value for the powersupply voltage(in V): ');

vstep = input('Input the step size for the powersupply voltage (in V): ');

n = (vend - vstart)/vstep + 1;

voltage = (vstart:vstep:vend)';

%initialize output data vector

current = zeros(n,1);

for i = 1:n

% Communicating with instrument object, obj2: Setting voltage

fprintf(obj2, horzcat('vset ', num2str(voltage(i))));

pause(1.0); %pause for 1 sec for current to change

% Communicating with instrument object, obj1: Reading current

tempcurrent = fscanf(obj1);

cutcurrent = tempcurrent(5:end);

current(i) = str2double(cutcurrent);

end

data = [voltage, current];

fprintf(obj2, 'vset 0');

volt\_list = data(:,1);

current\_list = data(:,2);

grid on

hold off

plot(volt\_list, current\_list, 'o');

Appendix B: MATLAB script used for data analysis

% Import Data

TITLE= 'Select the file with the data you want to bring into MATLAB';

[filename,filepath] = uigetfile('\*.\*', TITLE); %Prompts the user to select a data file

full\_filename = fullfile( filepath, filename );

[~,SheetNames] = xlsfinfo(full\_filename);

nSheets = length(SheetNames);

Data = [];

for ii=1:nSheets

Name = SheetNames{ii};

Data = [Data, xlsread(full\_filename, Name)];

end

forward\_voltage\_V577 = Data(:,1);

forward\_current\_A577 = Data(:,2);

reverse\_voltage\_V577 = Data(:,4);

reverse\_current\_A577 = Data(:,5);

forward\_voltage\_V546 = Data(:,6);

forward\_current\_A546 = Data(:,7);

reverse\_voltage\_V546 = Data(:,9);

reverse\_current\_A546 = Data(:,10);

forward\_voltage\_V436 = Data(:,11);

forward\_current\_A436 = Data(:,12);

reverse\_voltage\_V436 = Data(:,14);

reverse\_current\_A436 = Data(:,15);

forward\_voltage\_V405 = Data(:,16);

forward\_current\_A405 = Data(:,17);

reverse\_voltage\_V405 = Data(:,19);

reverse\_current\_A405 = Data(:,20);

forward\_voltage\_V365 = Data(:,21);

forward\_current\_A365 = Data(:,22);

reverse\_voltage\_V365 = Data(:,24);

reverse\_current\_A365 = Data(:,25);

%%%%% 577 analysis

%%% FORWARD

% Calculate average of three runs

firstrun577 = forward\_current\_A577(1:301);

secondrun577 = forward\_current\_A577(302:602);

thirdrun577 = forward\_current\_A577(603:903);

averagecurrent577 = [];

current\_standard\_dev\_577 = [];

for i = 1:301;

averagecurrent577i = (firstrun577(i) + secondrun577(i) + thirdrun577(i))/3;

current\_standard\_dev\_577(i) = std([firstrun577(i); secondrun577(i); thirdrun577(i)]);

averagecurrent577(i) = averagecurrent577i;

end

voltagetouse577 = forward\_voltage\_V577(1:301);

% Uncertainty in Voltage

voltage\_unc\_577 = 0.001\*ones(size(voltagetouse577));

% Uncertainty in Current

current\_standard\_error\_577 = current\_standard\_dev\_577/ sqrt(3);

% Find imax

figure

fitop\_577 = fitoptions('Method','NonlinearLeastSquares','StartPoint',[4e-8 8e-10 0.08]);

imax\_fittype\_577 = fittype('a-b\*exp(-c\*x)','options',fitop\_577);

imax\_577\_fit = fit(voltagetouse577,averagecurrent577',imax\_fittype\_577);

imax\_577\_fit\_coeff = coeffvalues(imax\_577\_fit);

errorbar(voltagetouse577, averagecurrent577',current\_standard\_error\_577,current\_standard\_error\_577,voltage\_unc\_577,voltage\_unc\_577);

hold on;

plot(imax\_577\_fit);

imax\_577 = imax\_577\_fit\_coeff(1);

title("Fitted Exponential Curve to Current vs. Voltage (577nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

%%% REVERSE

% Calculate average of three runs

firstrun\_reverse\_577 = reverse\_current\_A577(1:41);

secondrun\_reverse\_577 = reverse\_current\_A577(42:82);

thirdrun\_reverse\_577 = reverse\_current\_A577(83:123);

averagecurrent\_reverse\_577 =[];

current\_reverse\_standard\_dev\_577 = [];

for i = 1:41;

averagecurrent\_reverse\_577i = (firstrun\_reverse\_577(i) + secondrun\_reverse\_577(i) + thirdrun\_reverse\_577(i)) / 3;

current\_reverse\_standard\_dev\_577(i) = std([firstrun577(i); secondrun577(i); thirdrun577(i)]);

averagecurrent\_reverse\_577(i) = averagecurrent\_reverse\_577i;

end

voltagetouse\_reverse\_577 = reverse\_voltage\_V577(1:41);

voltagetouse\_reverse\_577 = -voltagetouse\_reverse\_577;

% Uncertainty in Voltage

voltage\_unc\_reverse\_577 = 0.001\*ones(size(voltagetouse\_reverse\_577));

% Uncertainty in Current

current\_reverse\_standard\_error\_577 = current\_reverse\_standard\_dev\_577/ sqrt(3);

% Plot a figure with both data sets on one graph

figure

errorbar(voltagetouse577, averagecurrent577',current\_standard\_error\_577,current\_standard\_error\_577,voltage\_unc\_577,voltage\_unc\_577);

hold on;

errorbar(voltagetouse\_reverse\_577, averagecurrent\_reverse\_577,current\_reverse\_standard\_error\_577,current\_reverse\_standard\_error\_577,voltage\_unc\_reverse\_577,voltage\_unc\_reverse\_577)

plot(voltagetouse577, averagecurrent577');

plot(voltagetouse\_reverse\_577, averagecurrent\_reverse\_577);

title("Current vs. Voltage (577nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Plot just the forward bias

figure

errorbar(voltagetouse577, averagecurrent577',current\_standard\_error\_577,current\_standard\_error\_577,voltage\_unc\_577,voltage\_unc\_577);

title("Forward Bias Current vs. Voltage (577nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Plot just the reverse bias

figure

errorbar(voltagetouse\_reverse\_577, averagecurrent\_reverse\_577,current\_reverse\_standard\_error\_577,current\_reverse\_standard\_error\_577,voltage\_unc\_reverse\_577,voltage\_unc\_reverse\_577)

title("Reverse Bias Current vs. Voltage (577nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Find knee using plateau uncertaities and deviation from that value

% Calculate aveage data point

running\_total\_577 = 0;

h577 = averagecurrent\_reverse\_577(20:35);

for i = 1:6;

running\_total\_577 = h577(i) + running\_total\_577;

end

average\_threshold\_point\_577 = running\_total\_577 / 6;

% Calculate average error bar size

running\_total\_577 = 0;

g577 = current\_reverse\_standard\_error\_577(20:35);

for i = 1:6;

running\_total\_577 = g577(i)^2 + running\_total\_577;

end

uncertainty\_mean\_577 = sqrt(running\_total\_577)/6;

% Calculate standard deviation

standard\_dev\_577 = std(current\_reverse\_standard\_error\_577(20:35));

% Calculate threshold bar

threshold\_bar\_577 = sqrt(uncertainty\_mean\_577^2 + standard\_dev\_577^2);

% Calculate threshold

threshold\_577 = threshold\_bar\_577 + average\_threshold\_point\_577;

% Plot the threshold line

figure

errorbar(voltagetouse\_reverse\_577, averagecurrent\_reverse\_577,current\_reverse\_standard\_error\_577,current\_reverse\_standard\_error\_577,voltage\_unc\_reverse\_577,voltage\_unc\_reverse\_577)

title("Reverse Bias Current vs. Voltage (577nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

hold on;

x = [-2 : 0.5 : 0];

Z = threshold\_577 \* ones(1, length(x));

plot(x, Z)

% Find knee using intersection of flat slopes

% Calculating slope of top line

k577 = averagecurrent\_reverse\_577(1:9);

j577 = voltagetouse\_reverse\_577(1:9);

% Plot intersection of lines

figure

errorbar(voltagetouse\_reverse\_577, averagecurrent\_reverse\_577,current\_reverse\_standard\_error\_577,current\_reverse\_standard\_error\_577,voltage\_unc\_reverse\_577,voltage\_unc\_reverse\_577)

title("Reverse Bias Current vs. Voltage (577nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

hold on;

% Plot top line

c\_577 = polyfit(j577,k577',1);

xFit\_577 = linspace(-2, 0, 100);

yFit\_577 = polyval(c\_577, xFit\_577);

hold on;

plot(xFit\_577, yFit\_577);

% Plot bottom line

x = [-2 : 0.5 : 0];

% Use already calculated value for average\_threshold\_point to get bottom

% line

Z\_577 = average\_threshold\_point\_577 \* ones(1, length(x));

plot(x, Z\_577)

grid on;

%%%%% 546 analysis

%%% FORWARD

% Calculate average of three runs

firstrun546 = forward\_current\_A546(1:301);

secondrun546 = forward\_current\_A546(302:602);

thirdrun546 = forward\_current\_A546(603:903);

averagecurrent546 = [];

current\_standard\_dev\_546 = [];

for i = 1:301;

averagecurrent546i = (firstrun546(i) + secondrun546(i) + thirdrun546(i))/3;

current\_standard\_dev\_546(i) = std([firstrun546(i); secondrun546(i); thirdrun546(i)]);

averagecurrent546(i) = averagecurrent546i;

end

voltagetouse546 = forward\_voltage\_V546(1:301);

% Uncertainty in Voltage

voltage\_unc\_546 = 0.001\*ones(size(voltagetouse546));

% Uncertainty in Current

current\_standard\_error\_546 = current\_standard\_dev\_546/ sqrt(3);

% Find imax

figure

fitop\_546 = fitoptions('Method','NonlinearLeastSquares','StartPoint',[4e-8 4e-10 0.04]);

imax\_fittype\_546 = fittype('a-b\*exp(-c\*x)','options',fitop\_546);

imax\_546\_fit = fit(voltagetouse546,averagecurrent546',imax\_fittype\_546);

imax\_546\_fit\_coeff = coeffvalues(imax\_546\_fit);

errorbar(voltagetouse546, averagecurrent546',current\_standard\_error\_546,current\_standard\_error\_546,voltage\_unc\_546,voltage\_unc\_546);

hold on;

plot(imax\_546\_fit);

imax\_546 = imax\_546\_fit\_coeff(1);

title("Fitted Exponential Curve to Current vs. Voltage (546nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

%%% REVERSE

% Calculate average of three runs

firstrun\_reverse\_546 = reverse\_current\_A546(1:41);

secondrun\_reverse\_546 = reverse\_current\_A546(42:82);

thirdrun\_reverse\_546 = reverse\_current\_A546(83:123);

averagecurrent\_reverse\_546 =[];

current\_reverse\_standard\_dev\_546 = [];

for i = 1:41;

averagecurrent\_reverse\_546i = (firstrun\_reverse\_546(i) + secondrun\_reverse\_546(i) + thirdrun\_reverse\_546(i)) / 3;

current\_reverse\_standard\_dev\_546(i) = std([firstrun546(i); secondrun546(i); thirdrun546(i)]);

averagecurrent\_reverse\_546(i) = averagecurrent\_reverse\_546i;

end

voltagetouse\_reverse\_546 = reverse\_voltage\_V546(1:41);

voltagetouse\_reverse\_546 = -voltagetouse\_reverse\_546;

% Uncertainty in Voltage

voltage\_unc\_reverse\_546 = 0.001\*ones(size(voltagetouse\_reverse\_546));

% Uncertainty in Current

current\_reverse\_standard\_error\_546 = current\_reverse\_standard\_dev\_546/ sqrt(3);

% Plot a figure with both data sets on one graph

figure

errorbar(voltagetouse546, averagecurrent546',current\_standard\_error\_546,current\_standard\_error\_546,voltage\_unc\_546,voltage\_unc\_546);

hold on;

errorbar(voltagetouse\_reverse\_546, averagecurrent\_reverse\_546,current\_reverse\_standard\_error\_546,current\_reverse\_standard\_error\_546,voltage\_unc\_reverse\_546,voltage\_unc\_reverse\_546)

plot(voltagetouse546, averagecurrent546');

plot(voltagetouse\_reverse\_546, averagecurrent\_reverse\_546);

title("Current vs. Voltage (546nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Plot just the forward bias

figure

errorbar(voltagetouse546, averagecurrent546',current\_standard\_error\_546,current\_standard\_error\_546,voltage\_unc\_546,voltage\_unc\_546);

title("Forward Bias Current vs. Voltage (546nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Plot just the reverse bias

figure

errorbar(voltagetouse\_reverse\_546, averagecurrent\_reverse\_546,current\_reverse\_standard\_error\_546,current\_reverse\_standard\_error\_546,voltage\_unc\_reverse\_546,voltage\_unc\_reverse\_546)

title("Reverse Bias Current vs. Voltage (546nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Find knee using plateau uncertaities and deviation from that value

% Calculate aveage data point

running\_total\_546 = 0;

h546 = averagecurrent\_reverse\_546(35:40);

for i = 1:6;

running\_total\_546 = h546(i) + running\_total\_546;

end

average\_threshold\_point\_546 = running\_total\_546 / 6;

% Calculate average error bar size

running\_total\_546 = 0;

g546 = current\_reverse\_standard\_error\_546(35:40);

for i = 1:6;

running\_total\_546 = g546(i)^2 + running\_total\_546;

end

uncertainty\_mean\_546 = sqrt(running\_total\_546)/6;

% Calculate standard deviation

standard\_dev\_546 = std(current\_reverse\_standard\_error\_546(35:40));

% Calculate threshold bar

threshold\_bar\_546 = sqrt(uncertainty\_mean\_546^2 + standard\_dev\_546^2);

% Calculate threshold

threshold\_546 = threshold\_bar\_546 + average\_threshold\_point\_546;

% Plot the threshold line

figure

errorbar(voltagetouse\_reverse\_546, averagecurrent\_reverse\_546,current\_reverse\_standard\_error\_546,current\_reverse\_standard\_error\_546,voltage\_unc\_reverse\_546,voltage\_unc\_reverse\_546)

title("Reverse Bias Current vs. Voltage (546nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

hold on;

x = [-2 : 0.5 : 0];

Z = threshold\_546 \* ones(1, length(x));

plot(x, Z)

% Find knee using intersection of flat slopes

% Calculating slope of top line

k546 = averagecurrent\_reverse\_546(5:20);

j546 = voltagetouse\_reverse\_546(5:20);

% Plot intersection of lines

figure

errorbar(voltagetouse\_reverse\_546, averagecurrent\_reverse\_546,current\_reverse\_standard\_error\_546,current\_reverse\_standard\_error\_546,voltage\_unc\_reverse\_546,voltage\_unc\_reverse\_546)

title("Reverse Bias Current vs. Voltage (546nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

hold on;

% Plot top line

c\_546 = polyfit(j546,k546',1);

xFit\_546 = linspace(-2, 0, 100);

yFit\_546 = polyval(c\_546, xFit\_546);

hold on;

plot(xFit\_546, yFit\_546);

% Plot bottom line

x = [-2 : 0.5 : 0];

% Use already calculated value for average\_threshold\_point to get bottom

% line

Z\_546 = average\_threshold\_point\_546 \* ones(1, length(x));

plot(x, Z\_546)

grid on;

%%%%% 436 analysis

%%% FORWARD

% Calculate average of three runs

firstrun436 = forward\_current\_A436(1:301);

secondrun436 = forward\_current\_A436(302:602);

thirdrun436 = forward\_current\_A436(603:903);

averagecurrent436 = [];

current\_standard\_dev\_436 = [];

for i = 1:301;

averagecurrent436i = (firstrun436(i) + secondrun436(i) + thirdrun436(i))/3;

current\_standard\_dev\_436(i) = std([firstrun436(i); secondrun436(i); thirdrun436(i)]);

averagecurrent436(i) = averagecurrent436i;

end

voltagetouse436 = forward\_voltage\_V436(1:301);

% Uncertainty in Voltage

voltage\_unc\_436 = 0.001\*ones(size(voltagetouse436));

% Uncertainty in Current

current\_standard\_error\_436 = current\_standard\_dev\_436/ sqrt(3);

% Find imax

figure

fitop\_436 = fitoptions('Method','NonlinearLeastSquares','StartPoint',[4e-8 4e-10 0.04]);

imax\_fittype\_436 = fittype('a-b\*exp(-c\*x)','options',fitop\_436);

imax\_436\_fit = fit(voltagetouse436,averagecurrent436',imax\_fittype\_436);

imax\_436\_fit\_coeff = coeffvalues(imax\_436\_fit);

errorbar(voltagetouse436, averagecurrent436',current\_standard\_error\_436,current\_standard\_error\_436,voltage\_unc\_436,voltage\_unc\_436);

hold on;

plot(imax\_436\_fit);

imax\_436 = imax\_436\_fit\_coeff(1);

title("Fitted Exponential Curve to Current vs. Voltage (436nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

%%% REVERSE

% Calculate average of three runs

firstrun\_reverse\_436 = reverse\_current\_A436(1:41);

secondrun\_reverse\_436 = reverse\_current\_A436(42:82);

thirdrun\_reverse\_436 = reverse\_current\_A436(83:123);

averagecurrent\_reverse\_436 =[];

current\_reverse\_standard\_dev\_436 = [];

for i = 1:41;

averagecurrent\_reverse\_436i = (firstrun\_reverse\_436(i) + secondrun\_reverse\_436(i) + thirdrun\_reverse\_436(i)) / 3;

current\_reverse\_standard\_dev\_436(i) = std([firstrun436(i); secondrun436(i); thirdrun436(i)]);

averagecurrent\_reverse\_436(i) = averagecurrent\_reverse\_436i;

end

voltagetouse\_reverse\_436 = reverse\_voltage\_V436(1:41);

voltagetouse\_reverse\_436 = -voltagetouse\_reverse\_436;

% Uncertainty in Voltage

voltage\_unc\_reverse\_436 = 0.001\*ones(size(voltagetouse\_reverse\_436));

% Uncertainty in Current

current\_reverse\_standard\_error\_436 = current\_reverse\_standard\_dev\_436/ sqrt(3);

% Plot a figure with both data sets on one graph

figure

errorbar(voltagetouse436, averagecurrent436',current\_standard\_error\_436,current\_standard\_error\_436,voltage\_unc\_436,voltage\_unc\_436);

hold on;

errorbar(voltagetouse\_reverse\_436, averagecurrent\_reverse\_436,current\_reverse\_standard\_error\_436,current\_reverse\_standard\_error\_436,voltage\_unc\_reverse\_436,voltage\_unc\_reverse\_436)

plot(voltagetouse436, averagecurrent436');

plot(voltagetouse\_reverse\_436, averagecurrent\_reverse\_436);

title("Current vs. Voltage (436nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Plot just the forward bias

figure

errorbar(voltagetouse436, averagecurrent436',current\_standard\_error\_436,current\_standard\_error\_436,voltage\_unc\_436,voltage\_unc\_436);

title("Forward Bias Current vs. Voltage (436nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Plot just the reverse bias

figure

errorbar(voltagetouse\_reverse\_436, averagecurrent\_reverse\_436,current\_reverse\_standard\_error\_436,current\_reverse\_standard\_error\_436,voltage\_unc\_reverse\_436,voltage\_unc\_reverse\_436)

title("Reverse Bias Current vs. Voltage (436nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Find knee using plateau uncertaities and deviation from that value

% Calculate aveage data point

running\_total\_436 = 0;

h436 = averagecurrent\_reverse\_436(35:40);

for i = 1:6;

running\_total\_436 = h436(i) + running\_total\_436;

end

average\_threshold\_point\_436 = running\_total\_436 / 6;

% Calculate average error bar size

running\_total\_436 = 0;

g436 = current\_reverse\_standard\_error\_436(35:40);

for i = 1:6;

running\_total\_436 = g436(i)^2 + running\_total\_436;

end

uncertainty\_mean\_436 = sqrt(running\_total\_436)/6;

% Calculate standard deviation

standard\_dev\_436 = std(current\_reverse\_standard\_error\_436(35:40));

% Calculate threshold bar

threshold\_bar\_436 = sqrt(uncertainty\_mean\_436^2 + standard\_dev\_436^2);

% Calculate threshold

threshold\_436 = threshold\_bar\_436 + average\_threshold\_point\_436;

% Plot the threshold line

figure

errorbar(voltagetouse\_reverse\_436, averagecurrent\_reverse\_436,current\_reverse\_standard\_error\_436,current\_reverse\_standard\_error\_436,voltage\_unc\_reverse\_436,voltage\_unc\_reverse\_436)

title("Reverse Bias Current vs. Voltage (436nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

hold on;

x = [-2 : 0.5 : 0];

Z = threshold\_436 \* ones(1, length(x));

plot(x, Z)

% Find knee using intersection of flat slopes

% Calculating slope of top line

k436 = averagecurrent\_reverse\_436(5:20);

j436 = voltagetouse\_reverse\_436(5:20);

% Plot intersection of lines

figure

errorbar(voltagetouse\_reverse\_436, averagecurrent\_reverse\_436,current\_reverse\_standard\_error\_436,current\_reverse\_standard\_error\_436,voltage\_unc\_reverse\_436,voltage\_unc\_reverse\_436)

title("Reverse Bias Current vs. Voltage (436nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

hold on;

% Plot top line

c\_436 = polyfit(j436,k436',1);

xFit\_436 = linspace(-2, 0, 100);

yFit\_436 = polyval(c\_436, xFit\_436);

hold on;

plot(xFit\_436, yFit\_436);

% Plot bottom line

x = [-2 : 0.5 : 0];

% Use already calculated value for average\_threshold\_point to get bottom

% line

Z\_436 = average\_threshold\_point\_436 \* ones(1, length(x));

plot(x, Z\_436)

grid on;

%%%%% 405 analysis

%%% FORWARD

% Calculate average of three runs

firstrun405 = forward\_current\_A405(1:301);

secondrun405 = forward\_current\_A405(302:602);

thirdrun405 = forward\_current\_A405(603:903);

averagecurrent405 = [];

current\_standard\_dev\_405 = [];

for i = 1:301;

averagecurrent405i = (firstrun405(i) + secondrun405(i) + thirdrun405(i))/3;

current\_standard\_dev\_405(i) = std([firstrun405(i); secondrun405(i); thirdrun405(i)]);

averagecurrent405(i) = averagecurrent405i;

end

voltagetouse405 = forward\_voltage\_V405(1:301);

% Uncertainty in Voltage

voltage\_unc\_405 = 0.001\*ones(size(voltagetouse405));

% Uncertainty in Current

current\_standard\_error\_405 = current\_standard\_dev\_405/ sqrt(3);

% Find imax

figure

fitop\_405 = fitoptions('Method','NonlinearLeastSquares','StartPoint',[4e-8 4e-10 0.04]);

imax\_fittype\_405 = fittype('a-b\*exp(-c\*x)','options',fitop\_405);

imax\_405\_fit = fit(voltagetouse405,averagecurrent405',imax\_fittype\_405);

imax\_405\_fit\_coeff = coeffvalues(imax\_405\_fit);

errorbar(voltagetouse405, averagecurrent405',current\_standard\_error\_405,current\_standard\_error\_405,voltage\_unc\_405,voltage\_unc\_405);

hold on;

plot(imax\_405\_fit);

imax\_405 = imax\_405\_fit\_coeff(1);

title("Fitted Exponential Curve to Current vs. Voltage (405nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

%%% REVERSE

% Calculate average of three runs

firstrun\_reverse\_405 = reverse\_current\_A405(1:41);

secondrun\_reverse\_405 = reverse\_current\_A405(42:82);

thirdrun\_reverse\_405 = reverse\_current\_A405(83:123);

averagecurrent\_reverse\_405 =[];

current\_reverse\_standard\_dev\_405 = [];

for i = 1:41;

averagecurrent\_reverse\_405i = (firstrun\_reverse\_405(i) + secondrun\_reverse\_405(i) + thirdrun\_reverse\_405(i)) / 3;

current\_reverse\_standard\_dev\_405(i) = std([firstrun405(i); secondrun405(i); thirdrun405(i)]);

averagecurrent\_reverse\_405(i) = averagecurrent\_reverse\_405i;

end

voltagetouse\_reverse\_405 = reverse\_voltage\_V405(1:41);

voltagetouse\_reverse\_405 = -voltagetouse\_reverse\_405;

% Uncertainty in Voltage

voltage\_unc\_reverse\_405 = 0.001\*ones(size(voltagetouse\_reverse\_405));

% Uncertainty in Current

current\_reverse\_standard\_error\_405 = current\_reverse\_standard\_dev\_405/ sqrt(3);

% Plot a figure with both data sets on one graph

figure

errorbar(voltagetouse405, averagecurrent405',current\_standard\_error\_405,current\_standard\_error\_405,voltage\_unc\_405,voltage\_unc\_405);

hold on;

errorbar(voltagetouse\_reverse\_405, averagecurrent\_reverse\_405,current\_reverse\_standard\_error\_405,current\_reverse\_standard\_error\_405,voltage\_unc\_reverse\_405,voltage\_unc\_reverse\_405)

plot(voltagetouse405, averagecurrent405');

plot(voltagetouse\_reverse\_405, averagecurrent\_reverse\_405);

title("Current vs. Voltage (405nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Plot just the forward bias

figure

errorbar(voltagetouse405, averagecurrent405',current\_standard\_error\_405,current\_standard\_error\_405,voltage\_unc\_405,voltage\_unc\_405);

title("Forward Bias Current vs. Voltage (405nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Plot just the reverse bias

figure

errorbar(voltagetouse\_reverse\_405, averagecurrent\_reverse\_405,current\_reverse\_standard\_error\_405,current\_reverse\_standard\_error\_405,voltage\_unc\_reverse\_405,voltage\_unc\_reverse\_405)

title("Reverse Bias Current vs. Voltage (405nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Find knee using plateau uncertaities and deviation from that value

% Calculate aveage data point

running\_total\_405 = 0;

h405 = averagecurrent\_reverse\_405(35:40);

for i = 1:6;

running\_total\_405 = h405(i) + running\_total\_405;

end

average\_threshold\_point\_405 = running\_total\_405 / 6;

% Calculate average error bar size

running\_total\_405 = 0;

g405 = current\_reverse\_standard\_error\_405(35:40);

for i = 1:6;

running\_total\_405 = g405(i)^2 + running\_total\_405;

end

uncertainty\_mean\_405 = sqrt(running\_total\_405)/6;

% Calculate standard deviation

standard\_dev\_405 = std(current\_reverse\_standard\_error\_405(35:40));

% Calculate threshold bar

threshold\_bar\_405 = sqrt(uncertainty\_mean\_405^2 + standard\_dev\_405^2);

% Calculate threshold

threshold\_405 = threshold\_bar\_405 + average\_threshold\_point\_405;

% Plot the threshold line

figure

errorbar(voltagetouse\_reverse\_405, averagecurrent\_reverse\_405,current\_reverse\_standard\_error\_405,current\_reverse\_standard\_error\_405,voltage\_unc\_reverse\_405,voltage\_unc\_reverse\_405)

title("Reverse Bias Current vs. Voltage (405nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

hold on;

x = [-2 : 0.5 : 0];

Z = threshold\_405 \* ones(1, length(x));

plot(x, Z)

% Find knee using intersection of flat slopes

% Calculating slope of top line

k405 = averagecurrent\_reverse\_405(5:20);

j405 = voltagetouse\_reverse\_405(5:20);

% Plot intersection of lines

figure

errorbar(voltagetouse\_reverse\_405, averagecurrent\_reverse\_405,current\_reverse\_standard\_error\_405,current\_reverse\_standard\_error\_405,voltage\_unc\_reverse\_405,voltage\_unc\_reverse\_405)

title("Reverse Bias Current vs. Voltage (405nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

hold on;

% Plot top line

c\_405 = polyfit(j405,k405',1);

xFit\_405 = linspace(-2, 0, 100);

yFit\_405 = polyval(c\_405, xFit\_405);

hold on;

plot(xFit\_405, yFit\_405);

% Plot bottom line

x = [-2 : 0.5 : 0];

% Use already calculated value for average\_threshold\_point to get bottom

% line

Z\_405 = average\_threshold\_point\_405 \* ones(1, length(x));

plot(x, Z\_405)

grid on;

%%%%% 365 analysis

%%% FORWARD

% Calculate average of three runs

firstrun365 = forward\_current\_A365(1:301);

secondrun365 = forward\_current\_A365(302:602);

thirdrun365 = forward\_current\_A365(603:903);

averagecurrent365 = [];

current\_standard\_dev\_365 = [];

for i = 1:301;

averagecurrent365i = (firstrun365(i) + secondrun365(i) + thirdrun365(i))/3;

current\_standard\_dev\_365(i) = std([firstrun365(i); secondrun365(i); thirdrun365(i)]);

averagecurrent365(i) = averagecurrent365i;

end

voltagetouse365 = forward\_voltage\_V365(1:301);

% Uncertainty in Voltage

voltage\_unc\_365 = 0.001\*ones(size(voltagetouse365));

% Uncertainty in Current

current\_standard\_error\_365 = current\_standard\_dev\_365/ sqrt(3);

% Find imax

figure

fitop\_365 = fitoptions('Method','NonlinearLeastSquares','StartPoint',[4e-8 4e-10 0.04]);

imax\_fittype\_365 = fittype('a-b\*exp(-c\*x)','options',fitop\_365);

imax\_365\_fit = fit(voltagetouse365,averagecurrent365',imax\_fittype\_365);

imax\_365\_fit\_coeff = coeffvalues(imax\_365\_fit);

errorbar(voltagetouse365, averagecurrent365',current\_standard\_error\_365,current\_standard\_error\_365,voltage\_unc\_365,voltage\_unc\_365);

hold on;

plot(imax\_365\_fit);

imax\_365 = imax\_365\_fit\_coeff(1);

title("Fitted Exponential Curve to Current vs. Voltage (365nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

%%% REVERSE

% Calculate average of three runs

firstrun\_reverse\_365 = reverse\_current\_A365(1:41);

secondrun\_reverse\_365 = reverse\_current\_A365(42:82);

thirdrun\_reverse\_365 = reverse\_current\_A365(83:123);

averagecurrent\_reverse\_365 =[];

current\_reverse\_standard\_dev\_365 = [];

for i = 1:41;

averagecurrent\_reverse\_365i = (firstrun\_reverse\_365(i) + secondrun\_reverse\_365(i) + thirdrun\_reverse\_365(i)) / 3;

current\_reverse\_standard\_dev\_365(i) = std([firstrun365(i); secondrun365(i); thirdrun365(i)]);

averagecurrent\_reverse\_365(i) = averagecurrent\_reverse\_365i;

end

voltagetouse\_reverse\_365 = reverse\_voltage\_V365(1:41);

voltagetouse\_reverse\_365 = -voltagetouse\_reverse\_365;

% Uncertainty in Voltage

voltage\_unc\_reverse\_365 = 0.001\*ones(size(voltagetouse\_reverse\_365));

% Uncertainty in Current

current\_reverse\_standard\_error\_365 = current\_reverse\_standard\_dev\_365/ sqrt(3);

% Plot a figure with both data sets on one graph

figure

errorbar(voltagetouse365, averagecurrent365',current\_standard\_error\_365,current\_standard\_error\_365,voltage\_unc\_365,voltage\_unc\_365);

hold on;

errorbar(voltagetouse\_reverse\_365, averagecurrent\_reverse\_365,current\_reverse\_standard\_error\_365,current\_reverse\_standard\_error\_365,voltage\_unc\_reverse\_365,voltage\_unc\_reverse\_365)

plot(voltagetouse365, averagecurrent365');

plot(voltagetouse\_reverse\_365, averagecurrent\_reverse\_365);

title("Current vs. Voltage (365nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Plot just the forward bias

figure

errorbar(voltagetouse365, averagecurrent365',current\_standard\_error\_365,current\_standard\_error\_365,voltage\_unc\_365,voltage\_unc\_365);

title("Forward Bias Current vs. Voltage (365nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Plot just the reverse bias

figure

errorbar(voltagetouse\_reverse\_365, averagecurrent\_reverse\_365,current\_reverse\_standard\_error\_365,current\_reverse\_standard\_error\_365,voltage\_unc\_reverse\_365,voltage\_unc\_reverse\_365)

title("Reverse Bias Current vs. Voltage (365nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

% Find knee using plateau uncertaities and deviation from that value

% Calculate aveage data point

running\_total\_365 = 0;

h365 = averagecurrent\_reverse\_365(35:40);

for i = 1:6;

running\_total\_365 = h365(i) + running\_total\_365;

end

average\_threshold\_point\_365 = running\_total\_365 / 6;

% Calculate average error bar size

running\_total\_365 = 0;

g365 = current\_reverse\_standard\_error\_365(35:40);

for i = 1:6;

running\_total\_365 = g365(i)^2 + running\_total\_365;

end

uncertainty\_mean\_365 = sqrt(running\_total\_365)/6;

% Calculate standard deviation

standard\_dev\_365 = std(current\_reverse\_standard\_error\_365(35:40));

% Calculate threshold bar

threshold\_bar\_365 = sqrt(uncertainty\_mean\_365^2 + standard\_dev\_365^2);

% Calculate threshold

threshold\_365 = threshold\_bar\_365 + average\_threshold\_point\_365;

% Plot the threshold line

figure

errorbar(voltagetouse\_reverse\_365, averagecurrent\_reverse\_365,current\_reverse\_standard\_error\_365,current\_reverse\_standard\_error\_365,voltage\_unc\_reverse\_365,voltage\_unc\_reverse\_365)

title("Reverse Bias Current vs. Voltage (365nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

hold on;

x = [-2 : 0.5 : 0];

Z = threshold\_365 \* ones(1, length(x));

plot(x, Z)

% Find knee using intersection of flat slopes

% Calculating slope of top line

k365 = averagecurrent\_reverse\_365(5:20);

j365 = voltagetouse\_reverse\_365(5:20);

% Plot intersection of lines

figure

errorbar(voltagetouse\_reverse\_365, averagecurrent\_reverse\_365,current\_reverse\_standard\_error\_365,current\_reverse\_standard\_error\_365,voltage\_unc\_reverse\_365,voltage\_unc\_reverse\_365)

title("Reverse Bias Current vs. Voltage (365nm Filter)")

xlabel('Voltage (V)')

ylabel('Current (A)')

hold on;

% Plot top line

c\_365 = polyfit(j365,k365',1);

xFit\_365 = linspace(-2, 0, 100);

yFit\_365 = polyval(c\_365, xFit\_365);

hold on;

plot(xFit\_365, yFit\_365);

% Plot bottom line

x = [-2 : 0.5 : 0];

% Use already calculated value for average\_threshold\_point to get bottom

% line

Z\_365 = average\_threshold\_point\_365 \* ones(1, length(x));

plot(x, Z\_365)

grid on;

% stopping voltage vs. frequency plotting

two\_line\_stop\_V = [0.47, 0.76, 0.92, 1.11, 1.25];

one\_line\_stop\_V = [0.45, 0.45, 0.85, 0.96, 1.35];

% Average values from both methods

avg\_stop\_V = [0.46, 0.605, 0.885, 1.035, 1.3];

frequency = [5.20\*10^14, 5.49\*10^14, 6.88\*10^14, 7.41\*10^14, 8.22\*10^14];

figure

plot(frequency, avg\_stop\_V,'o')

hold on;

fit1 = polyfit(frequency,avg\_stop\_V,1);

xFit = linspace(0, 8.5e14, 100);

yFit1 = polyval(fit1, frequency);

plot(frequency,yFit1)

title("Averaged Stopping Voltage vs. Light Frequency with Two linear Fits")

xlabel("Light Frequency (Hz)")

ylabel("Stopping Voltage (V)")

grid on;

hold off;

work\_func = -fit1(2)

figure

plot(frequency, two\_line\_stop\_V,'o')

hold on;

fit1 = polyfit(frequency,two\_line\_stop\_V,1);

xFit = linspace(0, 8.5e14, 100);

yFit1 = polyval(fit1, frequency);

plot(frequency,yFit1)

title("Stopping Voltage vs. Light Frequency with Intersedction Method")

xlabel("Light Frequency (Hz)")

ylabel("Stopping Voltage (V)")

grid on;

hold off;

figure

plot(frequency, one\_line\_stop\_V,'o')

hold on;

fit1 = polyfit(frequency,one\_line\_stop\_V,1);

xFit = linspace(0, 8.5e14, 100);

yFit1 = polyval(fit1, frequency);

plot(frequency,yFit1)

title("Stopping Voltage vs. Light Frequency with Threshold Method")

xlabel("Light Frequency (Hz)")

ylabel("Stopping Voltage (V)")

grid on;

hold off;