The Photo-Electric Effect

Light was thought of as a wave before the Photo-Electric effect proved it acted otherwise, it had wavelike properties, diffraction, interference, etc. However, if light truly acted as a wave then when it would transfer its energy to a surface it would depend on the intensity of the light and not the frequency however, the Photo-Electric effect proved that is not the case. Light was proven to act as a particle with quantized energy.

$$E = hv$$

The energy of a particle is given by Planck's constant: h, and v is the frequency of the light.

The energy required to knock an electron off a surface is dependent on the surface and is given by ϕ and is called the work function of the material.

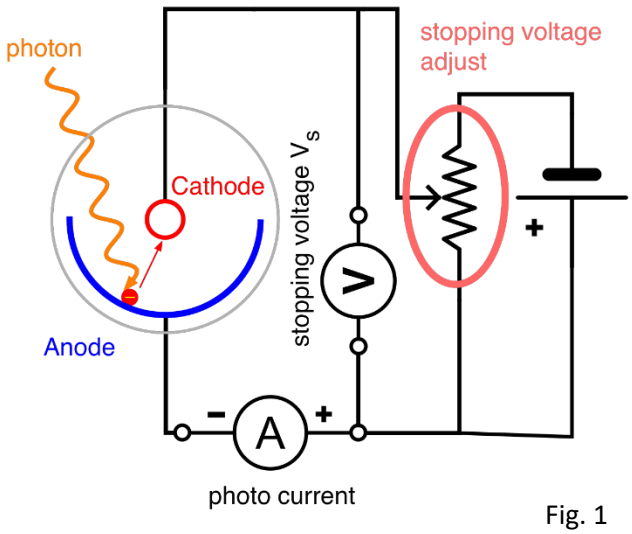

Fig. 1

Following the schematic to the right, we will be adjusting the voltage difference between the anode and cathode to find the minimum amount of kinetic energy the electrons need to reach the cathode.

$$hv = eV_s + \phi$$

At the minimum energy $V_s = 0$ we get the following equation

$$v_0 = \frac{\phi}{h}$$

We will be graphing voltage versus current to determine the point at which V_s becomes 0. Since we know the frequencies we are using, and h is constant we will be able to determine both h and the work function for the surface by graphing eV_s vs the frequency following,

$$hv - \Phi = eV_s$$

Where the slope is h and the offset is the work function of the material.

Uncertainties/Data Analysis

Uncertainties that exist and were considered include

$$\sigma_{V_s} \quad \sigma_I \quad \sigma_{adjust} \quad \sigma_\lambda$$

σ_{V_s}: Uncertainty in potential voltage

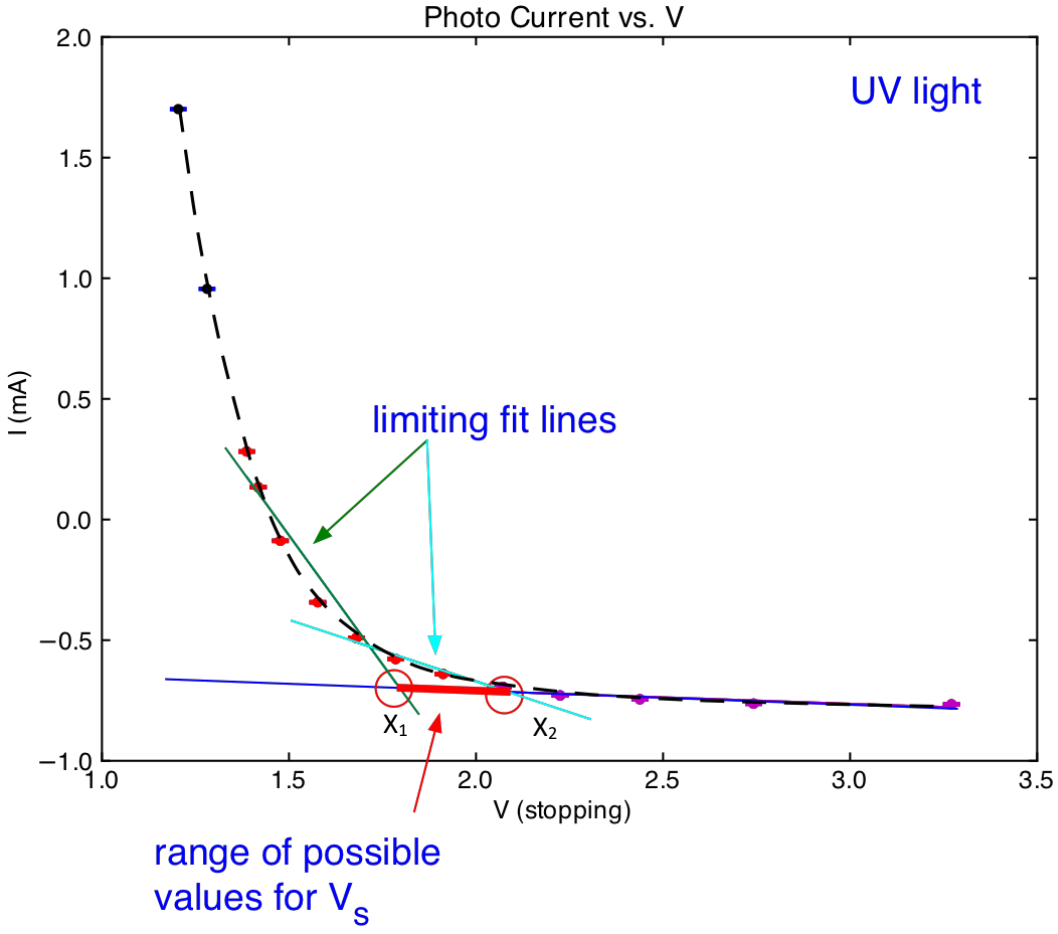

Fig. 2

Each limiting fit line has an uncertainty in its fitting. So technically we have an uncertainty in the uncertainty. In order to keep things simple, we will consider them insignificant (they are) and choose our significant figures such as to make them disappear (2 decimal places). We considered σ_{V_s} to be x_2-x_1 where x is the intercept of the limiting fit lines. I could've chosen the midpoint between x_2 and x_1 and the uncertainty would've been $\pm \frac{x_2 - x_1}{2}$ But, I chose x_2 to be my V_s.

σ_{adjust}: In the experiment we had to adjust the “Zero Adjust” to ideally get a current of 0, this was very hard to do so we got a small value and subtracted it from the data. This can be considered as a reference point. The multimeter would oscillate its final digit(s), so we rounded up and considered it an uncertainty.

σ_I: The multimeter would oscillate its final digit(s) when we would record the current throughout the experiment, we would record this rounded up and consider it an uncertainty. Since σ_{adjust} and this value are related we can just add them up to get $\sigma_{I\,new} = \sigma_{I\,old} + \sigma_{adjust}$ No quadrature needed.

σ_λ: In the experiment we had to optimize the monochromator for each color, of course there is error and uncertainty in this. If the wavelength was λ I considered the uncertainty $\pm\sqrt[3]{\lambda}$ This was chosen because it gave a realistic range of possible λ. This of course is a very coarse estimation of the true uncertainty, therefore this value is only graphed and mentioned but, it is not considered in the fit. (Also, because the programming of fitting x-axis uncertainty is not included in LT.box I will have to create my own)

Uncertainties/Assumptions that were not considered

The monochromator: The mercury lamp source not monochromatic, in order to get a close to a monochromatic beam you need to use the properties of a prism or of a diffraction grating, this is of course not perfect and has errors. (We get a small range of frequencies hitting our surface)

Background light: There is always the possibility that background light affected our experiment.

Resistance in current: Current had to travel from one place to another, there is resistance in those wires. This leads to systematic bias in our data which is more significant the lower the current.

Metallic surface: The surface which the beam hit is not perfectly uniform and therefore has differing values of work function ϕ

Perfect Vacuum: The photoelectric effect should be considered in a perfect vacuum; however, this is hard to accomplish experimentally so of course, we have uncertainties here. (We need a vacuum because air would affect not only the beam but, the electrons as well.)

As mentioned previously, we have

$$E = hf \qquad hv = K_e + \phi$$

Where the energy of the photon should represent the energy given to the electron due to the conservation of energy.

Of course, this is following the assumption that

$$eV_s = K_e$$

Which is most likely not the case. Either way, we get

$$hv - \Phi = eV_s$$

Which represents the formula for a line

$$y = mx + b$$

Graphing eV_s vs v we can get the value of h as slope and ϕ as intercept

In fig.2 we will have to find the intercepts between two lines.

This can be done by manipulating the formula for a line.

$$y_i = m_i x_{int} + b_i$$

At the interception y_1 should be equal to y_2 so we get (x_{int} is x-intercept)

$$m_1 x_{int} + b_1 = m_2 x_{int} + b_2$$

Rewriting

$$x_{int} = \frac{b_2 - b_1}{m_1 - m_2}$$

And since at the interception, $y_1 = y_2$ we can plug the value for x_{int} in any of the two line equations to get the y-intercept. Uncertainty in these intercepts are discussed later.

Percent difference and percent uncertainty are used later and are found by

$$\%\ Difference = \frac{|Accepted - Experimental|}{0.5 \cdot (Accepted + Experimental)} \cdot 100$$

$$\%\ Uncertainty = \frac{\sigma_M}{M}$$

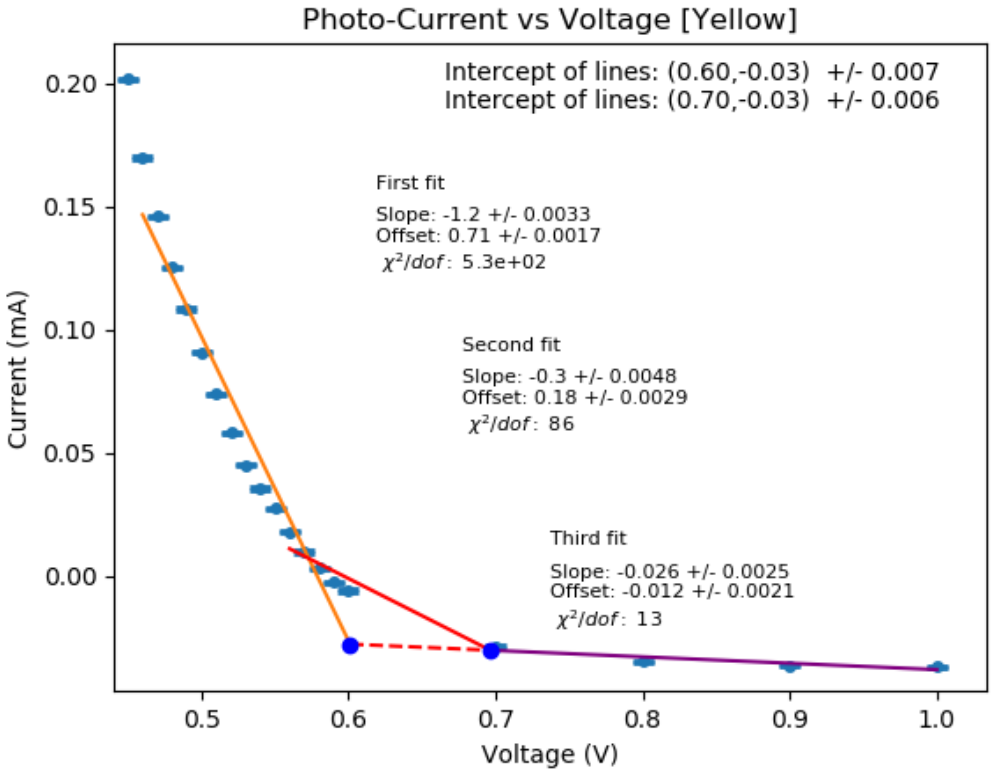
Photo-Current vs Voltage [Yellow]
Intercept of lines: (0.60,-0.03) +/- 0.007
Intercept of lines: (0.70,-0.03) +/- 0.006
First fit
Slope: -1.2 +/- 0.0033
Offset: 0.71 +/- 0.0017
χ^2/dof: 5.3e+02
Second fit
Slope: -0.3 +/- 0.0048
Offset: 0.18 +/- 0.0029
χ^2/dof: 86
Third fit
Slope: -0.026 +/- 0.0025
Offset: -0.012 +/- 0.0021
χ^2/dof: 13
Current (mA)
Voltage (V)
0.20
0.15
0.10
0.05
0.00
0.5
0.6
0.7
0.8
0.9
1.0

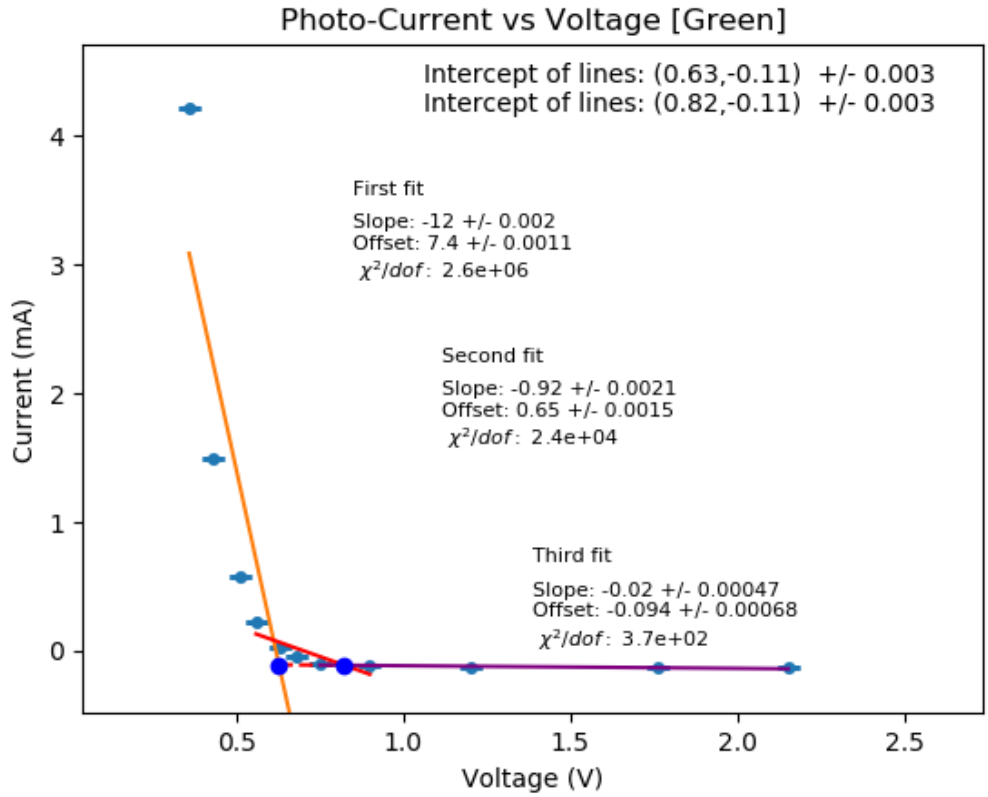
Photo-Current vs Voltage [Green]
Intercept of lines: (0.63,-0.11) +/- 0.003
Intercept of lines: (0.82,-0.11) +/- 0.003
First fit
Slope: -12 +/- 0.002
Offset: 7.4 +/- 0.0011
χ^2/dof: 2.6e+06
Second fit
Slope: -0.92 +/- 0.0021
Offset: 0.65 +/- 0.0015
χ^2/dof: 2.4e+04
Third fit
Slope: -0.02 +/- 0.00047
Offset: -0.094 +/- 0.00068
χ^2/dof: 3.7e+02
Current (mA)
Voltage (V)
4
3
2
1
0
0.5
1.0
1.5
2.0
2.5

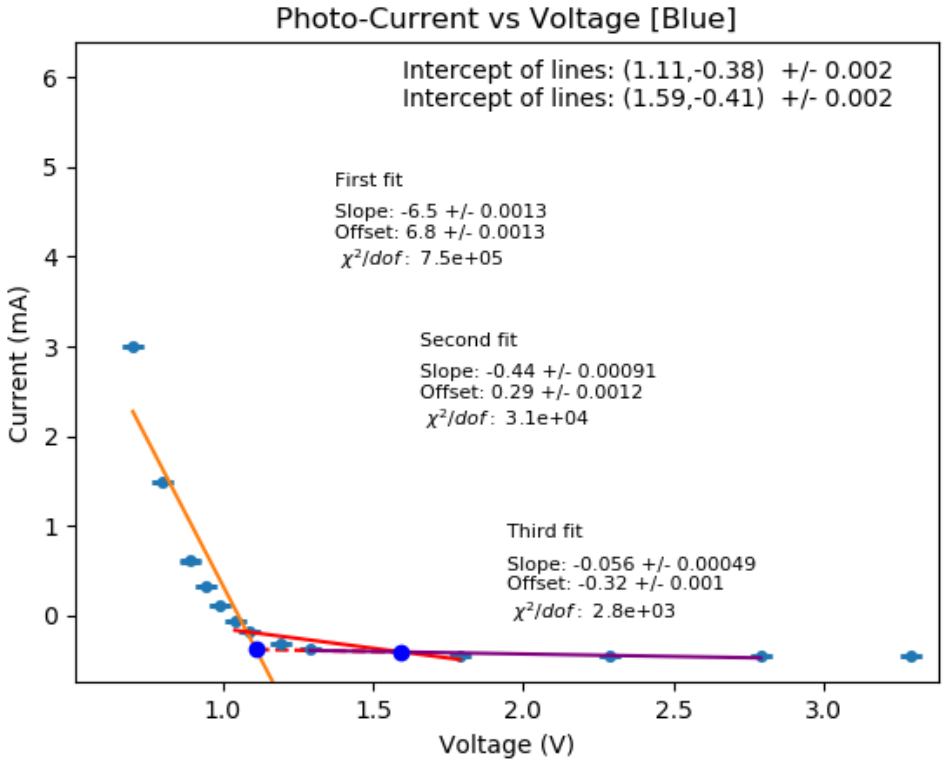
Photo-Current vs Voltage [Blue]
Intercept of lines: (1.11,-0.38) +/- 0.002
Intercept of lines: (1.59,-0.41) +/- 0.002
First fit
Slope: -6.5 +/- 0.0013
Offset: 6.8 +/- 0.0013
χ^2/dof: 7.5e+05
Second fit
Slope: -0.44 +/- 0.00091
Offset: 0.29 +/- 0.0012
χ^2/dof: 3.1e+04
Third fit
Slope: -0.056 +/- 0.00049
Offset: -0.32 +/- 0.001
χ^2/dof: 2.8e+03
Current (mA)
Voltage (V)

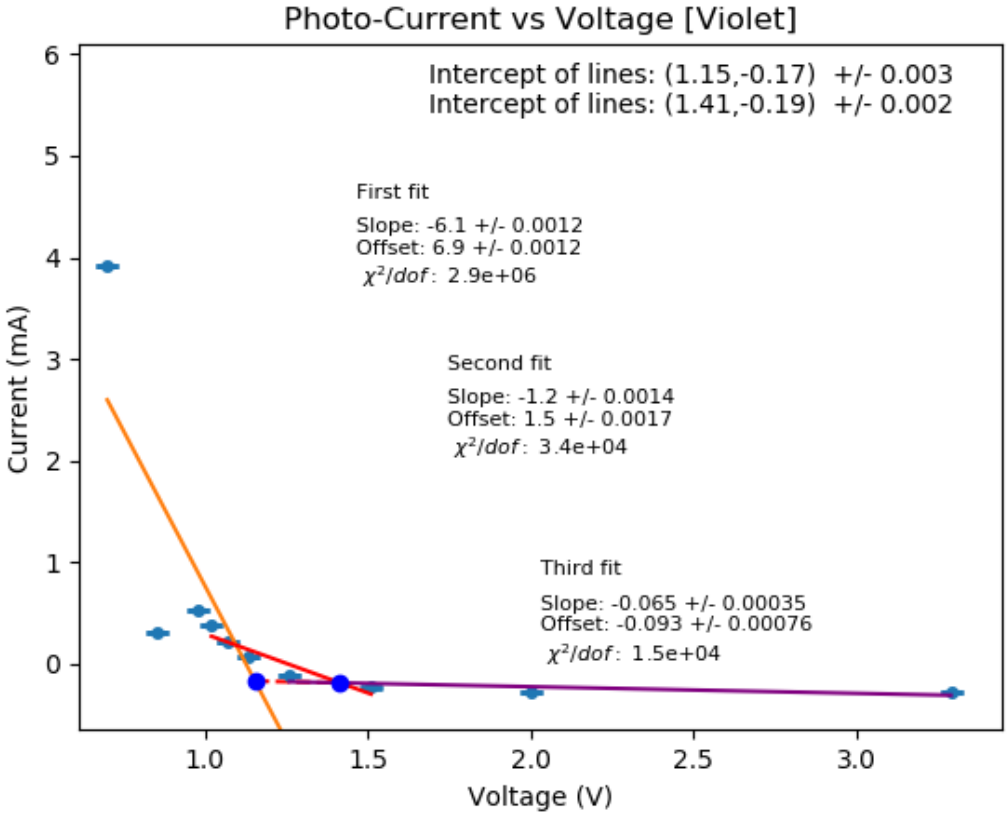
Photo-Current vs Voltage [Violet]
Intercept of lines: (1.15,-0.17) +/- 0.003
Intercept of lines: (1.41,-0.19) +/- 0.002
First fit
Slope: -6.1 +/- 0.0012
Offset: 6.9 +/- 0.0012
χ^2/dof: 2.9e+06
Second fit
Slope: -1.2 +/- 0.0014
Offset: 1.5 +/- 0.0017
χ^2/dof: 3.4e+04
Third fit
Slope: -0.065 +/- 0.00035
Offset: -0.093 +/- 0.00076
χ^2/dof: 1.5e+04
Current (mA)
Voltage (V)

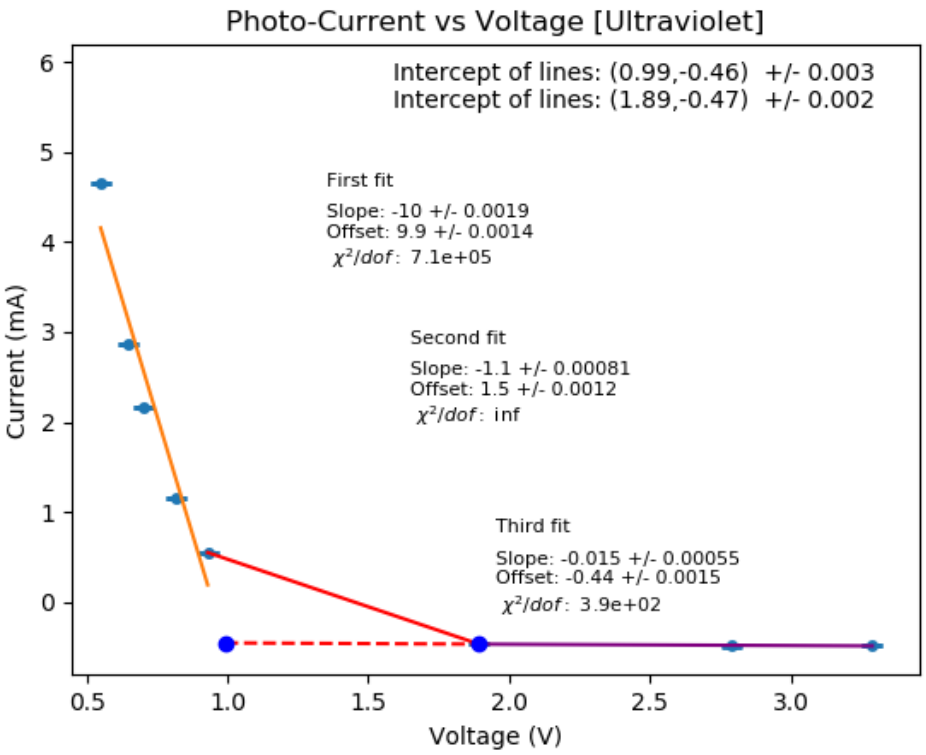
Photo-Current vs Voltage [Ultraviolet]
Intercept of lines: (0.99,-0.46) +/- 0.003
Intercept of lines: (1.89,-0.47) +/- 0.002
First fit
Slope: -10 +/- 0.0019
Offset: 9.9 +/- 0.0014
χ^2/dof: 7.1e+05
Second fit
Slope: -1.1 +/- 0.00081
Offset: 1.5 +/- 0.0012
χ^2/dof: inf
Third fit
Slope: -0.015 +/- 0.00055
Offset: -0.44 +/- 0.0015
χ^2/dof: 3.9e+02
Current (mA)
Voltage (V)

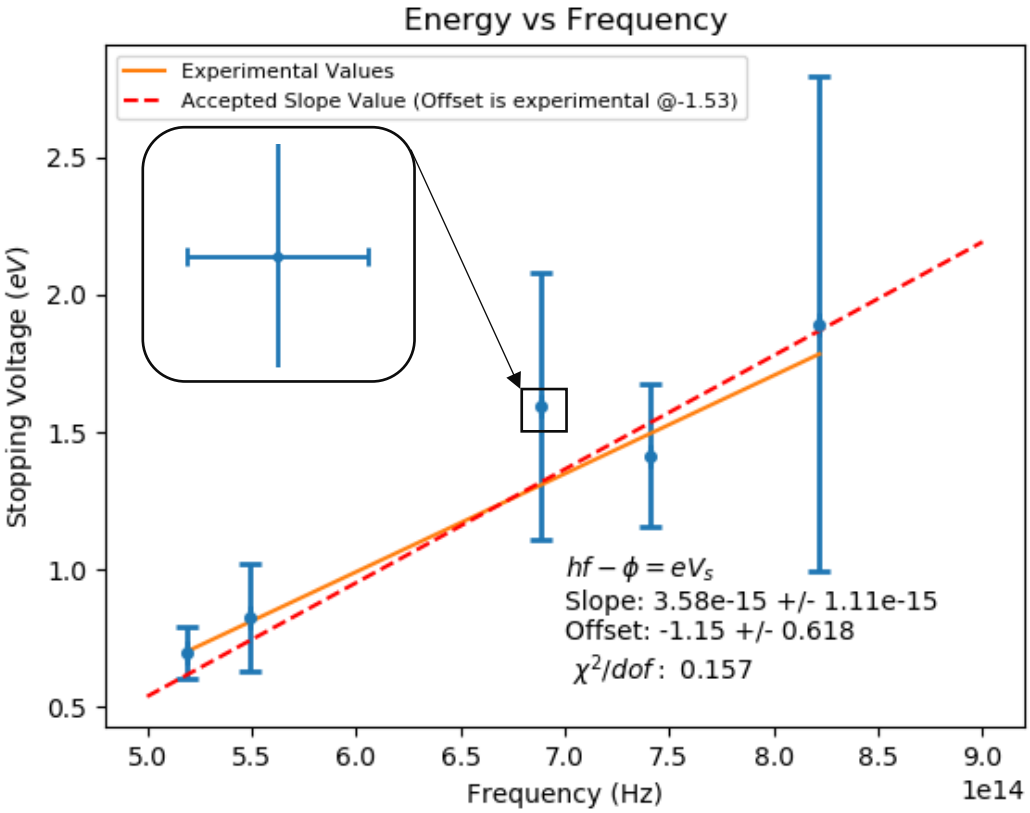
Energy vs Frequency
Experimental Values
Accepted Slope Value (Offset is experimental @-1.53)
$hf - \phi = eV_s$
Slope: 3.58e-15 +/- 1.11e-15
Offset: -1.15 +/- 0.618
χ^2/dof: 0.157
Stopping Voltage (eV)
Frequency (Hz)
1e14

Photo-Current vs Voltage Discussion

Photo-Current vs Voltage data is zoomed in to area of interest; in some of the data you can see a dashed red line, this represents the range of possible V_s. At the top right of each graph you can see the interception points.

The ± of the intercept was calculated using a quadrature of $\sigma_{x_{int}}\ and\ \sigma_{y_{int}}$ which were calculated by first calculating the intercepts using

$$x_{int} = \frac{b_2 - b_1}{m_1 - m_2} \text{ and } y_{int} = mx_{int} + b$$

Then calculating the partials and using the line fit slope and offset uncertainties. (See after conclusion) These uncertainties were excluded from the final fit because they were insignificant.

Energy vs Frequency Discussion

The red dashed line is **only** for visualization of the accepted Planck's constant. Planck's constant in eV is 4.135667696e-15 eV*s the dashed line has a slope of said constant. It is offset to an *arbitrary* -1.53eV in order to lie in the middle of the experimental value fit. You can visually see the difference in slope.

The slope we obtained from the Energy vs Frequency graph was **3.6e-15 eV*s** with an uncertainty of **1.1e-15eV*s**. This results in a difference of 13.8% Which falls within our 30% uncertainty and allows us to show consistently in the relation. The reason we got such a high uncertainty is primarily due to the fact we chose x_2 as our V_s which means +- x_2-x_1 is our uncertainty. If we had better window ranges, and more data points we would've selected the mean of x_2-x_1 as our V_s instead. Currently, we get unrealistic results with that method. So, I opted for higher uncertainty.

For our work function we have a value of **1.2eV** with an uncertainty of **0.62eV**

On the zoomed in image we can see σ_λ

We do not know the accepted value of the work function ϕ so we can not compare it, however, comparing it (1.15eV) to several different metals, it

looks quite low but, it lies in the same order of magnitude (eVs) Cesium for example is ~2.1 according to online sources.[1]

To find the threshold frequency, (The minimum frequency to excite electrons) we refer to the equation shown in the introduction,

$$v_0 = \frac{\phi}{h}$$

$$v_0 = \frac{1.2}{3.6 \cdot 10^{-15}} = 3.3 \cdot 10^{14}\ hertz$$

This has an uncertainty given by

$$\sigma_f = \sqrt{\left(\frac{\partial f}{\partial \phi}\sigma_\phi\right)^2 + \left(\frac{\partial f}{\partial h}\sigma_h\right)^2}$$

Again, partials left out due to it being trivial (See after conclusion),

$$\sigma_f = 2.0 \cdot 10^{14}\ hertz$$

Conclusion

$$v_0 = (3.3 \pm 2.0) \cdot 10^{14}\ hertz$$

$$h = (3.6 \pm 1.1) \cdot 10^{-15} eV \cdot s$$

$$\phi = (1.2 \pm 0.62) eV$$

Had we gathered better data, we could've selected better windows and therefore used the mean of the ranges. This would've cut down uncertainties by a significant amount seeing as a large percentage of uncertainty comes

[1] http://hyperphysics.phy-astr.gsu.edu/hbase/Tables/photoelec.html

from V_s. Our value for Planck's constant wasn't too off from the accepted however, with such a large uncertainty it's not very useful. The data in Energy vs Frequency graph had very closely related fits, (Chisq/dof values were low) which represents close relation, I do not consider the experiment a success because the data taken was bad and the uncertainties due to the bad data render the value obtained to be useless.

Partials

$$\frac{\partial x}{\partial b_2} = \frac{-b_1}{m_1 - m_2}$$

$$\frac{\partial x}{\partial b_1} = \frac{b_2}{m_1 - m_2}$$

$$\frac{\partial x}{\partial m_2} = \frac{(-1)(b_2 - b_1)}{(m_1 - m_2)^2}$$

$$\frac{\partial x}{\partial m_1} = \frac{b_2 - b_1}{(m_1 - m_2)^2}$$

$$\sigma_V = \sqrt{\left(\frac{\partial x}{\partial b_2}\sigma_{b2}\right)^2 + \left(\frac{\partial x}{\partial b_1}\sigma_{b1}\right)^2 + \left(\frac{\partial x}{\partial m_2}\sigma_{m2}\right)^2 + \left(\frac{\partial x}{\partial m_1}\sigma_{m1}\right)^2}$$

σ_{b_i} =fit.sigma_o σ_{m_i} = fit.sigma_s

$$\frac{\partial f}{\partial h} = \frac{-\phi}{h^2}$$

$$\frac{\partial f}{\partial \phi} = \frac{1}{h}$$

END OF REPORT

THE FOLLOWING VALUES ARE PRIOR TO CONSIDERING SIGNIFICANT FIGURES THEY SHOULD BE MODIFIED BEFORE USE

They are in order respective to appearance

Yellow

chisq/dof = 532.6202161474868

offset = 0.7145450000000064 +/- 0.0017478243278294325

slope = -1.2345000000000148 +/- 0.0032868786756695967

chisq/dof = 85.7514939605851

offset = 0.1801615384615282 +/- 0.0029029880715547673

slope = -0.3017692307692137 +/- 0.004823819106188524

chisq/dof = 12.507438016528887

offset = -0.011960000000000193 +/- 0.0021087318463948877

slope = -0.02589999999999959 +/- 0.002459674775249766

##########

Green

chisq/dof = 2555802.064883897

offset = 7.365175647787197 +/- 0.0011009769795786036

slope = -11.889291446915834 +/- 0.0020400709484944568

chisq/dof = 24206.496990648677

offset = 0.6471830065359492 +/- 0.0015124591497751494

slope = -0.9195781342840187 +/- 0.0021197792461215372

chisq/dof = 370.5297677200354

offset = -0.09369578200821227 +/- 0.000676328134986225

slope = -0.020491285496884504 +/- 0.00046598792261221087

##########

Blue

chisq/dof = 745699.696875461

offset = 6.804145381983918 +/- 0.0012691405144356373

slope = -6.471094640820862 +/- 0.00131325146810199

chisq/dof = 30624.452460313845

offset = 0.2899911602209917 +/- 0.0011956608935545592

slope = -0.43624309392265026 +/- 0.0009141309310574824

chisq/dof = 2764.7933884297527

offset = -0.31641800000000053 +/- 0.0010405441845496035

slope = -0.05580000000000007 +/- 0.0004919349550499536

##########

Violet

chisq/dof = 2867926.7817990757

offset = 6.866068015224158 +/- 0.001240943155028403

slope = -6.09849873132227 +/- 0.00122166215279121

chisq/dof = 34034.58413804612

offset = 1.4565218351962548 +/- 0.0016976010038236298

slope = -1.1612035352222494 +/- 0.0014020761463435446

chisq/dof = 15223.741219703745

offset = -0.09347217348729053 +/- 0.0007594434847600802

slope = -0.06539842506834226 +/- 0.00035131758895447354

##########

Ultraviolet

chisq/dof = 714098.7262561432

offset = 9.902324829157244 +/- 0.0013771407026177066

slope = -10.445102505694848 +/- 0.0018561600944227805

chisq/dof = inf

offset = 1.5312812500000015 +/- 0.0012068024156060104

slope = -1.0572916666666679 +/- 0.0008102265201095862

chisq/dof = 386.88632258770593

offset = -0.44245099337748606 +/- 0.0014905375658128628

slope = -0.014635761589403717 +/- 0.0005481757827027032

##

Energy vs Frequency

chisq/dof = 0.15661761980167843

offset = -1.1542068846474223 +/- 0.6177659430743697

slope = 3.575255344595911e-15 +/- 1.1098619326879356e-15

Code

getData.py module you will need to run actual code. You will need to install sympy module

```
import LT.box as B
import numpy as np
from sympy import *


def makeDict(names):
    Data = {}
    for k in range(len(names)):
        Data[names[k]] = dict(dataFile=B.get_file(names[k] + '.data'))
        Data[names[k]]['Parameters'] = {}
        # file.par.get_all_data() creates a dictionary but it stores it all it's
contents in strings.
        for x in Data[names[k]]['dataFile'].par.get_variable_names():
            Data[names[k]]['Parameters'][x] =
Data[names[k]]['dataFile'].par.get_value(x)
        for j in range(len(Data[names[k]]['dataFile'].get_keys()) - 1):
            Data[names[k]][Data[names[k]]['dataFile'].get_keys()[j]] = \
                B.get_data(Data[names[k]]['dataFile'],
Data[names[k]]['dataFile'].get_keys()[j])
    if len(names) == 1: Data = Data[names[0]];
    return Data
```

```
def wmean(x, sig):
    w = 1. / sig ** 2
    # weighted mean
    wm = np.sum(x * w) / np.sum(w)
    sig_wm = np.sqrt(1. / np.sum(w))
    return wm, sig_wm


def labels(xlabel='', ylabel='', title='', annotate='', fit=None, xy=(0.5, 0.1),
xycoords='axes fraction', sig=.4):
    if xlabel: B.pl.xlabel(xlabel);
    if ylabel: B.pl.ylabel(ylabel);
    if title: B.pl.title(title);
    if annotate and fit:
        # noinspection PyStringFormat
        B.pl.annotate(f'%s \nSlope: %{sig}g +/- %{sig}g \nOffset: %{sig}g +/-
%{sig}g\n $\chi^2/dof: $ %{sig}g' % (
            annotate, fit.slope, fit.sigma_s, fit.offset, fit.sigma_o, fit.chi_red),
xy=xy, xycoords=xycoords)


def quadrature(*args):
    args2 = (0, 0,
             0) + args  # bug fix, if you don't add a tuple of zeros the next
statement adds all the numbers together and doesn't create seperate arrays
    args = np.asarray(args2)
    return np.sqrt(np.sum(args ** 2))


def partials(expression, variables, **values):
    partials = {}
    var = variables.split()
    for k in var:
        locals()[k] = Symbol(k)
    expression = eval(expression)
    for k in expression.free_symbols:
        partials[k] = {}
        partials[k]['Partial'] = diff(expression, k)
    for k in partials:
        llist = [x for x in partials[k]['Partial'].free_symbols]
        partials[k]['Lambdify'] = lambdify(llist, partials[k]['Partial'])
        templist = [i for i, j in zip([str(x) for x in
partials[k]['Partial'].free_symbols], values)]
        results = [values[x] for x in templist]
        partials[k]['Evaluated'] = partials[k]['Lambdify'](*results)
    return partials
```

PEF.py actual code

```
from getData import *

colors = ['Yellow', 'Green', 'Blue', 'Violet', 'Ultraviolet']
## Windows ranges is just the domain for the 3 individual line fits, its in order
according to color and line.
```

```
windowRanges = [[0.46, 0.6, .56, .7, .7, 1.1], [0.35, 0.7, 0.54, 0.9, 0.74, 2.2],
[0.69, 1.22, 1.0, 1.9, 1.27, 3],
                [0.68, 1.3, 1.0, 1.55, 1.24, 3.5],
                [0.5, 1.0, 0.9, 2.0, 1.5, 3.5]]

Data = makeDict(colors)

# Graphing data for each color
for k in range(len(colors)):
    dataV = Data[colors[k]]['V']
    dataI = Data[colors[k]]['I'] - Data[colors[k]]['Parameters']['ZeroAdjust']
    datadI = Data[colors[k]]['dI'] + Data[colors[k]]['Parameters']['ZeroAdjustSig']

    B.pl.figure(colors[k] + ' Graph')
    B.plot_exp(dataV, dataI, datadI)
    B.pl.xlabel('Voltage (V)')
    B.pl.ylabel('Current (mA)')
    B.pl.title(f'Photo-Current vs Voltage [{colors[k]}]')

    # Getting the ranges and graphing lines for them
    print('##########################################################################')
    print(colors[k])
    tV = B.in_window(windowRanges[k][0], windowRanges[k][1], dataV)
    tV2 = B.linefit(dataV[tV], dataI[tV], datadI[tV])
    B.plot_line(tV2.xpl, tV2.ypl, zorder=3)

    tV3 = B.in_window(windowRanges[k][2], windowRanges[k][3], dataV)
    tV4 = B.linefit(dataV[tV3], dataI[tV3], datadI[tV3])
    B.plot_line(tV4.xpl, tV4.ypl, color='red', zorder=4)

    tV5 = B.in_window(windowRanges[k][4], windowRanges[k][5], dataV)
    tV6 = B.linefit(dataV[tV5], dataI[tV5], datadI[tV5])
    B.plot_line(tV6.xpl, tV6.ypl, color='purple', zorder=5)

    # Finding the x and y intercept by using basic y=mx+b manipulation, storing and
then plotting them.
    xintercept = (tV6.offset - tV2.offset) / (tV2.slope - tV6.slope)
    yintercept = (tV2.slope * xintercept + tV2.offset)

    xintercept2 = (tV6.offset - tV4.offset) / (tV4.slope - tV6.slope)
    yintercept2 = (tV4.slope * xintercept2 + tV4.offset)

    B.pl.plot(xintercept, yintercept, 'bo', zorder=6)
    B.pl.plot(xintercept2, yintercept2, 'bo', zorder=6)
    B.pl.plot([xintercept,xintercept2],[yintercept,yintercept2],'r--')

    ### Storing data into dictionary
    Data[colors[k]]['xint1'] = xintercept
    Data[colors[k]]['yint1'] = yintercept
    Data[colors[k]]['xint2'] = xintercept2
    Data[colors[k]]['yint2'] = yintercept2

    ##Adding uncertainties of slopes in quadrature
    sigmaint1 = quadrature(tV2.sigma_s,tV4.sigma_s,tV4.sigma_o,tV6.sigma_o)
    sigmaint2 = quadrature(tV4.sigma_s,tV6.sigma_s,tV4.sigma_o,tV6.sigma_o)
```

```
    sigmatotal = quadrature((xintercept2-xintercept))

    B.pl.annotate('Intercept of lines: (%.2f,%.2f)  +/- %.3f\nIntercept of lines:
(%.2f,%.2f)  +/- %.3f' % (
        xintercept, yintercept, sigmaint1, xintercept2, yintercept2, sigmaint2),
xy=(.38, .9),
                  xycoords='axes fraction')
    Data[colors[k]]['sigmatotal'] = sigmatotal

###################################  END OF FOR LOOP   ##############################
print('#############################################################')
B.pl.figure('Energy vs Frequency')
wavelengths = np.array([578, 546, 436, 405, 365]) * (1e-9)
sigmawavelength = np.cbrt(wavelengths)
frequency = 3e8 / (wavelengths)
frequencysigma = 3e8 / (sigmawavelength)
## A little bit of list comprehension, you could've done the same thing in a regular
for loop.
B.plot_exp(frequency, [Data[x]['xint2'] for x in colors], [Data[x]['sigmatotal'] for
x in colors],xerr=frequencysigma)

fit = B.linefit(frequency, np.asarray([Data[x]['xint2'] for x in colors]),
               np.asarray([Data[x]['sigmatotal'] for x in colors]))
B.plot_line(fit.xpl, fit.ypl, label='Experimental Values')

################################ PLOTTING ACCEPTED VALUE LINE
############################
x = np.linspace(5e14, 9e14)
y = 4.135667696e-15 * x - 1.53
B.pl.plot(x, y, 'r--', label='Accepted Slope Value (Offset is experimental @-1.53)')
#########################################################################################
#####

B.pl.legend(loc='best', prop={'size': 8})
labels('Frequency (Hz)','Stopping Voltage ($eV$)','Energy vs Frequency','$hf-\phi =
eV_s$',fit,xy=(0.5,0.07),sig=.3)
```

Yellow Data

```
#\ ZeroAdjust = 0.005
#\ ZeroAdjustSig = 0.00005


#Voltage, Current, Current Sigma
#! V[f,0]/  I[f,1]/ dI[f,2]/
3.29    -0.060   0.0005
2.79    -0.060   0.0005
2.29    -0.058   0.0005
1.79    -0.058   0.0005
1.29    -0.056   0.0005
0.99    -0.053   0.0005
0.89    -0.052   0.0005
0.84    -0.052   0.0005
0.81    -0.051   0.0005
```

```
0.79    -0.050   0.0005
0.76    -0.049   0.0005
```

Green Data

```
#\ ZeroAdjust = 0.002
#\ ZeroAdjustSig = 0.00005

#Voltage, Current, Current Sigma
#! V[f,0]/  I[f,1]/ dI[f,2]/
3.29    -0.132  0.0005
1.76    -0.129  0.0005
2.15    -0.131  0.0005
1.20    -0.126  0.0005
0.90    -0.117  0.0005
0.75    -0.094  0.0005
0.68    -0.044  0.0005
0.63    0.033   0.0005
0.56    0.231   0.0005
0.51    0.577   0.0005
0.43    1.497   0.0005
0.36    4.220   0.0005
0.29    6.636   0.0005
0.23    8.379   0.0005
0.17    10.61   0.0005
0.10    13.03   0.0005
0.04    14.66   0.0005
0.01    15.77   0.0005
```

Blue Data

```
#\ ZeroAdjust = 0.005
#\ ZeroAdjustSig = 0.00005

#Voltage, Current, Current Sigma
#! V[f,0]/  I[f,1]/ dI[f,2]/
3.29    -0.452   0.0005
2.79    -0.452   0.0005
2.29    -0.447   0.0005
1.79    -0.441   0.0005
1.29    -0.361   0.0005
1.09    -0.163   0.0005
1.19    -0.299   0.0005
1.04    -0.053   0.0005
0.99    0.113    0.0005
0.94    0.325    0.0005
0.89    0.613    0.0005
0.80    1.495   0.0005
0.70    3.003   0.0005
0.50    9.496   0.0005
0.20    18.544  0.0005
0.10    18.606  0.0005
0.05    18.591  0.0005
```

Violet Data

```
#\ ZeroAdjust = 0.005
#\ ZeroAdjustSig = 0.00005

#Voltage, Current, Current Sigma
#! V[f,0]/  I[f,1]/ dI[f,2]/
3.29    -0.280   0.0005
2.00    -0.271   0.0005
1.51    -0.226   0.0005
1.26    -0.104   0.0005
1.13    0.077  0.0005
1.07    0.224  0.0005
1.02    0.381  0.0005
0.98    0.528  0.0005
0.85    0.321  0.0005
0.70    3.920  0.0005
0.59    6.256  0.0005
0.45    11.111 0.0005
0.35    15.663 0.0005
0.25    18.471 0.0005
0.20    18.471 0.0005
0.15    18.564 0.0005
```

Ultraviolet Data

```
#\ ZeroAdjust = 0.005
#\ ZeroAdjustSig = 0.00005

#Voltage, Current, Current Sigma
#! V[f,0]/  I[f,1]/ dI[f,2]/
3.29    -0.480   0.0005
2.79    -0.487   0.0005
1.89    -0.462   0.0005
0.93    0.553    0.0005
0.82    1.167    0.0005
0.70    2.159    0.0005
0.55    4.653    0.0005
0.65    2.880    0.0005
0.45    7.915    0.0005
0.35    12.281   0.0005
0.20    14.909   0.0005
0.21    17.640   0.0005
0.15    18.418   0.0005
0.10    18.463   0.0005
0.05    18.498   0.0005
```