PHYC30170 Physics with Astronomy and Space Science Lab 1; An Investigation of the Ramsauer-Townsend Effect

Daragh Hollman[*]
(Dated: April 11, 2023)

The aims of this report were to investigate the Ramsauer-Townsend effect and determine the electron energy which yields a minimum probability of scattering. The contact potential and the emission electron energy were also determined to add to the accuracy of the measurement. The minimum probability of scattering was determined to be at electron energies of $(1.1 \pm 0.2)\,\mathrm{eV}$. The contact potential was determined to be $(0.2 \pm 0.01)\,\mathrm{V}$ and the emission electron energy was determined to be $(0.11 \pm 0.03)\,\mathrm{V}$.

I. INTRODUCTION

The Ramsauer-Townsend effect describes the phenomenon where electrons exhibit a minimum scattering cross section around electron energies of 1 eV in noble gases [1]. This interaction between particles cannot be described by a classical interpretation of particle collisions and requires a quantum mechanical description [2]. It is important to understand and experimentally test the Ramsauer-Townsend effect as to better understand how particle collisions work in quantum mechanics.

II. THEORY

Under classical mechanics, scattering particles are described by hard sphere scattering, where particles collide and interact as billiard balls might. Such a model is described in figure 1 where the potential experienced by a light scattered particle is $U(r) = 0$ for $r > R$ and $U(r) = \infty$ for $r < R$ where R is the radius of the sphere it scatters off [3]. It follows that, for a projectile particle moving towards the target but offset by a distance from the axis greater than R, the particle would not be deflected at all. We see there is some region with which the scattered particle cannot pass through, this area is known as the scattering cross section and when with respect to the total area it is analogous to the probability of scattering [3].

When considered in this way, the scattering cross section of the noble gas atoms is independent of the incident electron energy. However, if the noble gas atoms are treated within quantum mechanics and present an attractive potential, akin to a square well, the solution of Schrödinger's equation yields that the scattering cross section will have a minimum for electron energies near 1 eV [4][5]. The scattering of electrons by a square well can be predicted by a one-dimensional model or a three-dimensional model. These models represent the xenon atom as a square well with positive potential and

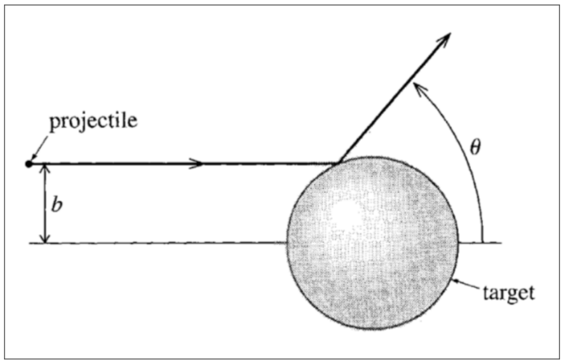


FIG. 1: A simple hard sphere scattering model described by the potential $U(r) = 0$ for $r > R$ and $U(r) = \infty$ for $r < R$

are further described by Kukolich [4].

A thyratron, see figure 2, was used to demonstrate the Ramsauer-Townsend effect. The shield splits the cathode from the plate using two apertures. A beam of electrons starts at the cathode and moves towards the plate. After the first aperture, some electrons are scattered and don't arrive at the plate. In this region, the intensity of the electron beam is given as follows:

$$J = J_0 \exp\left(\frac{-x}{\lambda}\right) \tag{1}$$

where x is the distance from the cathode and λ is the mean free path [4]. The intensity at the plate is hence:

$$J_p = J_0 \exp\left(\frac{-l}{\lambda}\right) = J_0\,(1 - P_s) \tag{2}$$

where l is the distance between the cathode and plate and P_s is the probability of scattering [4]. As the shield current is proportional to the intensity of the electron beam at the first aperture, we can substitute the initial intensity, J_0, to obtain an expression for the plate current:

$$I_p = I_s f(V)(1 - P_s) \tag{3}$$

where I_p and I_s are the plate and shield currents respectively, and $f(V)$ is a geometrical factor which depends on the ratio of angle intercepted by the plate to the angle

[*] daragh.hollman@ucdconnect.ie

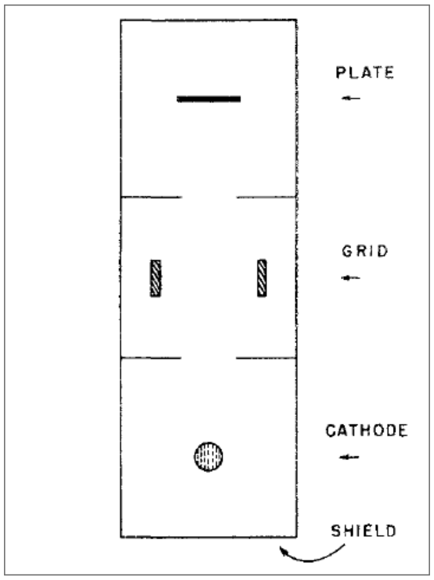


FIG. 2: An example of a thyratron used in this experiment [4].

intercepted by the shield and depends on a factor due to the space charge effects near the cathode [4]. This geometrical factor can be determined by cooling the xenon in the thyratron to reduce the pressure and eliminate scattering. P_s goes to zero and hence:

$$f(V) \approx \frac{I_p^\star}{I_s^\star} \tag{4}$$

where the * notation indicates when cooled. Substituting the geometric factor back into equation 3 we can solve for the probability of scattering.

$$P_s = 1 - \frac{I_p I_s^\star}{I_s I_p^\star} \tag{5}$$

A discrepancy in the data from the expected outcome is described by Kukolich and elaborated on by Woolsey [4] [6]. This discrepancy exists due to the contact potential and the emission energy of the thermionic electrons from the cathode. The contact potential arises from the shield and cathode being made from different metals which have different work functions. As the shield is made of nickel, it has a higher work function than the barium oxide cathode and hence as the two electrodes are in contact through the gas, there is a net flow of electrons to the shield without any potential [6]. To account for this, a negative potential at the shield is required to balance this flow. Hence the electrons leaving the cathode were affected by a total acceleration voltage of $(V + V_c)$ where V_c is the contact potential between the electrodes.

The emission energy of the thermionic electrons must also be accounted for. As the electrons leave the cathode they have a mean initial energy $e\bar{V}$ eV which also must be added to the acceleration voltage. Hence we have a total acceleration voltage of $(V + V_c + \bar{V})$ giving an electron energy of $(V - V_s + V_c + \bar{V})$. Woolsey describes the number of electrons with energy ϵ to be proportional to $\exp\left(\frac{-\epsilon}{kT}\right)$ [6]. Therefore for electrons with mean energy $\frac{3}{2}kT$ we have:

$$N(\epsilon) \propto \exp\left(\frac{-3V}{2\bar{V}}\right) \tag{6}$$

which gives a current at the shield as follows:

$$i = i_0 \exp\left(\frac{-3V_{\text{accel}}}{2\bar{V}}\right) \tag{7}$$

where i_0 is the current when the acceleration voltage, V, is zero.

Therefore, to measure the probability of scattering as a function of electron momentum we must measure the potential across the plate resistor and across the shield resistor to find the plate and shield currents for a range of accelerating voltages and then repeat this when the thyratron is submerged in liquid nitrogen. To correct for the contact potential and the electron emission energy we must take measurements with the voltage source reversed.

III. METHODOLOGY

A. Apparatus Setup

The apparatus was set up as shown in figure 3. It contains a 2D21 thyratron pictured left which was set up such that the xenon tube could be submersed in liquid nitrogen for cooling. A funnel was used as a reservoir to ensure the tube remained submerged and at constant temperature (i.e. constant pressure) for the entire dataset. The thyratron consists of a cathode and a plate at opposite ends of the tube, across which the acceleration voltage is applied. This creates an electric field between the cathode and the plate. A filament at the cathode was heated by a constant 4 V to induce thermionic emission to free the electrons.

B. Procedure

The electronics were switched on in advance of taking the data to ensure the apparatus was not changing in temperature. Data was first collected for the thyratron at room temperature (xenon pressure ≈ 0.05 Torr). The acceleration voltage was varied from 0 V to 15 V starting with steps of 0.1 V and changing to larger steps of 0.5 V after 2 V. This ensured that the expected minimum around 1 eV in the scattering cross section would be well described. For each step, the acceleration voltage, plate

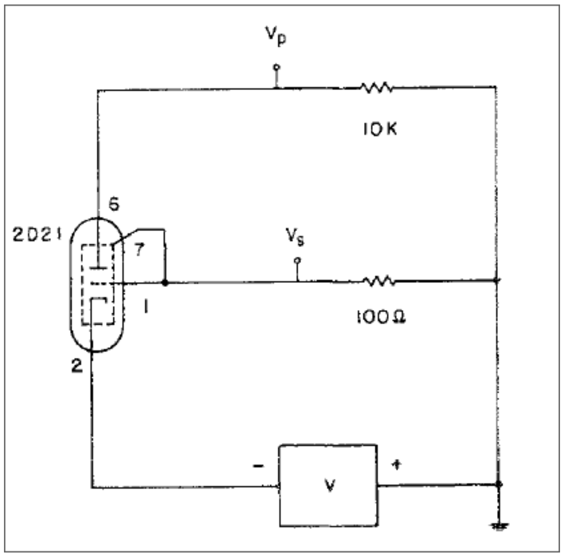


FIG. 3: A circuit diagram of the experiment apparatus from Kukolich [4].

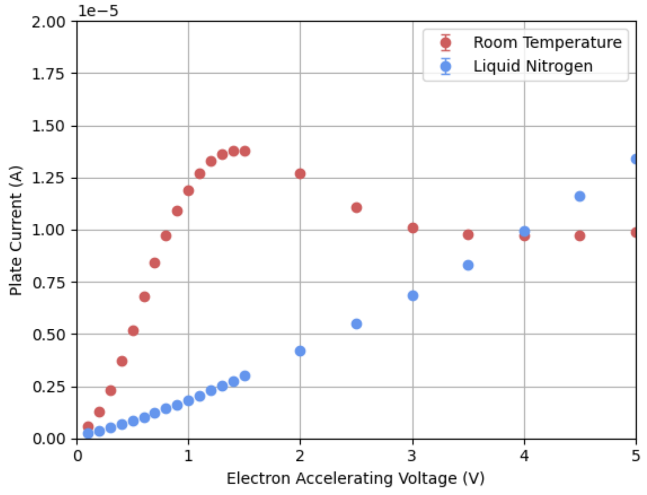


FIG. 4: Plate current against the acceleration voltage acting on the electron for both measurements, room temperature and liquid nitrogen cooled. Here the room temperature currents are multiplied by 10 to see them more easily. Note that error bars are present but are too small to see.

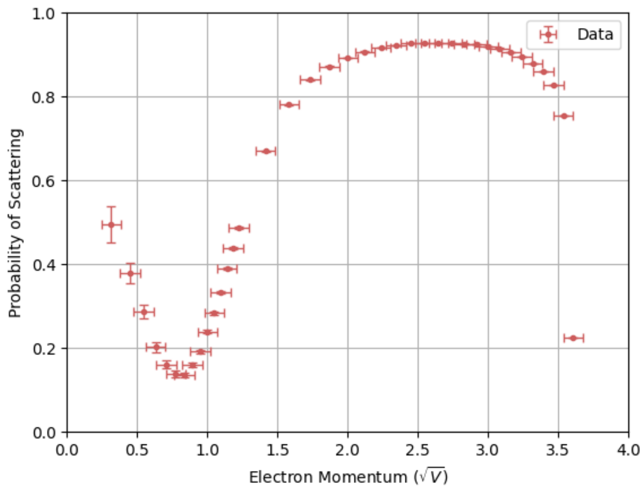


FIG. 5: The probability of scattering against the electron momentum. Note that this is unadjusted for contact potential and electron initial energy.

voltage, and shield voltage were measured and recorded along with their respective uncertainties. The apparatus was then cooled using liquid nitrogen to reduce the pressure and the procedure was repeated.

To account for the contact potential, V_c, and the emission energy of the thermionic electrons, $\bar{V}$, the voltage generator controlling the acceleration voltage was reversed such to drive the circuit with a negative potential to counteract the electron flow because of the contact potential. The shield current was recorded for voltages between 0 V and −0.3 V.

IV. RESULTS & ANALYSIS

The data recorded for the shield and plate voltage were converted to currents using Ohm's law. The plate currents, I_p and $I_p^\star$, were plotted against the acceleration voltage as shown in figure 4. The uncertainties on the voltages were propagated through Ohm's law to obtain uncertainties on the currents. The uncertainty on the voltage was limited by the accuracy of the multimeter. It is clear in this graph that the scattering is nullified when the thyratron is cooled by liquid nitrogen.

Using equation 5, the probability of scattering was determined as a function of electron momentum. This is plotted in figure 5. Once again, the uncertainties here are propagated through the relevant equations to yield the uncertainties plotted on the data. Two distinct minima can be seen in the data. The first of which is just below 1 eV due to the Ramsauer-Townsed effect, and second at the point of ionisation. The minimum probability of scattering due to the Ramsauer-Townsend effect was determined by fitting an inverted Gaussian function in the range of 0.3 V to 1.3 V. This was done through least squares fitting. The minimum point occurred at (0.794 ± 0.002) V with the uncertainty being determined by the square root of the diagonal elements of the covariance matrix of this Gaussian fit.

To calculate the contact potential and the emission energy of the thermionic electrons, the data from the reverse acceleration voltage measurement was plotted in figure 6. $\log I_s^\star$ was plotted against the reverse acceleration voltage. We see a straight line which gets steeper after the potential balances that of the contact potential. This turnover point is a measurement of the contact potential. Two exponential lines were fitted using least squares fitting to each portion of the data to determine

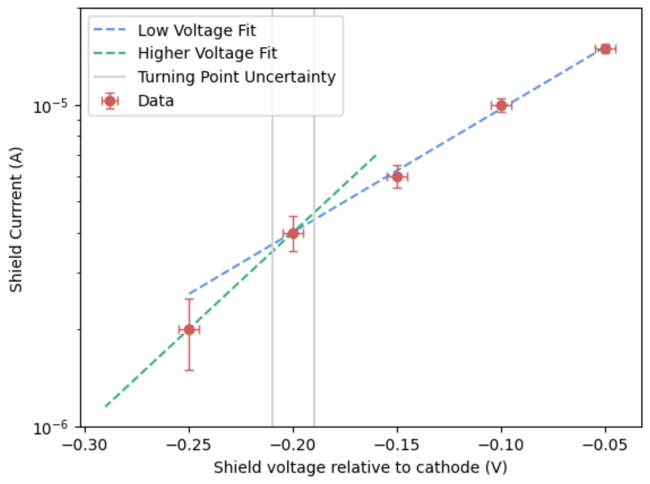


FIG. 6: The shield current against the accelerating voltage in reverse. Note that due the lack of a higher precision ammeter, data below $-0.25\,\mathrm{V}$ could not be measured.

the intersection point. The contact potential was determined to be $(0.2 \pm 0.01)\,\mathrm{V}$, the uncertainty on which was determined by the method of extremes. $\bar{V}$ can also be determined from this graph. Using equation 7, $\bar{V}$ was determined to be $(0.11 \pm 0.03)\,\mathrm{V}$. The calculation of both of these values from the graphs is included in full in appendix A. From these values, a final determination of the electron energy yielding a minimum probability of scattering is at $(1.05 \pm 0.2)\,\sqrt{\mathrm{V}}$.

V. CONCLUSION

The aims of this report were to investigate the Ramsauer-Townsend effect and determine the electron energy which yields a minimum probability of scattering. The contact potential and the emission electron energy were also determined to add to the accuracy of the measurement. The minimum probability of scattering was determined to be at electron energies of $(1.1 \pm 0.2)\,\sqrt{\mathrm{V}}$. This agrees well with the analytical findings of Vahedi [7] and Kukolich [4] that the electron energy should be around $1\,\mathrm{eV}$. The contact potential was determined to be $(0.2 \pm 0.01)\,\mathrm{V}$ and the emission electron energy was determined to be $(0.11 \pm 0.03)\,\mathrm{V}$.

[1] Ramsauer-Townsend Effect, University of Wisconsin `https://www.physics.wisc.edu/courses/home/spring2018/407/experiments/ramsauer/ramsauer.pdf`, accessed on 07/04/23.

[2] Quantum Scattering and a Determination of The Scattering Cross-Section of Xenon Gas in a 2D21 Thyratron Tube, University of Texas `http://wwwrel.ph.utexas.edu/Public/Students/rcrane/quantum/Ramsauer_Townsend/Ramsauer.htm`, accessed on 07/04/23.

[3] Scattering, UC Santa Barbara `https://web.physics.ucsb.edu/~fratus/phys103/LN/Scattering.pdf`, accessed on 07/04/23.

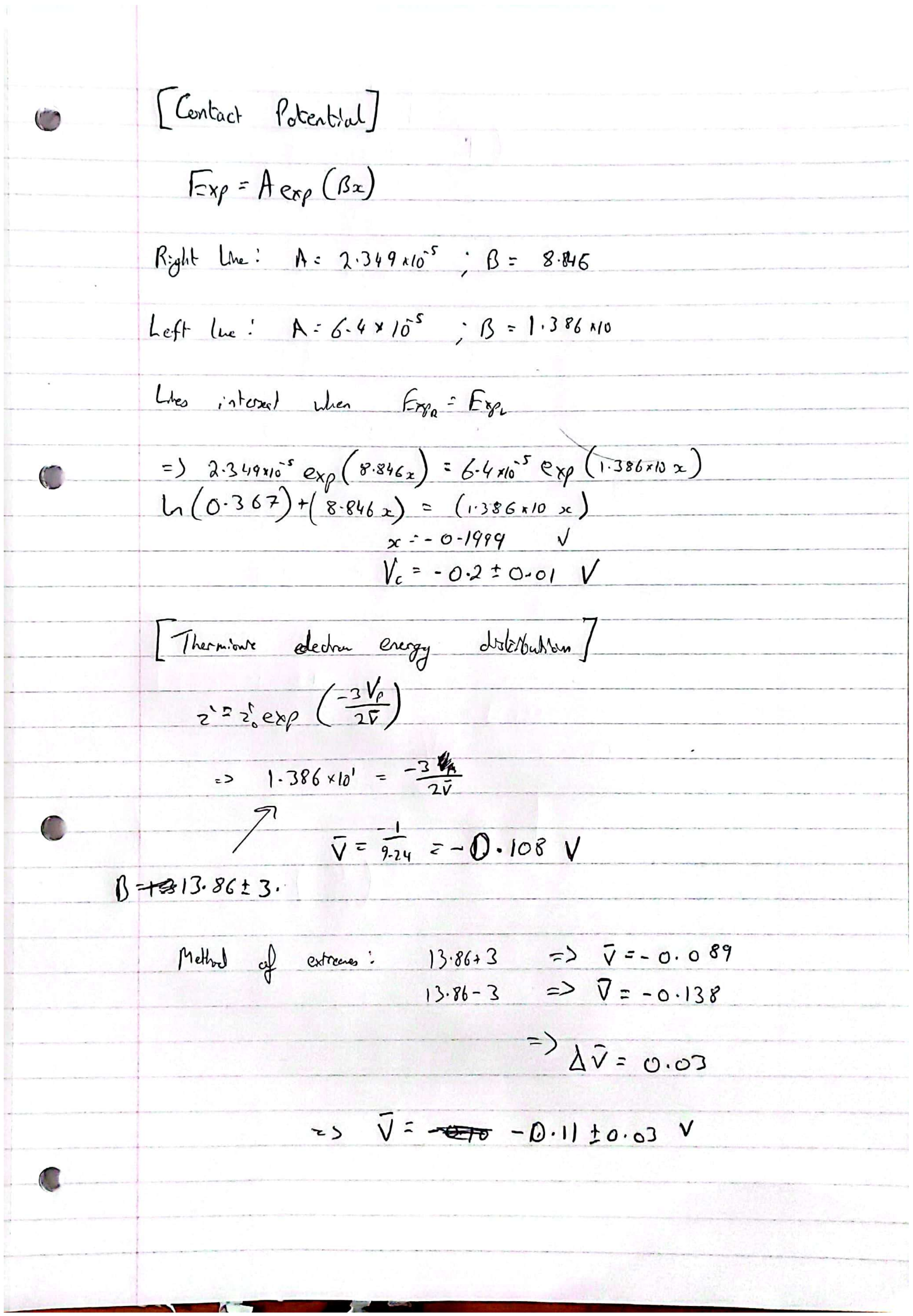
[4] S. G. Kukolich, Demonstration of the ramsauer-townsend effect in a xenon thyratron, American journal of physics **36**, 701 (1968).

[5] H. Sobhani, H. Hassanabadi, and W. S. Chung, Observations of the ramsauer–townsend effect in quaternionic quantum mechanics, The European physical journal. C, Particles and fields **77**, 1 (2017).

[6] G. A. Woolsey, An extension of the ramsauer-townsend experiment in a xenon thyratron, American journal of physics **39**, 558 (1971).

[7] J. Vahedi and K. Nozari, The ramsauer-townsend effect in the presence of the minimal length and maximal momentum, Acta physica Polonica, A **122**, 38 (2012).

Appendix A: Calculation of V_c and $\bar{V}$



[Contact Potential]

$$Exp = A \exp(Bx)$$

Right Line: $A = 2.349\times10^{-5}$; $B = 8.846$

Left line: $A = 6.4\times10^{-5}$; $B = 1.386\times10$

Lines intersect when $Exp_R = Exp_L$

$$\Rightarrow 2.349\times10^{-5} \exp(8.846x) = 6.4\times10^{-5} \exp(1.386\times10\, x)$$

$$\ln(0.367) + (8.846\, x) = (1.386\times10\, x)$$

$$x = -0.1999 \quad ✓$$

$$V_c = -0.2 \pm 0.01 \text{ V}$$

[Thermionic electron energy distribution]

$$z' = z'_0 \exp\left(\frac{-3V_p}{2\bar{V}}\right)$$

$$\Rightarrow 1.386\times10^1 = \frac{-3\cancel{V_p}}{2\bar{V}}$$

$$\bar{V} = \frac{-1}{9.24} = -0.108 \text{ V}$$

$B = ~~+13~~ 13.86 \pm 3.$

Method of extremes: $13.86+3 \Rightarrow \bar{V} = -0.089$

$13.86-3 \Rightarrow \bar{V} = -0.138$

$$\Rightarrow \Delta\bar{V} = 0.03$$

$\Rightarrow \bar{V} =$ ~~0.10~~ -0.11 ± 0.03 V

CS Scanned with CamScanner

ramsauerCode

April 10, 2023

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Change default colours to personal colour scheme
import matplotlib as mpl
mpl.rcParams['axes.prop_cycle'] = mpl.cycler(color=["indianred",␣
 ↪"cornflowerblue", "mediumseagreen", "plum", "sandybrown"])

rootPath = r"/home/daraghhollman/Main/UCD_PASS_Labs/RamsauerTownsend/Data"

def LoadFile(path, skiprows=2):
    data = np.array(np.loadtxt(path, skiprows=skiprows))
    return data
```

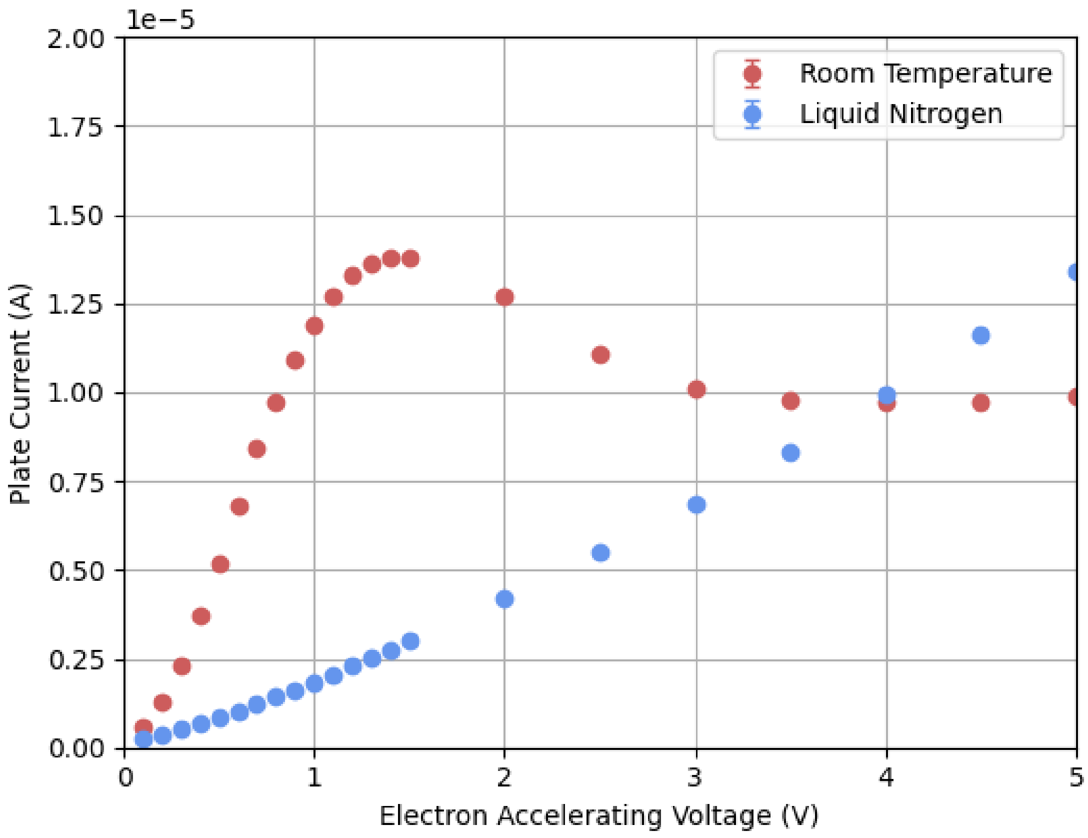
[ ]:
```
def Current(voltage, resistance):
    return voltage / resistance
```

[ ]:
```
def ExtractData(data, uncertainnty=False):
    inputVoltage = data[:,0]
    plateCurrent = [Current(el/1000, 10000) for el in data[:,1]] # note␣
 ↪conversion from milivolts to volts
    shieldCurrent = [Current(el/1000, 100) for el in data[:,2]]

    if uncertainnty:
        plateCurrentUncertainty = [Current(0.05/1000, 10000) for el in data[:
 ↪,3]]
        return [inputVoltage, plateCurrent, shieldCurrent,␣
 ↪plateCurrentUncertainty]
    else:
        return [inputVoltage, plateCurrent, shieldCurrent]
```

[ ]:
```
warmData = ExtractData(LoadFile(rootPath + r"/warmDataLess.txt"),␣
 ↪uncertainnty=True)

coldData = ExtractData(LoadFile(rootPath + r"/coldData.txt"),␣
 ↪uncertainnty=False)
```

```
plt.errorbar(warmData[0], [el*10 for el in warmData[1]], yerr=warmData[3],␣
 ↪fmt="o", capsize=3, linewidth=1, label="Room Temperature")
plt.errorbar(coldData[0], coldData[1], yerr=Current(0.05/1000, 10000), fmt="o",␣
 ↪capsize=3, linewidth=1, label="Liquid Nitrogen")

plt.legend()
plt.grid()
plt.xlim(0, 5)
plt.ylim(0, 2e-5)
plt.xlabel("Electron Accelerating Voltage (V)")
plt.ylabel("Plate Current (A)")
```

```
[ ]: Text(0, 0.5, 'Plate Current (A)')
```



```
[ ]: def ProbabilityOfScattering(warmIp, warmIs, coldIp, coldIs):
         return 1 - warmIp * coldIs / (warmIs * coldIp)

     def dPdWIp(warmIp, warmIs, coldIp, coldIs):
         return - coldIs / (warmIs * coldIp)
```

```
def dPdWIs(warmIp, warmIs, coldIp, coldIs):
    return warmIp * coldIs / (warmIs**2 * coldIp)

def dPdCIp(warmIp, warmIs, coldIp, coldIs):
    return warmIp * coldIs / (warmIs * coldIp**2)

def dPdCIs(warmIp, warmIs, coldIp, coldIs):
    return - warmIp / (warmIs * coldIp)

def ProbabilityOfScatteringUncertainty(warmIp, warmIs, coldIp, coldIs, uWarmIp,␣
 ↪uWarmIs, uColdIp, uColdIs):

    warmIpContribution = dPdWIp(warmIp, warmIs, coldIp, coldIs)**2 * uWarmIp**2
    warmIsContribution = dPdWIs(warmIp, warmIs, coldIp, coldIs)**2 * uWarmIs**2
    coldIpContribution = dPdCIp(warmIp, warmIs, coldIp, coldIs)**2 * uColdIp**2
    coldIsContribution = dPdCIs(warmIp, warmIs, coldIp, coldIs)**2 * uColdIs**2

    return np.sqrt(warmIpContribution + warmIsContribution + coldIpContribution␣
 ↪+ coldIsContribution)

def dIdV(R):
    return 1/R

def CurrentUncertainty(uV, R):
    return np.sqrt(dIdV(R)**2 * uV**2)
```

[ ]:
```
plateVoltageLists = [warmData[1], coldData[1]]
shieldVoltageLists = [warmData[2], coldData[2]]

plateCurrentLists = []
for list in plateVoltageLists:
    newCurrentList = []
    for voltage in list:
        newCurrentList.append(Current(voltage, 10000))
    plateCurrentLists.append(newCurrentList)

shieldCurrentLists = []
for list in shieldVoltageLists:
    newCurrentList = []
    for voltage in list:
        newCurrentList.append(Current(voltage, 100))
    shieldCurrentLists.append(newCurrentList)

probabilityOfScattering = [ProbabilityOfScattering(a, b, c, d) \
    for a, b, c, d in zip(plateCurrentLists[0], shieldCurrentLists[0],␣
 ↪plateCurrentLists[1], shieldCurrentLists[1])]
```

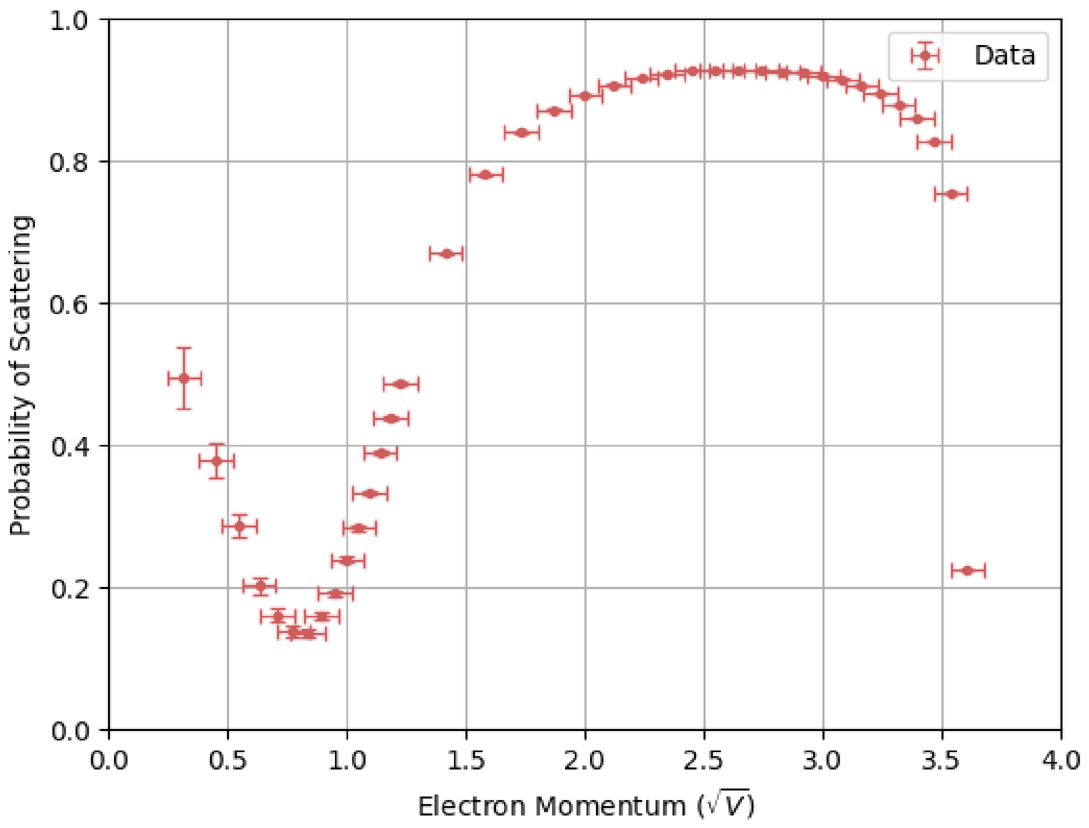
```
warmShieldVoltageError = []
for i in range(len(warmData[2])):
    if warmData[2][i] < 200/1000:
        warmShieldVoltageError.append(0.05/1000)
    elif warmData[2][i] > 200/1000:
        warmShieldVoltageError.append(0.5/1000)

coldShieldVoltageError = []
for i in range(len(coldData[2])):
    if coldData[2][i] < 200:
        coldShieldVoltageError.append(0.05/1000)
    elif coldData[2][i] > 200:
        coldShieldVoltageError.append(0.5/1000)

probabilityOfScatteringUncertainty = []
for warmIp, warmIs, coldIp, coldIs, uWarmIp, uWarmIs, uColdIp, uColdIs in␣
 ↪zip(plateCurrentLists[0], \
    shieldCurrentLists[0], plateCurrentLists[1], shieldCurrentLists[1], \
        [CurrentUncertainty(0.05/1000, 10000)]*len(plateCurrentLists[0]), \
        [CurrentUncertainty(uV, 100) for uV in warmShieldVoltageError], \
            [CurrentUncertainty(0.05/1000, 10000)]*len(plateCurrentLists[0]), \
            [CurrentUncertainty(uV, 100) for uV in coldShieldVoltageError]):

    probabilityOfScatteringUncertainty.append(\
        ProbabilityOfScatteringUncertainty(warmIp, warmIs, coldIp, coldIs,␣
 ↪uWarmIp, uWarmIs, uColdIp, uColdIs))

print(probabilityOfScatteringUncertainty)
probabilityOfScatteringUncertainty = [el/10000 for el in␣
 ↪probabilityOfScatteringUncertainty]
```

```
[431.9289149848619, 252.5157400027043, 169.58129208652815, 122.83559315840829,
94.68138126811247, 76.01240056729162, 62.505310648180185, 52.46985201923111,
44.74463634535257, 38.095797486444965, 33.17852604257133, 29.025188399706074,
25.528922932959098, 22.754087941288464, 20.468115656736533, 13.547138748092992,
10.09115224738238, 8.000144895217508, 6.577475278420045, 5.57813835677166,
4.807737777535432, 4.212666880320136, 3.7342773839934065, 3.3660041689175566,
3.086644132642144, 2.849764657974558, 2.641193438687383, 2.4296813283876197,
2.2496255794021005, 2.0842950651428516, 1.936576219360492, 1.827656714480186,
1.7453829559190897, 1.6776639942688392, 1.5829601627239867, 1.4542950453929377,
1.2765219351608958, 1.1894341574327283, 1.451619884551446, 1.6440654576782265,
1.2261232482605633, 0.9450554210838951]
```

[ ]:
```
electronMomentum = [np.sqrt(el) for el in warmData[0]]
```

```
plt.errorbar(electronMomentum, probabilityOfScattering, xerr=np.sqrt(0.005),␣
↪yerr=probabilityOfScatteringUncertainty, fmt=".", capsize=3, linewidth=1,␣
↪label="Data")

plt.legend()
plt.grid()
plt.xlim(0, 4)
plt.ylim(0, 1)
plt.xlabel("Electron Momentum ($\sqrt{V}$)")
plt.ylabel("Probability of Scattering")
```

```
[ ]: Text(0, 0.5, 'Probability of Scattering')
```

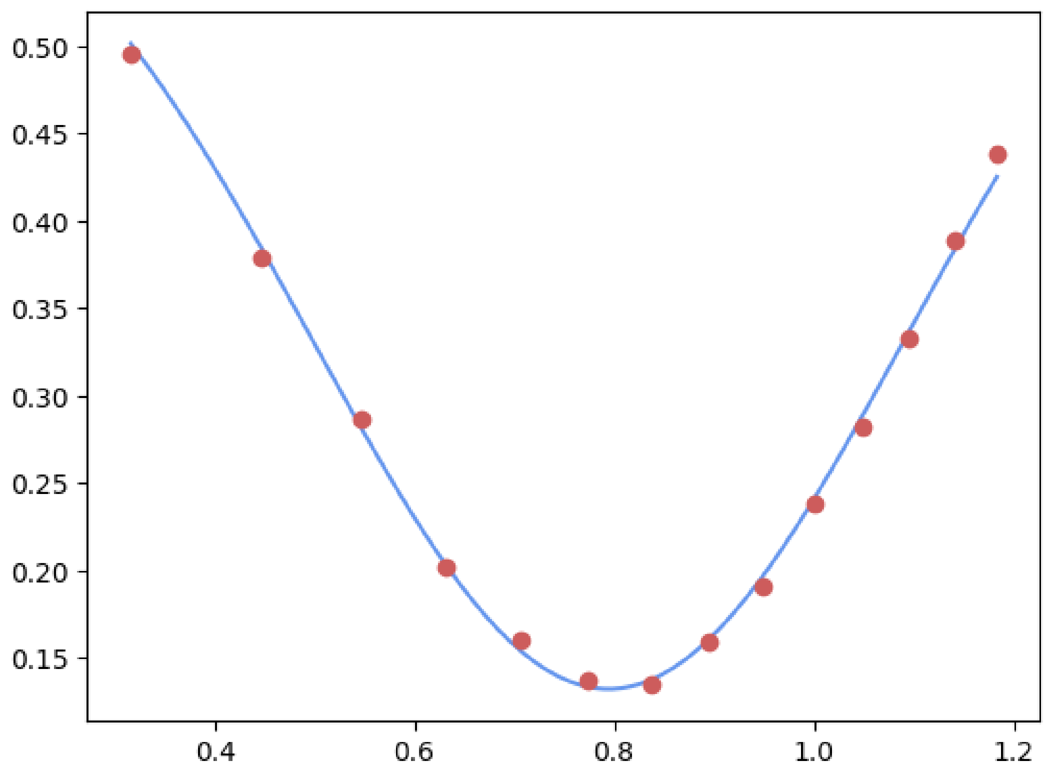


```
[ ]: def GaussianFunction(x, mu, std, scale, height):
         ans = []
         for el in x:
             ans.append(- scale * np.exp(-(el-mu)**2/(2*std**2)) + height)
         return ans

     def FWHM(standardDeviation):
```

```
    return 2 * np.sqrt(2 * np.log(2)) * standardDeviation

def CalculateMinimumProbability(electronMomentum, probabilityOfScattering,␣
 ↪showPlot=False):
    dipLocation = electronMomentum[0:14]
    dipProbability = probabilityOfScattering[0:14]

    pars, cov = curve_fit(GaussianFunction, dipLocation, dipProbability, [0.8,␣
 ↪0.5, 1, 1.15])

    xRange = np.linspace(np.min(dipLocation), np.max(dipLocation), 100)

    if showPlot:
        plt.errorbar(dipLocation, dipProbability, fmt="o")
        plt.plot(xRange, GaussianFunction(xRange, pars[0], pars[1], pars[2],␣
 ↪pars[3]))
    print(cov)
    return [pars[0], np.sqrt(cov[0][0])]

CalculateMinimumProbability(electronMomentum, probabilityOfScattering,␣
 ↪showPlot=True)
```

```
[[ 5.50751763e-06 -1.07751624e-05 -2.08896894e-05 -2.14370733e-05]
 [-1.07751624e-05  2.08683233e-04  4.02427148e-04  4.29699475e-04]
 [-2.08896894e-05  4.02427148e-04  8.39228547e-04  8.74228691e-04]
 [-2.14370733e-05  4.29699475e-04  8.74228691e-04  9.21005543e-04]]
```

```
[ ]: [0.7943370769321534, 0.0023468100957812162]
```

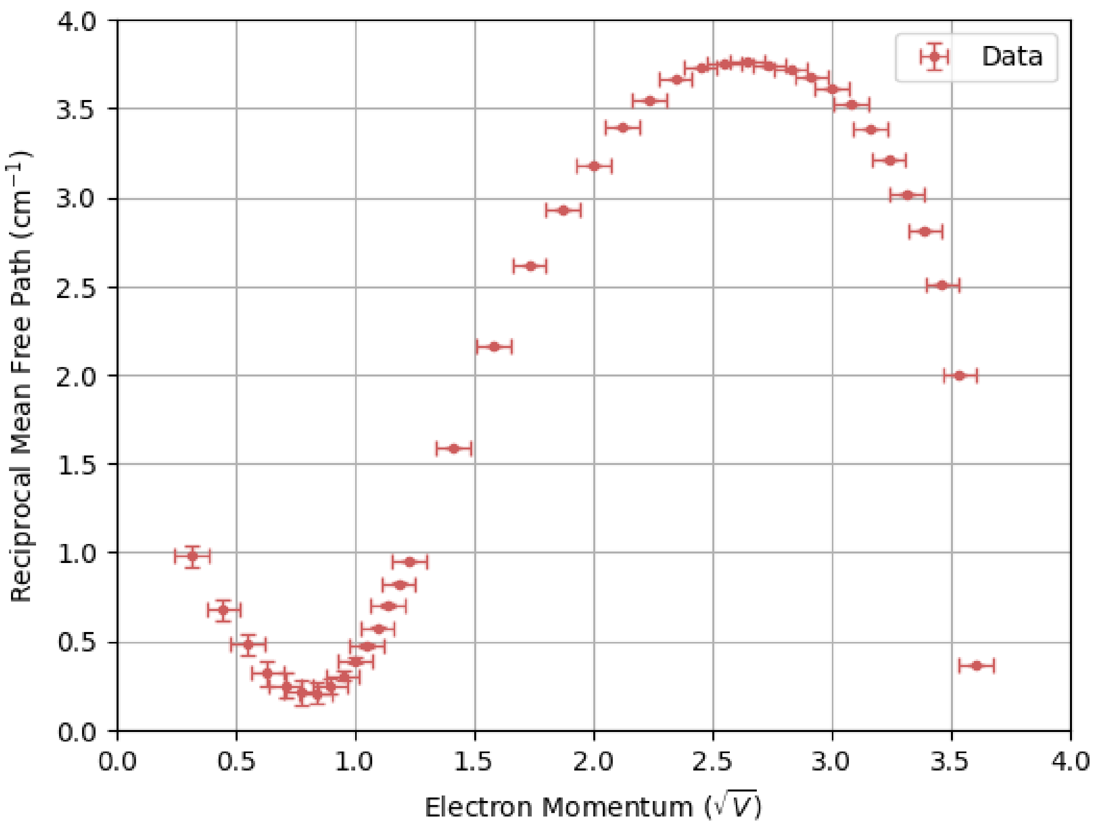


```
[ ]: def DensityCrossSection(probabilityOfScattering):
         return - np.log(1 - probabilityOfScattering) / 0.7
```

```
[ ]: densityCrossSectionUncertainty = [np.sqrt((((1-1/probabilityOfScattering)/0.
      ↪7)**2 * (uncertainnty)**2) \
         for probabilityOfScattering, uncertainnty in zip(probabilityOfScattering,␣
      ↪probabilityOfScatteringUncertainty)]
```

```
[ ]: reciprocalMeanFreePath = [DensityCrossSection(el) for el in␣
      ↪probabilityOfScattering]

     plt.errorbar(electronMomentum, reciprocalMeanFreePath, xerr=np.sqrt(0.005),␣
      ↪yerr=densityCrossSectionUncertainty, fmt=".", capsize=3, linewidth=1,␣
      ↪label="Data")

     plt.legend()
     plt.grid()
     plt.xlim(0, 4)
     plt.ylim(0, 4)
     plt.xlabel("Electron Momentum ($\sqrt{V}$)")
     plt.ylabel("Reciprocal Mean Free Path (cm$^{-1}$)")
```

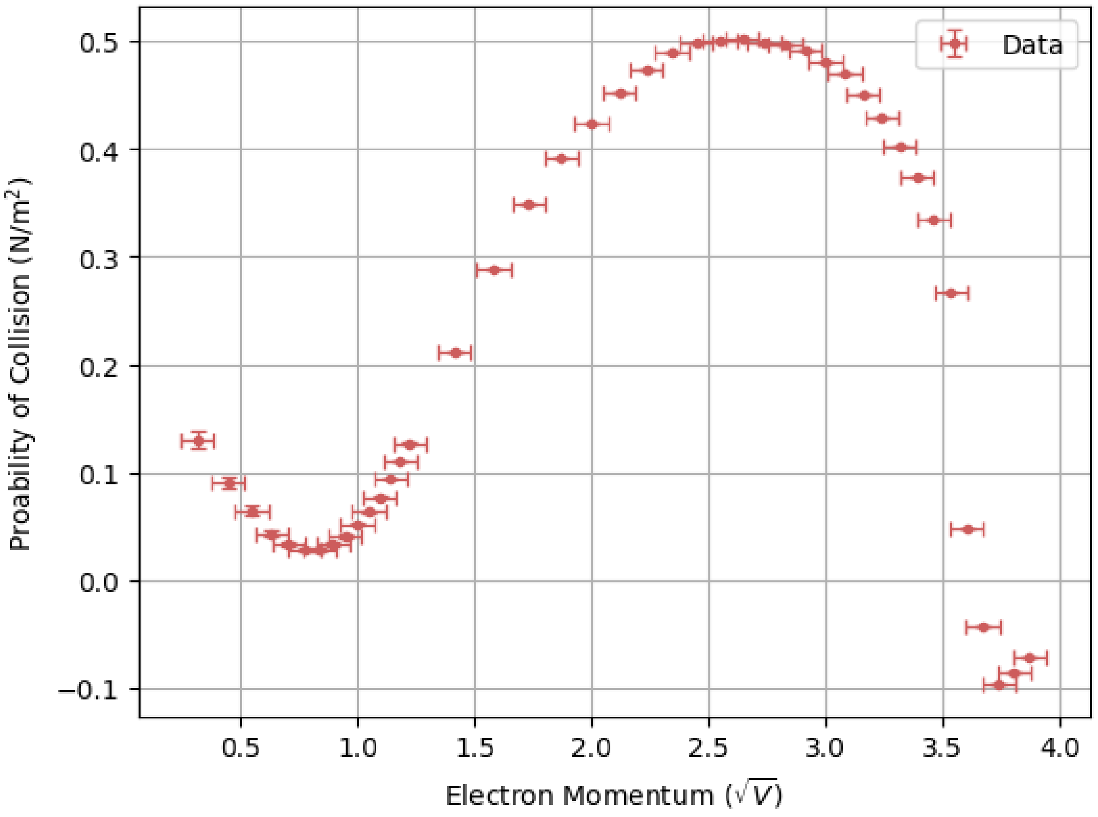
```
[ ]: Text(0, 0.5, 'Reciprocal Mean Free Path (cm$^{-1}$)')
```



```
[ ]: def ProbabilityOfCollision(pressure, reciprocalMeanFreePath):
         return pressure * reciprocalMeanFreePath
```

```
[ ]: torr = 133.322 # pascals

     probabilityOfCollision = [ProbabilityOfCollision(1e-3 * torr, el) for el in␣
      ↪reciprocalMeanFreePath]
     probabilityOfCollisionUncertainty = [np.sqrt((ProbabilityOfCollision(1e-3 *␣
      ↪torr, reciprocalMeanFreePath))**2 * \
         densityCrossSectionUncertainty**2) for reciprocalMeanFreePath,␣
      ↪densityCrossSectionUncertainty in \
             zip(reciprocalMeanFreePath, densityCrossSectionUncertainty)]

     plt.errorbar(electronMomentum, probabilityOfCollision, xerr=np.sqrt(0.005),␣
      ↪yerr=probabilityOfCollisionUncertainty, fmt=".", capsize=3, linewidth=1,␣
      ↪label="Data")

     plt.legend()
     plt.grid()
     plt.xlabel("Electron Momentum ($\sqrt{V}$)")
     plt.ylabel("Proability of Collision (N$/$m$^2$)")
```

```
[ ]: Text(0, 0.5, 'Proability of Collision (N$/$m$^2$)')
```



```
[ ]: extensionData = LoadFile(rootPath + r"/extensionsData.txt")
reversePolarityAccelerationVoltage = extensionData[:,0]
reversePolarityShieldVoltage = [el/1000 for el in extensionData[:,1]]

reversePolarityShieldCurrent = [Current(voltage, 100) for voltage in␣
 ↪reversePolarityShieldVoltage]
```

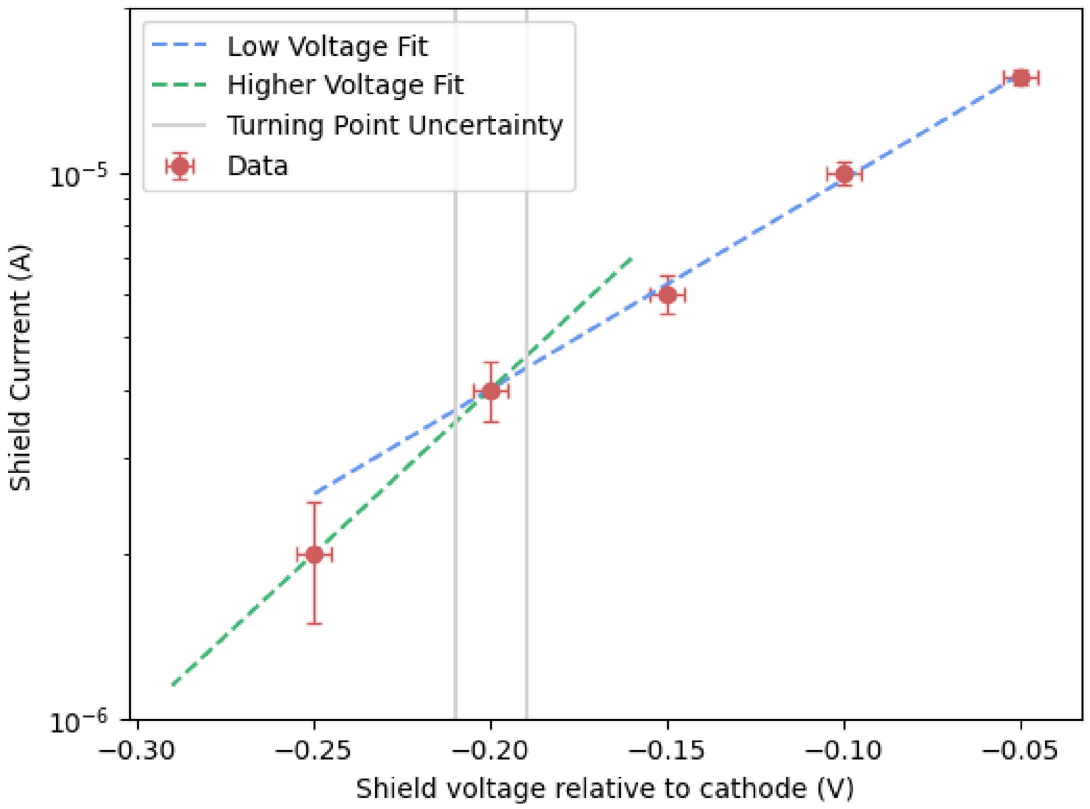
```
[ ]: def ExpLine(x, A, B):
    return A * np.exp(B*x)
```

```
[ ]: plt.errorbar(reversePolarityAccelerationVoltage[0:-2],␣
 ↪reversePolarityShieldCurrent[0:-2], xerr=0.005, yerr=Current(0.05/1000,␣
 ↪100), fmt="o", capsize=3, linewidth=1, label="Data")

xRangeRight = np.arange(-0.05, -0.26, -0.01)
```

```
xRangeLeft = np.arange(-0.16, -0.30, -0.01)
parsRight, covRight = curve_fit(ExpLine, reversePolarityAccelerationVoltage[0:
 ↪3], reversePolarityShieldCurrent[0:3])
parsLeft, covLeft = curve_fit(ExpLine, reversePolarityAccelerationVoltage[3:
 ↪-1], reversePolarityShieldCurrent[3:-1])

parsLowerFit, covLowerFit = curve_fit(ExpLine, [-.2, -0.25],␣
 ↪[reversePolarityShieldCurrent[3]-Current(0.05/1000, 100),␣
 ↪reversePolarityShieldCurrent[4]-Current(0.05/1000, 100)])
parsUpperFit, covUpperFit = curve_fit(ExpLine, [-.2, -0.25],␣
 ↪[reversePolarityShieldCurrent[3]+Current(0.05/1000, 100),␣
 ↪reversePolarityShieldCurrent[4]+Current(0.05/1000, 100)])

plt.plot(xRangeRight, ExpLine(xRangeRight, parsRight[0], parsRight[1]),␣
 ↪label="Low Voltage Fit", linestyle="--")
plt.plot(xRangeLeft, ExpLine(xRangeLeft, parsLeft[0], parsLeft[1]),␣
 ↪label="Higher Voltage Fit", linestyle="--")

#plt.plot(xRangeLeft[4:10], ExpLine(xRangeLeft[4:10], parsLowerFit[0],␣
 ↪parsLowerFit[1]), linestyle="dotted", color="black", label="Slope␣
 ↪Uncertainty")
#plt.plot(xRangeLeft[4:10], ExpLine(xRangeLeft[4:10], parsUpperFit[0],␣
 ↪parsUpperFit[1]), linestyle="dotted", color="black")

plt.vlines(x=-0.1997+0.01, ymin=0, ymax=1e-4, color="lightgrey", label="Turning␣
 ↪Point Uncertainty")
plt.vlines(x=-0.1997-0.01, ymin=0, ymax=1e-4, color="lightgrey")

plt.yscale("log")
plt.ylim(1e-6, 2e-5)
plt.ylabel("Shield Currrent (A)")
plt.xlabel("Shield voltage relative to cathode (V)")
plt.legend()

print(f"Right line parameters: {parsRight}")
print(f"Left line parameters: {parsLeft}")
print(f"Slope uncertainty: {np.max([abs(parsLeft[1] - parsUpperFit[1]),␣
 ↪abs(parsLeft[1] - parsLowerFit[1])])}")
```

```
/home/daraghhollman/.local/lib/python3.10/site-
packages/scipy/optimize/_minpack_py.py:906: OptimizeWarning: Covariance of the
parameters could not be estimated
  warnings.warn('Covariance of the parameters could not be estimated',
```

```
Right line parameters: [2.34936550e-05 8.84565998e+00]
Left line parameters: [6.40000000e-05 1.38629436e+01]
Slope uncertainty: 3.0830135965451717
```



Low Voltage Fit
Higher Voltage Fit
Turning Point Uncertainty
Data
Shield Currrent (A)
10^{-5}
10^{-6}
−0.30
−0.25
−0.20
−0.15
−0.10
−0.05
Shield voltage relative to cathode (V)