QUEEN MARY UNIVERSITY OF LONDON
SCHOOL OF PHYSICAL AND CHEMICAL SCIENCES

A New Tool for Investigating Dispersive Time-of-Flight Data

Research Project

Luke Hasler
Supervisor: Dr. Theo Kreouzis

A report submitted in partial fulfilment of the
requirements for the Physics MSci (Hons) degree

April 2022

Abstract

A Python tool integrating a user interface has been created and extensively tested to streamline the cleaning and analysis of electron time-of-flight data. Time-of-flight data is used to investigate the properties of organic semiconductors for use in radiation detectors. The algorithm has been used to investigate the time-of-flight properties of a blend of poly(3-hexylthiophene), and phenyl-C_{61}-butyric acid methyl ester before and after it was exposed to the air. It was found that the vacuum electron mobility of the sample with no external electric field was about 0.01$cm^2V^{-1}s^{-1}$ with a γ value of $2.9x10^{-5}$, after exposure to air that rose to 0.1$cm^2V^{-1}s^{-1}$ with a γ value of $-1.4x10^{-4}$. For the vacuum data, an α_1 range of about 0.02 was found and after exposure, this increased to 0.25 although α_1 didn't behave well with pad voltage. α_2 was found to have behaved non-physically before and after exposure. It was found that the sample had about 36.5nC of photogeneration occurring without an external field and after exposure this ability to photogenerate without an external field stops. For holes, the results on photogeneration were not so clear. Significant increases in mobility occurred once the sample was exposed to air and α_1 and α_2 were found not to be reliable dispersion parameters. It was decided that the times found for the exposed sample were likely artefacts as a result of the strange behaviours with mobility and α_1 and α_2 and not arrival times. A trend was noted where the ratio of the photocharge after and before exposure and the pad voltage was non-linear and differed completely for electrons and holes which is believed to have been caused by the impact of noise in measuring negligible dark currents. It was decided that the algorithm completed its task successfully and that the algorithm would be used in the future by time-of-flight researchers at Queen Mary University of London's Centre for Condensed Matter Physics.

List of Contents

1 Acknowledgments

I would like to thank my supervisor Dr Theo Kreouzis for taking on the challenging task of teaching me both the specific science behind the organic semiconductors that this report will primarily consist of and whose expertise was invaluable in the creation of this report.

Thank you to Fani Eirini Taifakou for fabricating the sample discussed in this report and to Fa Zhang and Sarah Alsharif for taking some of the measurements that will be analysed here.

2 Introduction

Radiation detection is an area of active interest in a variety of fields. The first of these is particle physics in which it is envisioned that new detectors may be able to use organic semiconductors to improve upon existing particle detectors [1] and another use is in the detection of radioactive substances for security. Modern medicine uses radioactive substances in nuclear scans where radioactive substances enter the body and are detected externally and for x-rays [2] and various other industries have need of radiation detection such as in nuclear power stations where detecting the leakage of radioactive substances is key to maintaining public safety. In attempting to create materials that meet the strict budgets and tight engineering conditions that are necessary for the construction of effective and accurate radiation detectors we could use organic semiconductors as they can undergo large-area device fabrication, are low cost, and are relatively flexible [3] and they have already been used in a wide variety of other applications such as in organic light-emitting diodes (OLEDs) and other organic photovoltaics (OPVs), and in organic field-effect transistors (OFETs) [4].

There has been little research into the usage of organic semiconductors as radiation detectors but most of that work has been completed in detecting γ radiation [5] and x-rays [2] [6] [7]. Other research into the detection of α radiation has been discussed in [3]. The least amount of research has been into detecting β radiation [8] and into detecting neutrons [9]. At Queen Mary University of London, our researchers are primarily researching methods of detecting α and neutron emission [3] and a large part of this research involves taking samples of organic semiconductors and increasing their thickness from their uses in previous applications such as with OPVs where only a thin strip of material was required [4]. α emission is very highly ionising, and whilst neutrons are not charged, they can ionise as a result of elastic and inelastic collisions, and because of this they do not travel far through our sample, which as a result of investigations in [3] we know the thickness that a sample must be to capture all ionising radiation that we wish to detect. This thickness is greater than samples used in previous applications of organic semiconductors.

The difficulty that is posed in our usage of these materials is that, whilst much is known of their properties such as the mobility when they are prepared in extremely thin samples which are suitable for use in OLEDs, we cannot be certain that these properties will remain constant when the thickness of the samples is vastly scaled up. As such, time-of-flight experiments have been conducted to investigate the properties of thicker samples of organic semiconductors generating large and complex sets of data that need to be analysed to find vital information used in the selection process to find the ideal material for radiation detection.

In order to analyse this data, researchers currently plot the data and then choose trend lines by hand to fit the data which is a slow and inaccurate process that uses up valuable research time. It was therefore decided to create an algorithm in Python in a Jupyter Notebook that would attempt to streamline the process and improve the ability of researchers to analyse the data. It was decided that the success of the algorithm would initially be tested on time-of-flight data for a sample of a blend of poly(3-hexylthiophene) (P3HT) and phenyl-C_{61}-butyric acid methyl ester (PCBM) (which is overall known as $P3HT : PC_{60}BM)$ [3] [10] [11] which was held under vacuum and then, once the algorithm was found to work satisfactorily, it was tested again on data for the same sample after it had been exposed to the air as this data would be far noisier so it would be a good test as to whether or not the algorithm could analyse noisy data and as it would allow us to probe the changes in the properties of the sample resulting from exposure to the air.

Success in this project could provide time-of-flight researchers with a new tool for improving the workflow of analysing data and potentially modernise a system of analysis that has been used for many years whilst also providing an insight into the behaviour of $P3HT : PC_{60}BM$ before and after exposure to the air which could lead to advancement in radiation detection technologies. Note also that the algorithm that will be used throughout this report is useful not only for radiation detection research but also for a myriad of other applications relating to electron time-of-flight data for organic semiconductors.

3 Theory

The first step in investigating the time-of-flight for a sample is to start at the simple fact that to find the transit time we shall need to measure the drift velocity of charge carriers using equation 1 where d is the thickness of the sample, v_{drift} is the drift velocity of charge carriers and t_t is the transit time for charge carriers to move from one side of the sample to the other [4].

$$v_{drift} = \frac{d}{t_t} \tag{1}$$

To continue our mathematical explanation, the drift velocity can be related to the external electric field (known as E) and the charge mobility (known as μ) using equation 2 [4]. The charge mobility is the ability for charged particles to move through a medium when it is impacted by an external electric field and as such the higher the value of the mobility, the shorter the transit time, and in a typical sample this means that a greater externally applied electric field decreases the transit times of charge carriers [4].

$$v_{drift} = \mu E \tag{2}$$

Next, we know that the electric field can be related to the thickness of the sample and the pad voltage across the sample using equation 3

$$E = \frac{V_0}{d} \tag{3}$$

where V_0 is the pad voltage across the sample [4] when we approximate that the electric field is uniform. Another important step we shall need for our work is to rearrange Ohm's Law to find an expression for the current running through the sample in equation 4

$$I = \frac{V(t)}{R_{load}} \tag{4}$$

where $V(t)$ is the potential across the load resistor [4] which was in series with the sample. By reformulating equations 2 and Ohm's Law we can find equation 5 for the current density $J = \frac{I}{A}$ (where A is the cross-sectional area) which helps us to relate the overall current density passing through the sample to the behaviours of the individual charge carriers [12]

$$J = nqv_{drift} = nq\mu E = \sigma E \tag{5}$$

where n is the number of charge carriers passing through a surface in the sample at any one time, q is the charge of each carrier (which in our case is always the electron charge), and σ is the conductivity. The conductivity can be found using equation 6

$$\sigma = \frac{nq^2\tau}{m} \tag{6}$$

where τ is the mean time between collisions, m is the mass of the charge carriers and the conductivity σ is also the reciprocal of the resistivity ρ [12].

It is predicted in [13] that we can relate the current passing through a sample to the transit time using two bisecting lines relating to the current through equations 7 and 8

$$I \propto t^{-(1-\alpha)} \tag{7}$$

$$I \propto t^{-(1+\alpha)} \tag{8}$$

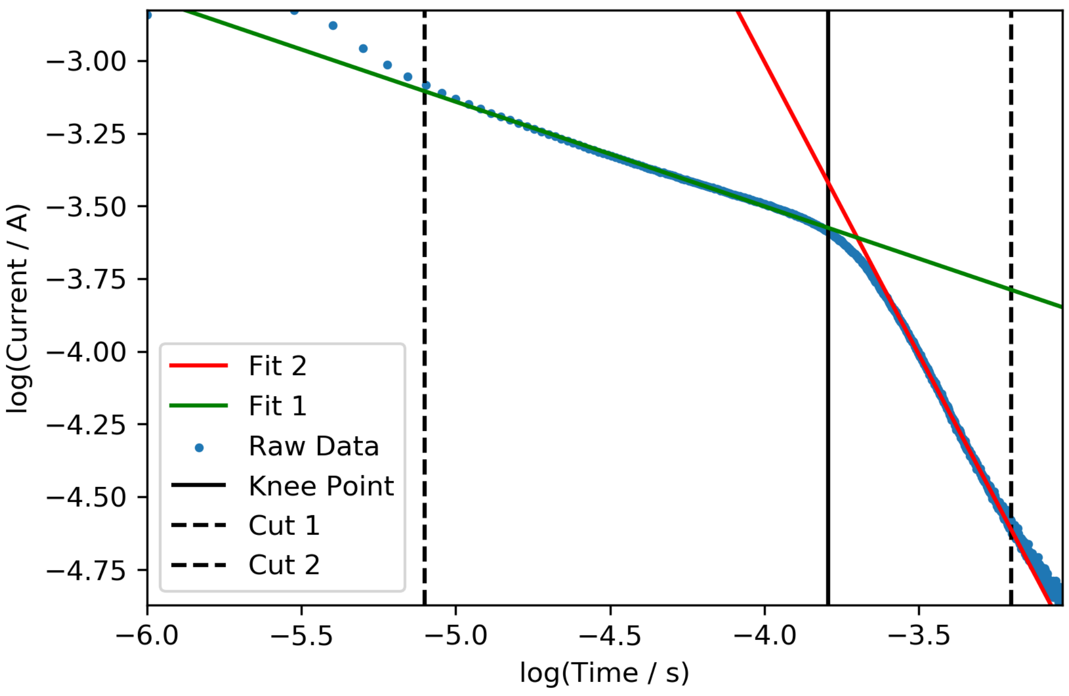


Figure 1: A typical time-of-flight data double logarithmic graph for less dispersive data where the lines derived from α_1 and α_2 are shown in green and red respectively. This graph is a standard output of the algorithm.

where α is a constant and t is the transit time. α is predicted to have physical values of $0 \leq \alpha \leq 1$ where $\alpha = 0$ indicates that there is a lot of dispersion and $\alpha = 1$ means there is no dispersion at all. It was soon realised in [14] that during experiment a single value for α would not suffice as a knee in the double logarithmic plot of current and time meant that two distinctly different values and equations for relating the current and the time-of-flight were needed. These relations are shown in equations 9 and 10 and illustrated in Figure 1

$$I \propto t^{-(1-\alpha_1)} \tag{9}$$

$$I \propto t^{-(1+\alpha_2)} \tag{10}$$

where α_1 and α_2 are constants dependent on the gradients of straight lines along the trend of a double logarithmic graph of the current against the time [14]. Physical values of α_1 and α_2 match those of α in equations 7 and 8. Next, we must relate the charge mobility to the current in a way that we can investigate the time-of-flight. We can use equation 11 which helps us to link the pad voltage and the charge mobility of the sample [4].

$$\mu = \frac{d^2}{V_{applied}\ t_t} \tag{11}$$

It is currently theorised by Bässler [14] that the mobility of a sample is altered by an external electric field according to equation 12

$$\mu(E,T) = \mu(E=0,T)\ e^{\gamma(T)\sqrt{E}} \tag{12}$$

where T is the temperature of the sample. Relationships of this kind can be demonstrated using Poole-Frenkel plots where a linear relationship is found when comparing the logarithm of the mobility and $E^{1/2}$. This can be seen more explicitly using equation 13 where γ is the gradient of the graph and $ln(\mu(E=0))$ is the y-intercept.

$$ln(\mu) = ln(\mu(E=0)) + \gamma\sqrt{E} \tag{13}$$

If we decide to further investigate the properties of equation 12 we can find the Gill model given by equation 14 which demonstrates the temperature and field dependence of the mobility [14]

$$\mu(E,T) = \mu_0 e^{-\frac{E_a}{kT}+B(\frac{1}{T}-\frac{1}{T_0})\sqrt{E}} \tag{14}$$

where μ_0 is the initial charge mobility, E_a is the activation energy, B is the magnetic field strength, k is the Boltzmann constant, T is the temperature of the sample when the external field is applied, and T_0 is the initial temperature of the sample before the external electric field is applied [15]. Notice that this equation has the same form as equation 12 where we have set values for γ and expanded $\mu(E=0,T)$ to find a more accurate expression to match nature.

Similar, more accurate, attempts have since been made at matching the behaviour of nature using Monte-Carlo methods such as the Gaussian disorder model by Heinz Bässler [14] and the correlated disorder model by Sergei Novikov [16] which are given by equations 15 and 16 respectively

$$\mu(E,T) = \mu_0 e^{-(\frac{2\sigma}{3kT})^2} e^{C_0\sqrt{E}((\frac{\sigma}{kT})^2-\Sigma^2)},\ for\ \Sigma \geq 1.5,\ else\ let\ \Sigma^2 = 2.25 \tag{15}$$

where σ is the energy disorder parameter, C_0 is the concentration dependence, and Σ is the off-diagonal disorder (this describes spatial disorder) [14]. Unfortunately, the Gaussian disorder model fails to accurately predict trends in the mobility at low values of the electric field [14] so the correlated disorder model is a more useful method predicting the behaviour of temperature dependent mobilities over the pad voltage.

$$\mu(E,T) = \mu_0 e^{-(\frac{3}{5}(\frac{\sigma}{kT})^2} e^{C_0\sqrt{\frac{eER}{\sigma}}((\frac{\sigma}{kT})^{\frac{3}{2}}-\Gamma))} \tag{16}$$

where R is the cell spacing between two adjacent sites in a cubic lattice, and Γ is a variable characterising disorder in the geometry. Once again, notice that both equations take the form of equation 12. Whilst it would be difficult to characterise which equation best fits our experimental data, they are both derived from equation 12 using Monte-Carlo methods so we can use the data from graphs such as Figure 1 to investigate the behaviour of samples under the influence of external electric fields.

Another way to determine the charge mobility is using the Mott-Gurney Law given by equation 17 by fitting the Mott-Gurney Law to a graph of J against V in the region where $J \propto V^2$ [17] [18].

$$J = \frac{9}{8}\epsilon\mu\frac{V^2}{d^3} \tag{17}$$

where $\epsilon = \epsilon_0\epsilon_r$ is the permittivity and ϵ_0 is the permittivity of free space.

Unfortunately, the Mott-Gurney Law relies on the assumptions that our contacts with the sample are ohmic, that the sample is defect-free and free of energetic disorder, and that there is no electric field at the interface with the sample [17]. These assumptions are plainly not true in our experiment - particularly as we shall allow the sample to be exposed to the air so plenty of defects will exist in the sample - so we shall not be using the Mott-Gurney Law during our analysis.

Now that we have found expressions for the charge mobility we should provide an explanation for some of the phenomena that will be encountered in the experiment. The first phenomenon I would like to explore is the dark current - any current that is generated in the device before it has been exposed to light [4]. This current should be created purely by heat in the sample but external photons hitting the sample can also contribute to the dark current [4] as can doping. In our experiment, the sample was kept in the dark during operation so this should have a negligible effect on our results. When we take measurements of the time-of-flight we shall find that this includes the dark current but all of the interesting behaviour we want to study will be seen in the photocurrent which we can find by subtracting the dark current from our overall measured current.

To define a dark current value that we can subtract from all measured values of the current we find the mean average current value of all data points measured before a laser pulse is measured at $t = 0s$. This is shown in equation 18

$$\bar{I_d} =< (I \,\forall\, t < 0s) > \tag{18}$$

where $\bar{I_d}$ is the average dark current. Next, we can define the photocharge using equation 19

$$Q_{photo} = \int I dt = q\frac{s}{d} \tag{19}$$

where s is the distance that charges are displaced and d is the thickness of the sample. With the assumption that $s = d$ we can then show that $Q_{photo} = q$. It is at this point that we may decide to calculate the quantum efficiency of our experiment using equation 20

$$q_{photo} = N_{ph}(n\psi)e \tag{20}$$

where N_{ph} is the number of photons hitting the sample, $(n\psi)$ is a quantum efficiency, and e is the charge of an electron [4].

A significant effect we shall notice in our data is that of charge trapping. This is an effect where charge carriers are unable to transport as they have been "trapped" often in highly electronegative regions (for electrons) [19]. The general result of charge trapping is that the charge mobility efficiency decreases according to equation 21 where n is the number of charge carriers in the sample and n_{trap} is the number of charge carriers that have been prevented from contributing to the charge mobility by charge trapping [4].

$$\mu_{eff} = \frac{n - n_{trap}}{n}\mu \tag{21}$$

There are many causes of charge trapping but the most important in this experiment is when a sample is exposed to certain chemicals which can cause impurities that act as quantum wells and trap charges [19]. For us, this will mean exposure to air and more importantly exposure to oxygen which creates a largely electronegative region in the sample. These regions are then excellent electron traps. This means that we predict a large decrease in the electron mobility of a semi-conductor sample after it has been exposed to the air which is caused by the trapping of electrons in electronegative regions which prevents electrons from carrying their charge.

4 Methodology

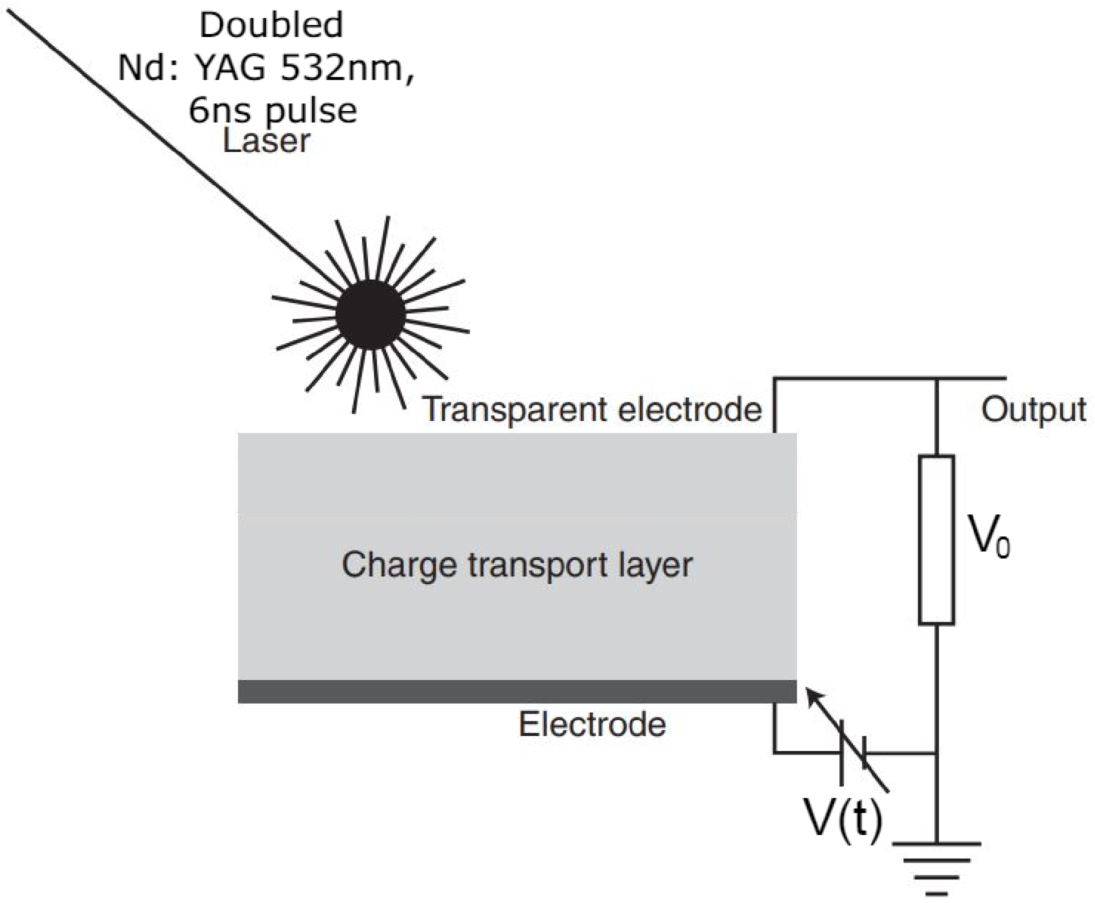


Figure 2: Diagram illustrating the experimental apparatus behind the time-of-flight experiment. A short laser pulse is fired at an electrode which generates charge carriers in a nearby layer in the sample. The sample in our experiment is 16.9 ± 1.3mm in thickness and we used a green doubled Nd: YAG 532nm, 6ns pulse. The charge carriers travel through the sample to the other electrode and transients are detected by being sent to an oscilloscope and then recorded into a CSV file on a computer. This diagram was adapted from Figure 5.6 in [4].

To conduct the experiment required to gather the datasets that will be analysed in this paper, an experimental setup shown in Figure 2 (adapted from [4]) was used where a sample of $P3HT$: $PC_{60}BM$ was tested under vacuum by myself - it had been fabricated by Fani Eirini Taifakou in an atmosphere of dry nitrogen but it was known that the sample would not react with nitrogen gas

- and then once again after it had been exposed to the air by Fa Zhang and Sarah Alsharif in the laboratory. Sets of comma-separated value (CSV) data were structured such that two columns of data, the time in seconds and pad voltage in volts, were captured using a digitised oscilloscope and a computer would then be used to put those files into the code that will be discussed later which analysed the data. This data had various amounts of headings and unwanted information in the first few rows which had to be cleaned by the algorithm. The length of the CSV file is not limited by the algorithm so some datasets contained 1000 data points and others contained 10,000 data points. The thickness of the sample was measured using stylus profilometry by Fa Zhang and Sarah Alsharif using a DektakXT profilometer.

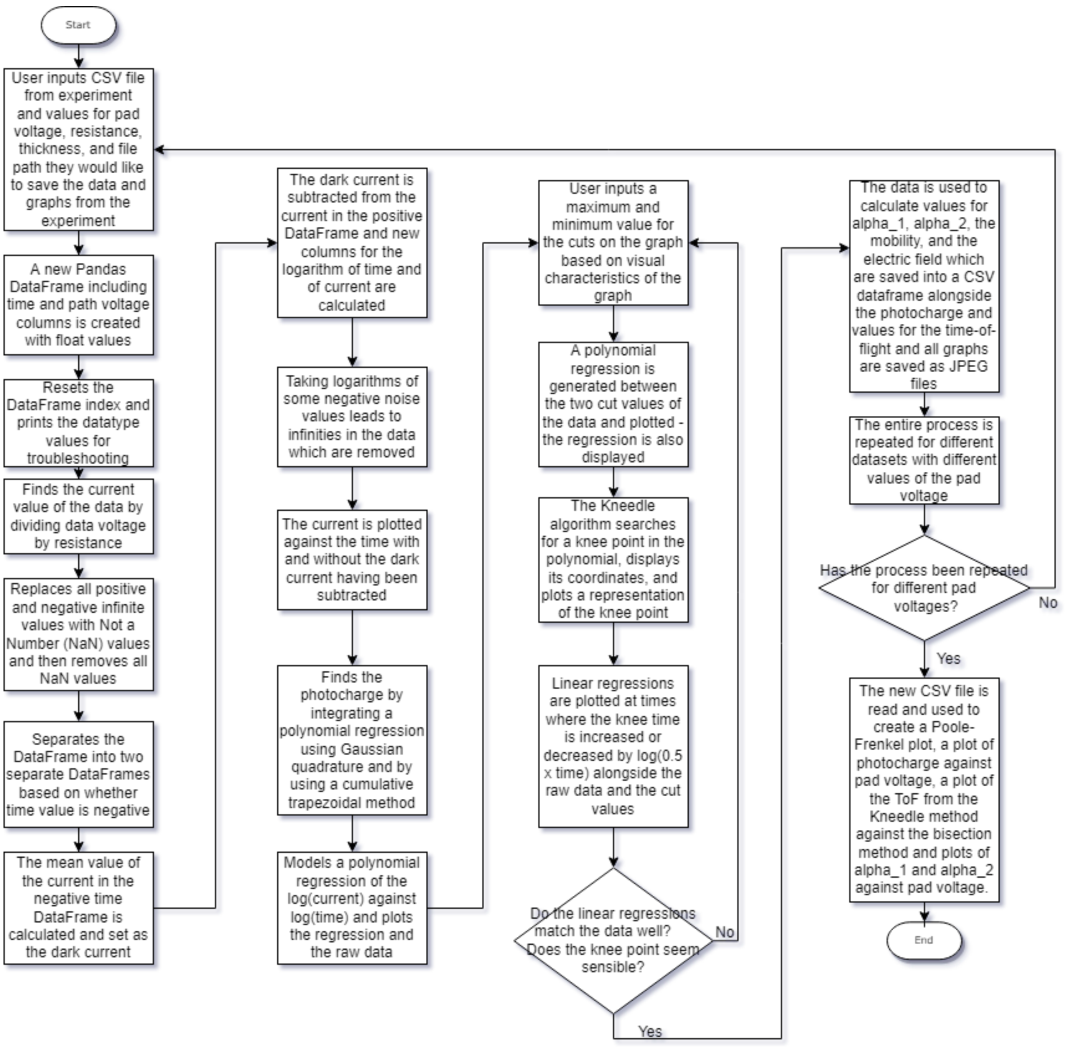

Start
User inputs CSV file from experiment and values for pad voltage, resistance, thickness, and file path they would like to save the data and graphs from the experiment
A new Pandas DataFrame including time and path voltage columns is created with float values
Resets the DataFrame index and prints the datatype values for troubleshooting
Finds the current value of the data by dividing data voltage by resistance
Replaces all positive and negative infinite values with Not a Number (NaN) values and then removes all NaN values
Separates the DataFrame into two separate DataFrames based on whether time value is negative
The mean value of the current in the negative time DataFrame is calculated and set as the dark current
The dark current is subtracted from the current in the positive DataFrame and new columns for the logarithm of time and of current are calculated
Taking logarithms of some negative noise values leads to infinities in the data which are removed
The current is plotted against the time with and without the dark current having been subtracted
Finds the photocharge by integrating a polynomial regression using Gaussian quadrature and by using a cumulative trapezoidal method
Models a polynomial regression of the log(current) against log(time) and plots the regression and the raw data
User inputs a maximum and minimum value for the cuts on the graph based on visual characteristics of the graph
A polynomial regression is generated between the two cut values of the data and plotted - the regression is also displayed
The Kneedle algorithm searches for a knee point in the polynomial, displays its coordinates, and plots a representation of the knee point
Linear regressions are plotted at times where the knee time is increased or decreased by log(0.5 x time) alongside the raw data and the cut values
Do the linear regressions match the data well? Does the knee point seem sensible?
No
Yes
The data is used to calculate values for alpha_1, alpha_2, the mobility, and the electric field which are saved into a CSV dataframe alongside the photocharge and values for the time-of-flight and all graphs are saved as JPEG files
The entire process is repeated for different datasets with different values of the pad voltage
Has the process been repeated for different pad voltages?
No
Yes
The new CSV file is read and used to create a Poole-Frenkel plot, a plot of photocharge against pad voltage, a plot of the ToF from the Kneedle method against the bisection method and plots of alpha_1 and alpha_2 against pad voltage.
End

Figure 3: Flowchart describing the workflow of the time-of-flight investigation method that has been used in this investigation.

This project will be primarily completed using the code that is available in the appendix. It is an algorithm that can be used to take a CSV file containing time-of-flight data - whether it is saved online or in physical storage, convert it into a Pandas DataFrame [20], clean it, and then analyse the cleaned data using regressions in scikit-learn [21] and knee detection algorithms from the kneedle package [22] to find and represent our results in a set of automatically generated graphs produced using Matplotlib [23]. A flowchart designed to simply illustrate the workflow of using the code is displayed in Figure 3.

The code went through several research and development phases before its current form. The first iteration of the code used only the kneedle algorithm [22] to find a value for the time-of-flight without using any cuts to the data at all but it was found that the algorithm would almost always fail to find the knee point and would instead mistakenly find a knee point at the very highest or lowest time due to the curvature caused by noise at high and low values of the logarithm of time. This was solved by including cuts on either side of the knee point to remove any noise and unwanted curvature in the data. However, it was found by comparing the results of using only the kneedle algorithm with other attempts at analysing the data in [3] that using only the kneedle algorithm frequently underestimated the time-of-flight and visually seemed to slightly miss the knee of the graphs themselves.

It was understood that the time-of-flight is usually defined using the bisection point of two trendlines defined by α_1 and α_2 as seen in equations 9 and 10 [14] so in the next iteration of the code the user was encouraged to enter two new values of the logarithm of time where they believed the best trendlines could be fitted for the bisection method. Whilst this was found to be more effective at finding an accurate value for the time-of-flight it also increased the amount of input the user had to give and as such meant testing many sets of data would take a greater amount of time - which was against the entire goal of reducing the time it takes to complete analysis. As such it was noticed that the kneedle algorithm seemed to underestimate the logarithm of time by about 0.3 along the logarithmic time axis in each experiment and as such, it was decided that instead of making the user input a second set of bisection point values, the value of the bisection points could be decided by the point found by the kneedle algorithm adding and subtracting an arbitrary 0.3 log(seconds) from the time the kneedle algorithm had found. This was found to produce excellent results and cut the amount of user input needed to run the algorithm at the expense of introducing arbitrariness to the creation of the trendlines used in the bisection process.

Later versions of the code would also include automation of generating plots such as Poole-Frenkel plots and plots of the raw data including the data after the subtraction of the dark current amongst other plots to allow for quicker visual analysis and changes which allowed the user to save their plots and data in a file structured manner onto their local computer at a given file path to allow for researchers to more easily keep track of their research.

An example of the CSV file output by the algorithm once a set of readings for a range of different pad voltages of the sample before it was exposed to the air is available in Tables 1, 2, and 3. Note that this would usually appear within one singular CSV file and I have had to separate this into three separate tables in order to fit the raw output onto a page.

There are a wide variety of graphs created and saved automatically by the algorithm with each graph being generated and saved again every time a new set of data points is added to the output CSV file. A full representation of the code and the output of each block of code is available in the appendix of this report. To briefly summarise these graphs include, but are not limited to graphs of the current against the pad voltage before and after the subtraction of the dark current, a double logarithmic graph of the current against time including markers for the knee point found by the kneedle algorithm, the cut points, and the bisection lines derived from α_1 and α_2, a Poole-Frenkel plot, a plot of photocharge against the pad voltage, a plot of the time-of-flight found by the kneedle algorithm against the time-of-flight found using the bisection method, and two plots of α_1 and α_2 against the pad voltage. Various other graphs are generated by the algorithm but not saved to the user's files as they are intended to be used as tools to refine the choice of cutting points rather than as plots for scientific analysis such as plots demonstrating the usage of polynomial regressions, plots zoomed-in on the knee point, and a plot illustrating the methodology taken by the kneedle algorithm.

Many, but not all, of the plots shown in the results section of this report have been created automatically by the algorithm.

Table 1: The first portion of the output CSV file generated automatically by the algorithm for the sample before it was exposed to the air.

Pad Voltage/V	Resistance/Ohms	Thickness/m	Time [Fit]/microsecs	Time [Knee]/microsecs	Dark current/ micro A	Photocharge [Raw]/C	
50	1000	1.69E-05	609.8479187	482.5895969	75.26998024	7.38E-08	
60	1000	1.69E-05	477.135835	398.8357588	85.02029454	8.27E-08	
70	1000	1.69E-05	365.7919935	305.2805716	129.5585711	9.48E-08	
80	1000	1.69E-05	295.2309652	241.4858157	192.0516582	1.00E-07	
90	1000	1.69E-05	244.057314	181.0344531	273.7544643	1.10E-07	
100	1000	1.69E-05	199.3564941	149.5685348	415.4391447	1.13E-07	
110	1000	1.69E-05	165.3438448	126.4650234	598.0250822	1.22E-07	
120	1000	1.69E-05	134.3673578	106.9629664	640.8972039	1.31E-07	

Table 2: The second portion of the output CSV file generated automatically by the algorithm for the sample before it was exposed to the air.

Pad Voltage/V	Photocharge [Fit]/C	Photocharge error [Fit]/C	Alpha1	Alpha2	Electric Field / (V/m)	
50	7.35E-08	2.35E-09	0.639086058	0.676390091	2958579.882	
60	8.23E-08	2.07E-09	0.646946621	0.747987747	3550295.858	
70	9.46E-08	7.02E-10	0.648875287	0.77013524	4142011.834	
80	1.00E-07	1.84E-13	0.648804622	0.864887398	4733727.811	
90	1.09E-07	9.07E-09	0.639945936	0.914615917	5325443.787	
100	1.12E-07	4.70E-14	0.645768038	1.006815467	5917159.763	
110	1.22E-07	8.60E-09	0.654295231	1.137883733	6508875.74	
120	1.31E-07	1.07E-11	0.660891437	1.071868336	7100591.716	

Table 3: The third portion of the output CSV file generated automatically by the algorithm for the sample before it was exposed to the air.

Pad Voltage/V	Mobility [Fit] / cm2̂ V−1 s−1	Mobility [Knee] / cm2̂ V−1 s−1	sqrt(Electric Field) / (V/m)0̂.5	
50	9.37E-05	0.000118366	1720.05229	
60	9.98E-05	0.000119352	1884.222879	
70	0.000111543	0.000133652	2035.193316	
80	0.000120927	0.00014784	2175.713173	
90	0.000130029	0.000175295	2307.692308	
100	0.000143266	0.000190956	2432.521277	
110	0.000157034	0.00020531	2551.249839	
120	0.000177133	0.000222515	2664.69355	

5 Results

5.1 Electrons: Vacuum Sample

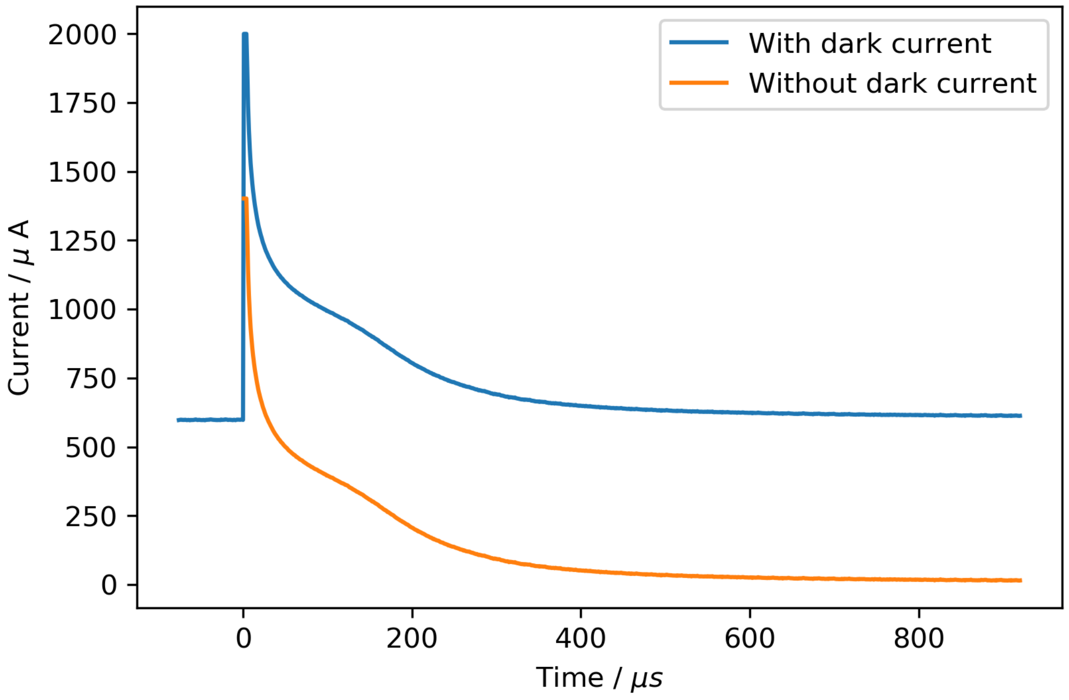


Figure 4: Graph of current measured against the time to measure it with a pad voltage of 110V. The curve is shown before and after the dark current has been subtracted. This graph is a standard output of the algorithm.

Figure 4 shows a fairly typical set of data for the sample whilst it was kept in a vacuum at a pad voltage of 110V and has some important characteristics. The first characteristic you are likely to notice is the shape of the graph with a flat area quickly increasing to a peak at the zeroed time which then decreases with a slope. The level of current at which the flat blue line is seen is the dark current [4] and the yellow line indicates the same set of data when each data point has had the dark current subtracted from its value. The peak that appears is the result of counter carriers - in this case, holes - taking some time to drift to the electrode during measurement and in reality, could appear to be closer to a Dirac delta function but due to the slow instrumental response time in the equipment used to take measurements a gentle slope is instead observed [4].

Next, notice a small kink in the line near the 100μs point which is indicative of the transit time which will become much more pronounced when we produce a double logarithmic plot of this data. Subtracting the dark current from the raw data allows us to produce a double logarithmic plot that isn't skewed and doesn't produce unwanted results. This is shown in Figure 5.

From this raw data with the dark current subtraction, we can use equation 19 to find the photocharge for each value of the pad voltage [4]. The integration has been computed twice using both a cumulative trapezoidal integration algorithm [24] which works numerically from the raw data and an algorithm that creates a polynomial regression of the data [25] and then uses Gaussian quadrature to estimate the integral of the line [24]. Both of these plots have been included in Figure 6 but there is a negligible difference between their values which leads me to believe that either method is an effective method of calculating the photocharge - although from my experience using the

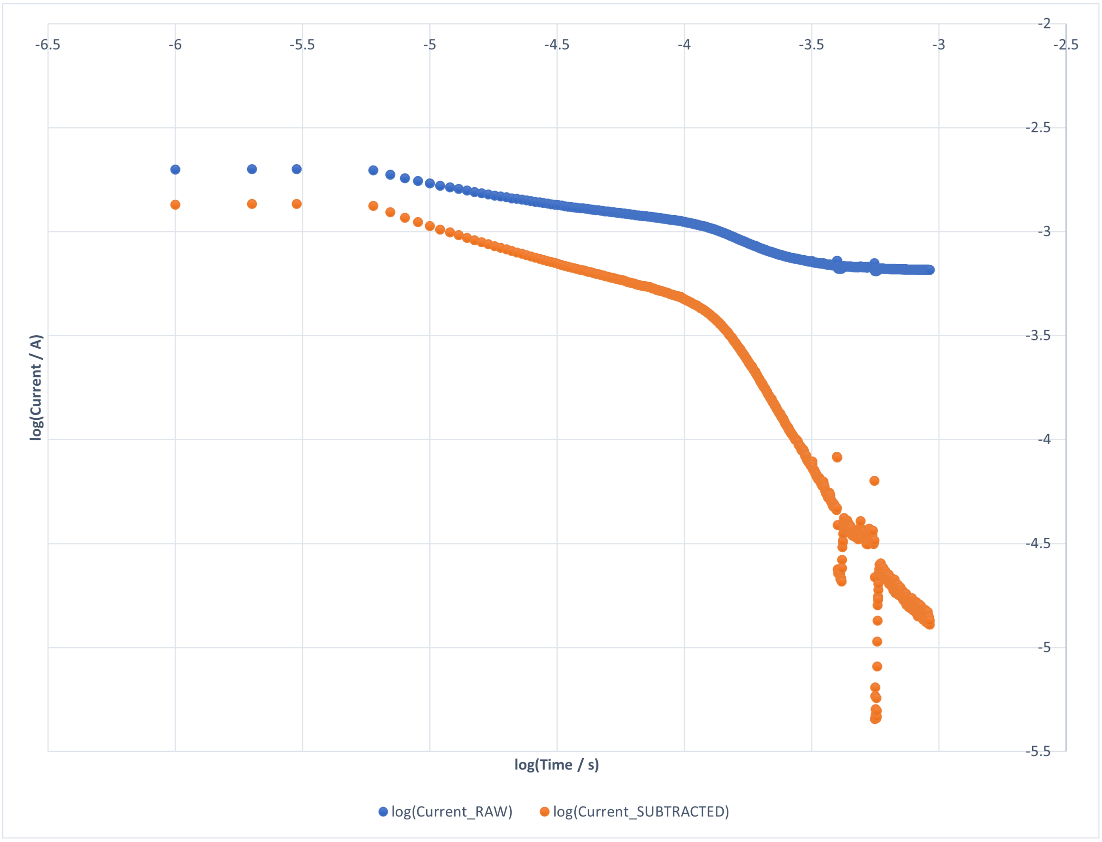


Figure 5: Double logarithmic plot of current against time for the sample before it was exposed to the air at a pad voltage of 120V where the blue data shows the trend before the dark current is subtracted and the orange data shows the trend after the subtraction has occurred. It is clear here that the logarithmic nature of the graph amplifies the impact of the subtraction such that not subtracting the dark current leads to a poor fit of the data.

polynomial method can lead to small inaccuracies in the shape of the polynomial which will add unneeded uncertainties into the photocurrent so I would recommend the cumulative trapezoidal integration method for use.

The positive gradient of Figure 6 indicates that the quantum efficiency is increasing linearly with the pad voltage which tells us that $(n\psi) = f(E)$ as explained in equation 20 [4]. This is expected behaviour for the sample and an important point we must make here is that the linear trend appears to intercept the photocharge axis at a non-zero point (namely at about 36 nC) which means that no external field is required for photogeneration in this material which again is a well-understood phenomenon.

Another useful set of information we can gather from this data provided we know the number of photons being absorbed by our sample is the quantum efficiency using equation 20 [4] but I have been unable to gather this information at the moment.

Now that we have investigated the photocharge of our sample under different pad voltages and established the relationship between quantum efficiency and the electric field we can move on to finding the transit time for our charge carriers. We have done this by creating a double logarithmic plot of the current against the time and then by using the kneedle algorithm [22] to find an approximate kneedle point which is then used to find a more accurate time-of-flight by bisecting two linear regression lines [21] which are plotted at an arbitrary range from the kneedle point. A visual example of how the algorithm works for the 110V data is shown in Figure 7. The kink in the line

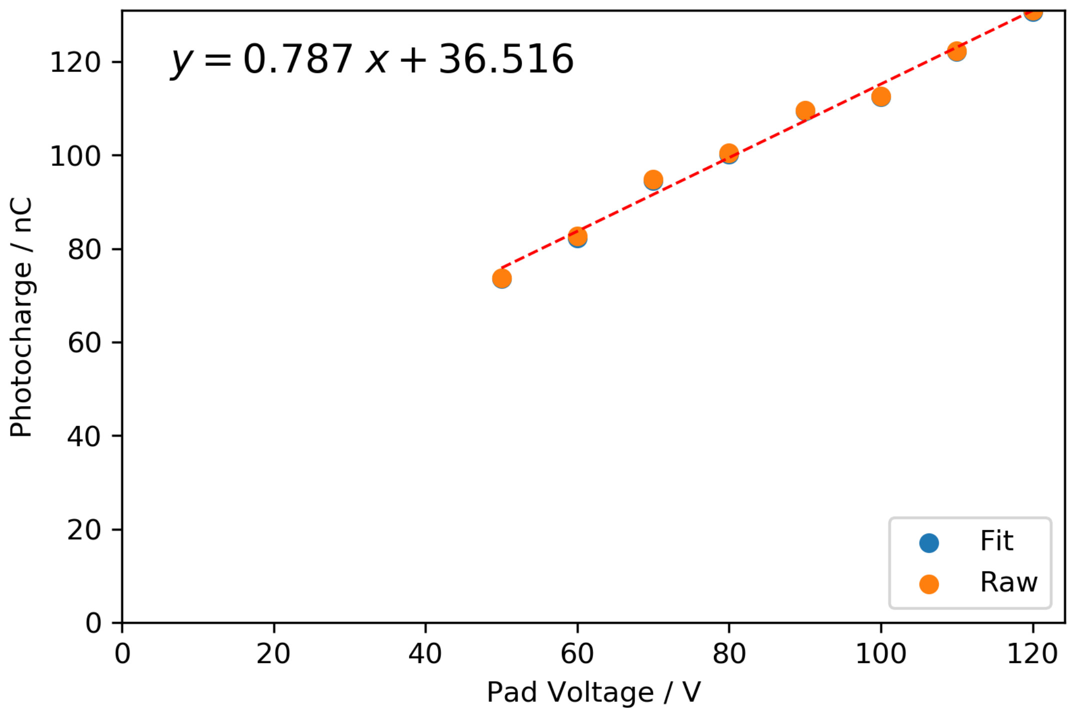


Figure 6: Graph of the measured photocharge against the pad voltage. There is a clear positive linear correlation that doesn't intersect the origin. This graph is a standard output of the algorithm.

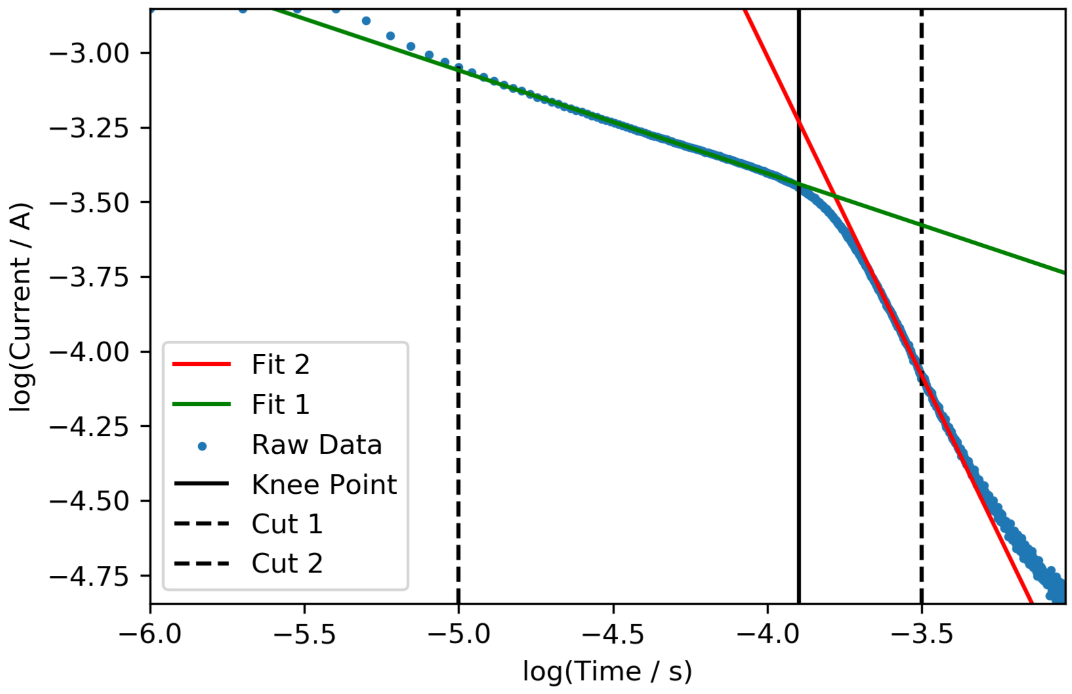


Figure 7: Double logarithmic plot of current against time for a pad voltage of 110V. Notice that the knee fitting algorithm has been given cut points and has then found the knee point using the kneedle algorithm. The knee point is then used to create two trends lines that intersect at the fitted knee point. This graph is a standard output of the algorithm.

that was seen in Figure 4 has been made far more obvious by the double logarithmic nature of this plot. Notice that the cuts used by the kneedle algorithm have been chosen such that they avoid as much curvature and noise as possible as equations 9 and 10 mean that searching for two different straight lines is how we find the transit time and α_1 and α_2.

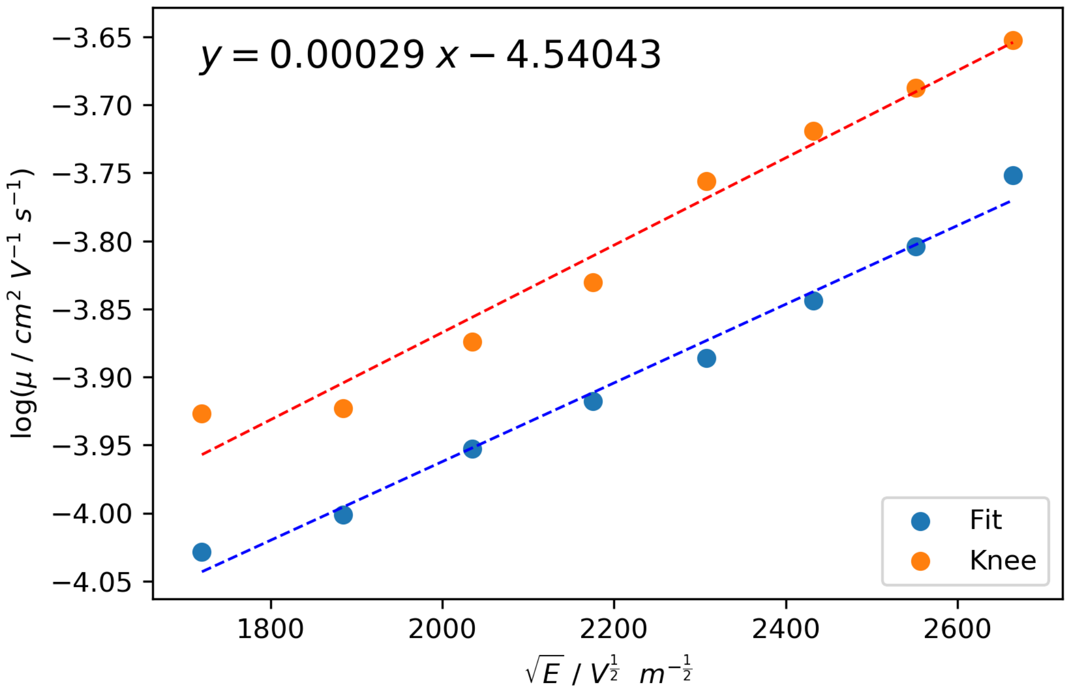


Figure 8: A Poole-Frenkel plot for the vacuum sample data. There is a clear positive linear correlation to the graph as is expected for a typical semiconductor. This graph is a standard output of the algorithm.

We can also use this data to find the mobility using equation 11 [4]. Now that we have the mobility for our sample at a given pad voltage we can create a Poole-Frenkel plot for our data in Figure 8. It is clear from this graph that there is a positive linear correlation between the mobility on a logarithmic scale and the square root of the electric field which indicates that for this sample the mobility is field dependent. This is a result of disorder within the sample which can be analogous to shallow trapping where shallow trapped electrons have trapping which is shallow compared to kT such that they can escape the trap thermally [26]. This means that as the applied electric field is increased the amount of shallow trapped electrons decreases which allows more electrons to carry charge and as such the mobility increases too. This is fairly typical behaviour for most organic semiconductors and upon comparison with [3] it is clear that our Poole-Frenkel plots are very similar which is to be expected as the analysis is of the same data from the same sample. This indicates that the automated analysis used in my investigation has matched the analysis performed by hand and beaten it in its expeditiousness without a loss in the quality of the data.

It is clear, however, that when the kneedle algorithm is used on its own to find the transit time without the corrections of the bisecting lines it overestimates the mobility albeit recreating a similar trendline to the fitted line. The trendline for the bisection fit matches the fit found in [3] better than the fit using only the kneedle algorithm so once again I am sure that the corrections are necessary to improve the accuracy of the program.

Looking at Figure 9 there is a positive correlation between the pad voltage and the value for α_1 which once again lets us know that the time-of-flight is dependent on the value of the applied electric field as is expected for the sample [14]. The sample has low dispersion as the $\alpha_1 > 0.5$. The range of

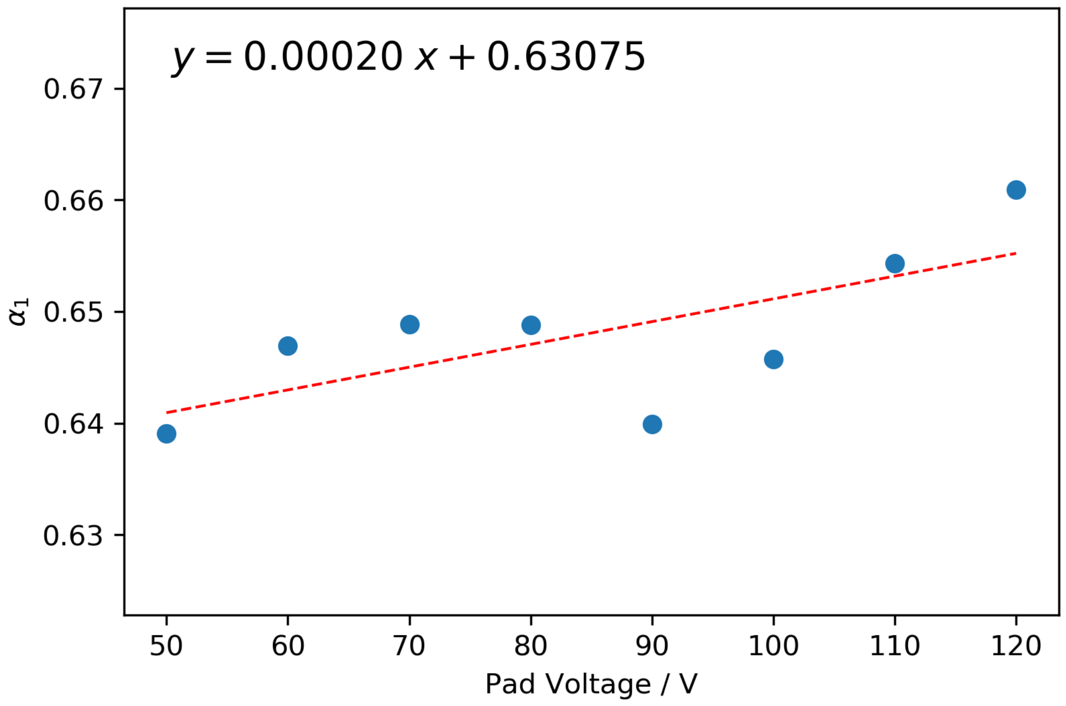


Figure 9: Plot of α_1 against the pad voltage. There is a minor positive correlation here as was expected for the sample. α_1 appears to have a range of roughly 0.02.

the value of α_1 is roughly 0.02 which tells us again that this sample has low dispersion. To check whether we have a separate α_1 and α_2 we should also plot a similar graph for α_2 and see whether our results differ greatly. That graph is Figure 10.

As can be seen in Figure 10, α_2 and the pad voltage are strongly positively correlated which is to be expected but the trend doesn't match that of α_1 so we can be sure that there indeed are two α values. However, at high pad voltages, we have obtained non-physical values for α_2 as α_2 cannot be greater than 1. Since α_2 is derived from the gradient of the second bisection line in Figure 7 we can check that the bisection algorithm is finding an accurate value of the gradients from the graphs by hand and it is. This strange behaviour is explained as α_1 and α_2 are not reliable dispersion parameters [14]. It appears that in our experiment α_1 is acting as a dispersion parameter whereas α_2 is acting non-physically so is not. This could also have been explained as a result of the fact that there is more noise at higher voltages and that would have a large impact as the kneedle algorithm uses a polynomial fit [25] of the raw data to counteract this effect. However, we have chosen to cut the majority of the noise from the dataset before we create the regression so since only curvature present in the regression itself can affect the kneedle algorithm's transit time we can assume that curvature at high voltages has a minimal effect on the gradient of the line derived from α_2.

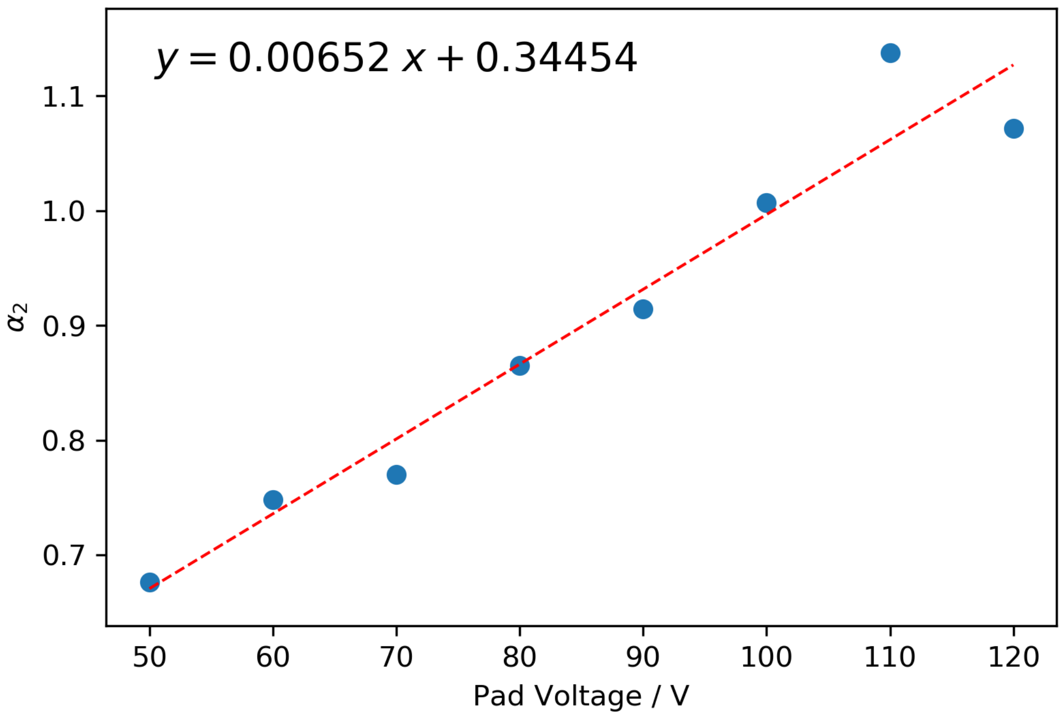


Figure 10: A plot of α_2 against the pad voltage. There is a clear positive correlation here as was expected for the sample but at high pad voltages, there are unphysical values of α_2. This is likely due to the arbitrary method of selecting points to create the intersection lines using the kneedle algorithm.

5.2 Electrons: Exposed Sample

Figure 11 is a good example of the structure of raw data that has been analysed for the sample after it was exposed to the air. The first clear comparison we can make between the sample before and after exposure to air is that the currents and the times measured are much smaller in scale for the sample after exposure. This indicates to us that the sample has far fewer free electrons to conduct current and that the time of flight for the sample has decreased as a result of the reduction in electrons carrying charge [4]. This has likely been caused by an increase in electron trapping caused by highly electronegative regions created when the sample reacts with chemicals in the air such as oxygen [4].

The next notable point is that the dark current (that is subtracted between the two lines on the graph) is far smaller in scale than before the sample was exposed. This indicates greater trapping is occurring in the sample after it has been exposed as a result of increased chemical impurities within the sample creating electron traps [19] [27] [28].

Furthermore, notice that the small peak that was visible in Figure 4 is invisible here. If we were to use only this graph we may believe that no knee would exist in our data but we will show later that a knee does still exist - albeit a much shallower one. This suggests that fewer electrons have been measured which once again supports the hypothesis that increased electron trapping is occurring [19].

Plotting the photocharge values against the pad voltage values for the exposed sample in Figure 12 leads us once again to a positive linear trending curve but with an interesting conclusion as the trendline for the data, once we disregard an outlying point at 160V, passes through the origin unlike

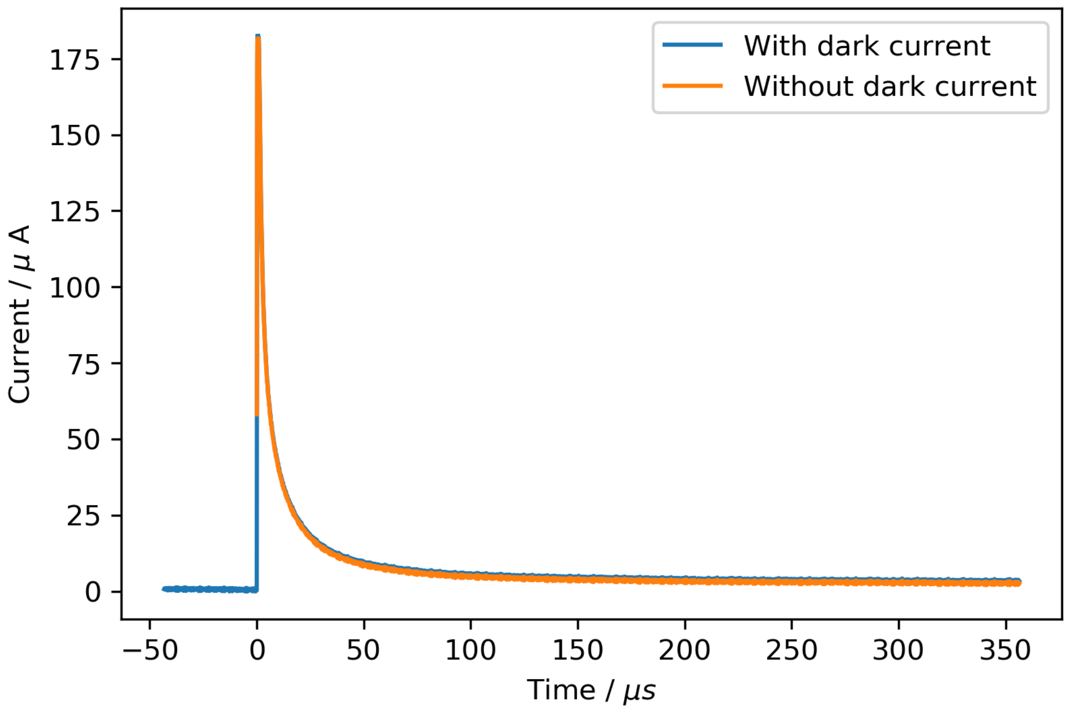


Figure 11: Graph of current measured against the time to measure it with a pad voltage of 110V for the sample after it was exposed to air. The curve is shown before and after the dark current has been subtracted. Notice that the dark current is much smaller in scale than the dark current before the sample was exposed to the air.

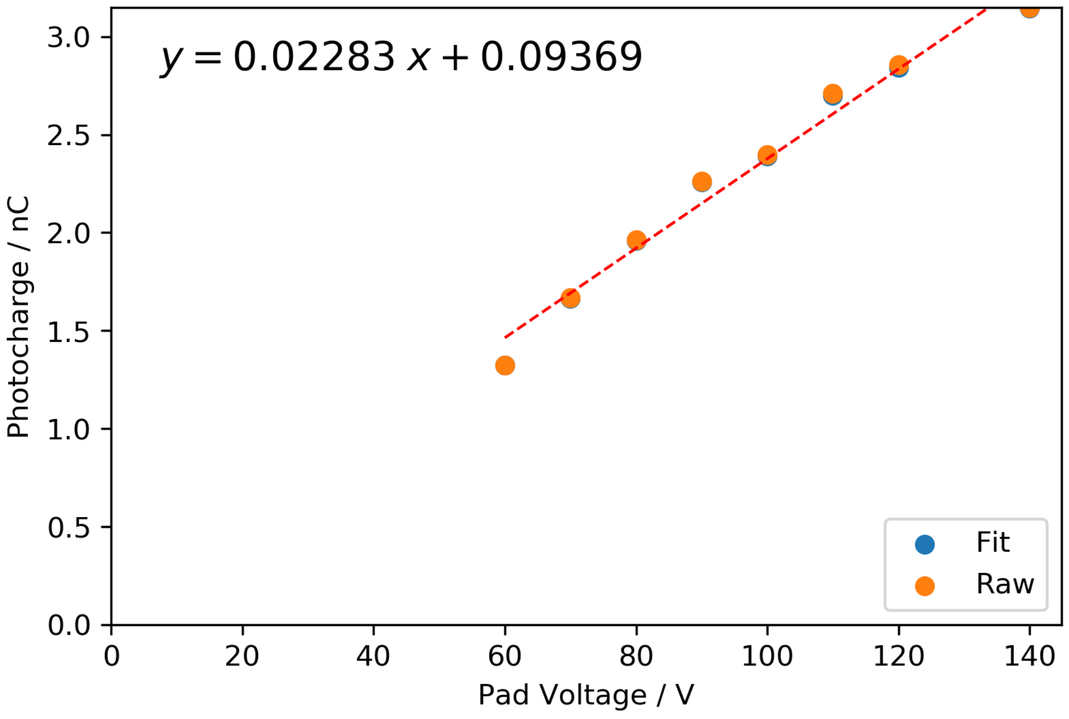


Figure 12: Graph of the measured photocharge against the pad voltage. There is a clear positive linear correlation that intersects the origin. An outlier point has been removed at 160V as it is the only point that doesn't continue the linear trend towards the origin.

the sample before it was exposed. This behaviour change likely tells us that, due to increased trapping, the ability for photogeneration without an external electric field has been removed as a result of the lack of free electrons [19]. In the future, taking more measurement points at lower path voltages would be advisable to ensure this trend towards the origin continues.

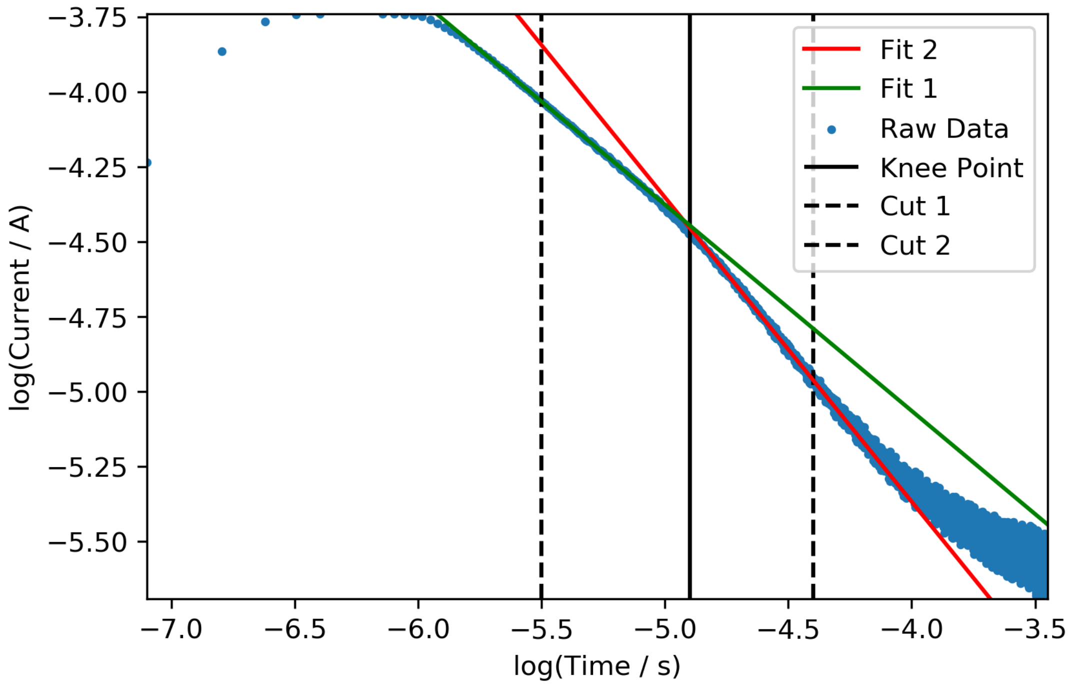


Figure 13: A double logarithmic plot of current against time for a pad voltage of 110V. Notice that the knee is far less pronounced than before the sample was exposed but is still visible. There is also far more noise in this data but the fitting method allows this noise to be ignored.

Notice that Figure 13 has far greater levels of noise at higher values of time than its pre-exposure counterpart and has a far less obvious knee point with the change in slope being far less pronounced. This introduces an extra level of challenge for the bisection algorithm as the user failing to provide cut values which avoid general curvature in the polynomial regression of the raw data can lead to strange and inaccurate results for the transit time and sometimes even leads the kneedle algorithm to find a knee at the endpoint of one of the cuts. When this occurs the only remedy is to choose a new set of cut values and, more often than not, it is wise to be conservative with the placement of the cuts in terms of the levels of curvature near the knee point.

However, despite all of the difficulty created by this lack of a clear knee point, the bisection algorithm generally can find an accurate value for the time of flight as can be seen in Figure 13.

A point I shall make that helps with this analysis is that there are far fewer points on the left side of the graph than on the right side of the graph which means that any strange curvature which occurs on the left side (usually caused by instrumental errors in early measurements such as slow response times) can generally be ignored. This helps us to justify the lower cut ignoring any curvature on the left. Curvature on the right side of the graph is ignored as often it is as a result of noise at high pad voltages and because equations 9 and 10 rely on the presence of straight lines on either side of the knee point [14].

It is also helpful that, due to the use of a polynomial regression [25] to simplify the trend of the data, the noise at high pad voltages tends to have little impact on the trendline which means it also has very little impact on finding the knee point or the transit time.

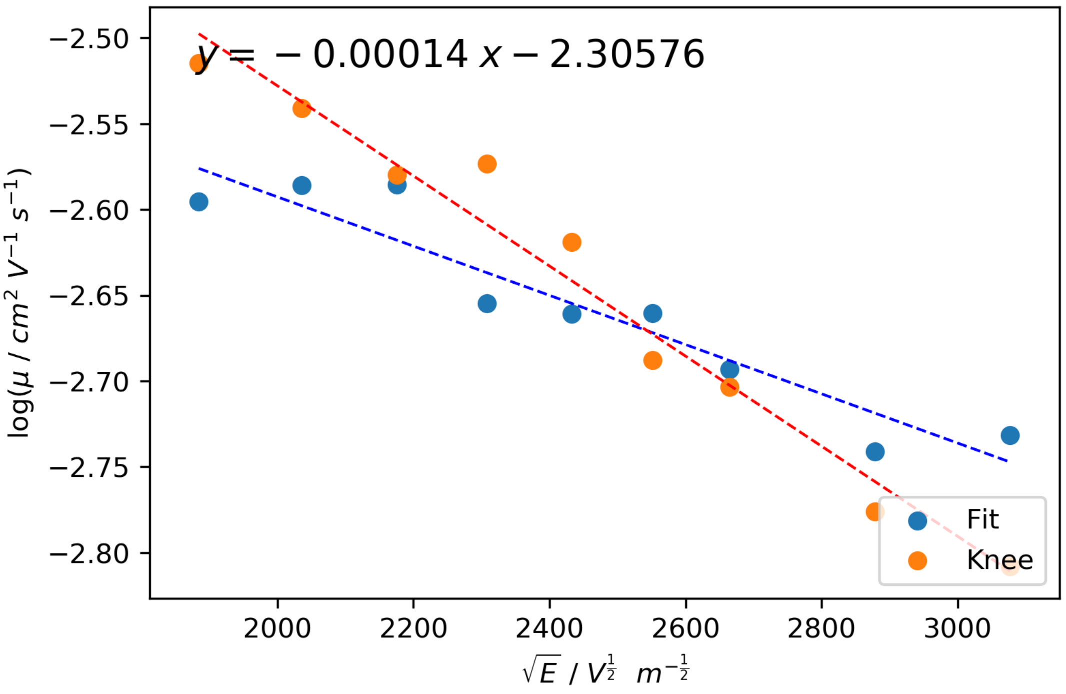


Figure 14: A Poole-Frenkel plot for the sample after it was exposed to the air. There is a clear negative linear trend in the graph which is strange but not unexplained in the literature.

When we create a Poole-Frenkel plot for our sample after exposure to the air we find a striking difference with Figure 14 in that we find a negative coefficient. This is an unusual occurrence as generally Poole-Frenkel plots have a positive slope as a result of an external electric field helping reduce electron trapping [14] but this trend seems to suggest quite the opposite - that increasing the strength of the external electric field actually increases the amount of trapping. It is thought this may be caused by the creation of hot electrons in the sample by the chemical impurities introduced during exposure to air [26] which means that the external electric field effectively increases the depth of electron traps in the sample or this may simply be an artefact in the experiment.

Similarly, we also find for the exposed sample in Figure 15 that the value of α_1 decreases linearly as the pad voltage is increased which is an unexpected result given the slight positive trend of α_1 in Figure 9 [14]. It is possible that an unexplained artefact has been measured instead of the arrival time and that we are not measuring a value for α_1 or indeed we may be seeing signs of the pinning of charge carriers. The magnitude of the range of α_1 values has increased from Figure 9 which is a result of the increase in dispersion caused by the sample's exposure to air [14].

Strangely, we see that in Figure 16 the value of α_2 increases as the pad voltage which is completely against the trend of α_1. Notice that, once again, we have unphysical values for α_2 which we can explain by the fact that α_1 and α_2 are not reliable dispersion parameters [14]. It is believed that the reason the behaviours of α_1 and α_2 differ so much here is as a result of a skew in the data as a result of dispersion in the data. α_1 is measured from lower values of time than α_2 and the dispersion in the sample affects the section of data with larger times more than it affects the data with smaller times. This means a skew is created that makes values of α_1 appear more negative and α_2 appear more positive. From this data, I think it becomes obvious that α_1 and α_2 are not very reliable ways of indicating behaviour within the sample (although α_1 did behave well for the

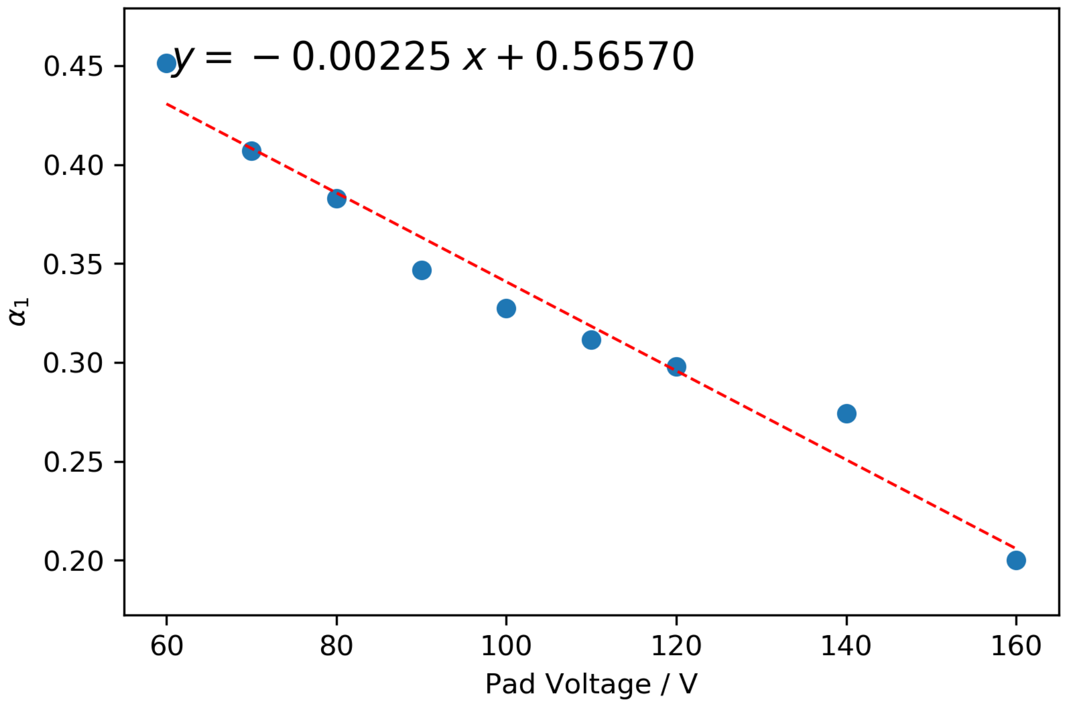


Figure 15: A plot of α_1 against the pad voltage. There is a clear negative correlation here which was unexpected but shall be explained below.

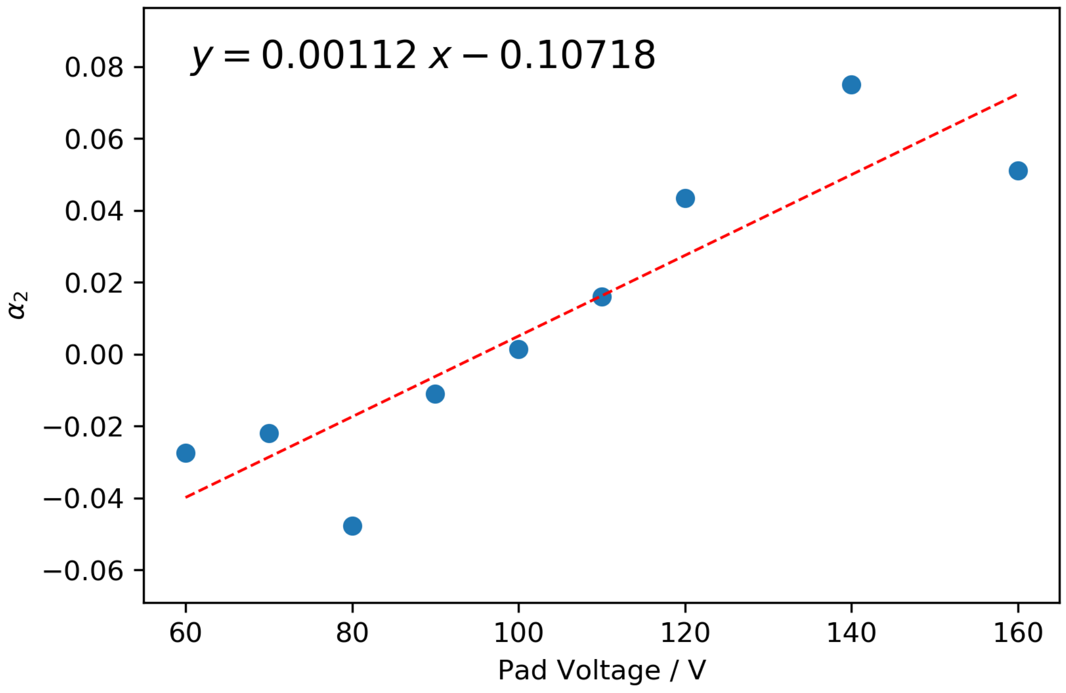


Figure 16: A plot of α_2 against the pad voltage. There is a minor positive correlation here as was expected for the sample. Some of the lower values are unphysical.

vacuum data, α_2 provided unphysical values) and that the assertion that they do not behave well as dispersion parameters is indeed correct [14]. Overall, this suggests that the knee found by the algorithm does not always act as an arrival time and sometimes is influenced by artefacts.

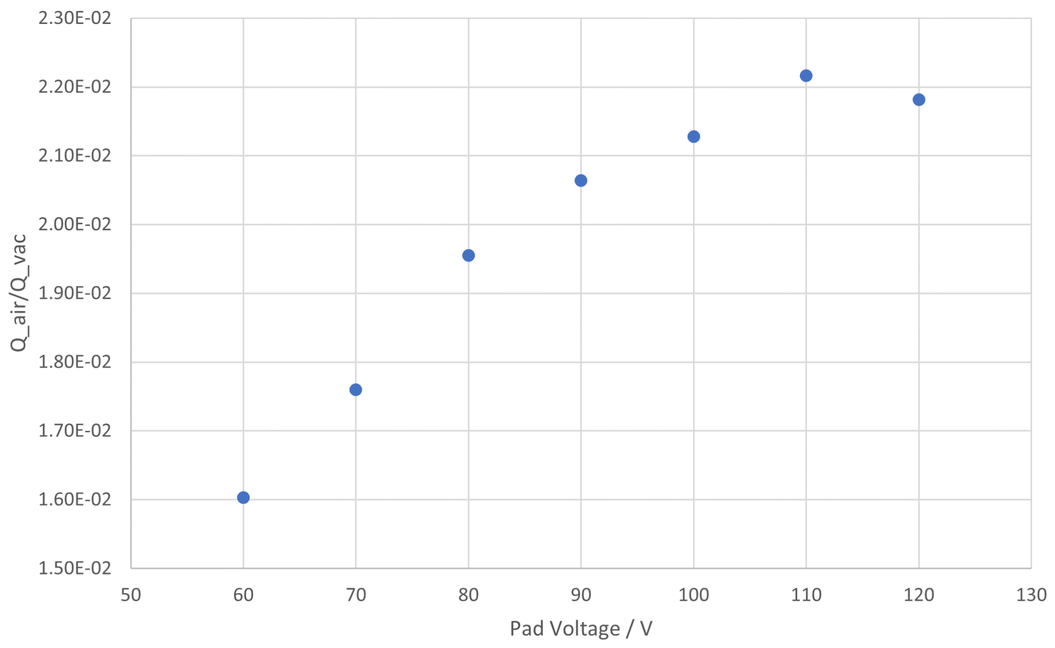


Figure 17: A plot of the ratio between the photocharge of the sample after exposure with the photocharge before exposure against the pad voltage. It shows us a concave downwards trend which is to be expected as a result of the photocharge trend for the non-exposed sample not passing through the origin.

Figure 17 shows us that the fact that the two trends in photocharge against pad voltage don't both pass through the origin creates a non-linear trend between their ratio and the pad voltage. This suggests an artefact may be interrupting our measurements of the photocharge.

5.3 Holes: Vacuum Sample

A hole is the absence of an electron in a place or an atom and is treated as a positively charged charge carrier when dealing with semiconductor physics [4]. Our experiment involved taking time-of-flight measurements for holes as well as electrons but my analysis cannot be as robust with space charge data (in this case for holes) as it exhibits very different behaviour to dispersive data (in this case for electrons) in time-of-flight experiments in that no knee forms in the double logarithmic plot of current against time in hole data [29]. To illustrate this look at Figures 18 and 19.

In Figure 18 there is a space charge hump [30] clearly visible near the $400\mu s$ mark which when we move on to the double logarithmic Figure 19 we can see has been amplified into a large peak. The peak of this point is the transit time for this space charge data. The curvature of the entire graph is different in shape to Figures 7 and 13 so it would make no sense at all to attempt to find the transit time using the bisection method as used for the electron data particularly as the algorithm makes no attempt to find a peak in the trend.

One part of the code does still provide us with information on the behaviour of holes, however, and that is the section of the code that finds the photocharge as this only requires a set of data that it can integrate. As such, we can create Figure 20 which demonstrates that a strong positive linear relationship between the photocharge and the pad voltage exists for holes as well as for electrons. This is unsurprising but provides us with a logic test for the photocharge algorithm which has been completed successfully and tells us that the quantum efficiency is field-dependent for holes as well

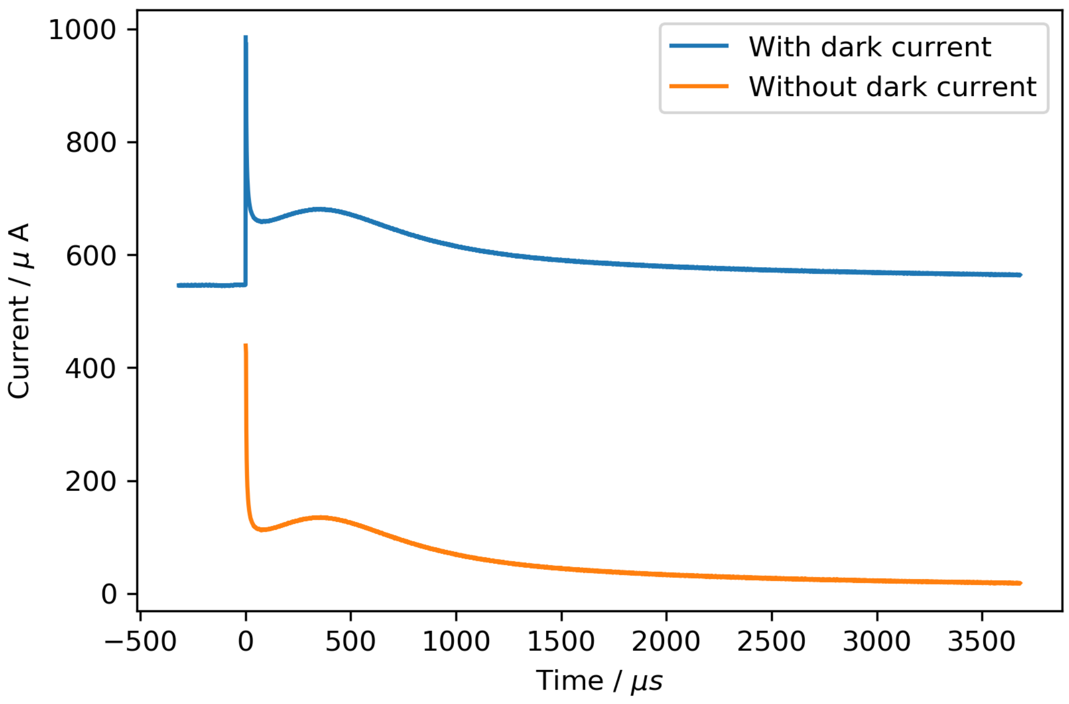


Figure 18: A plot of the raw hole data for the vacuum sample at a pad voltage of 60V. Notice that there is a large dark current of about 550μA and that a hump appears after the initial peak of the graph.

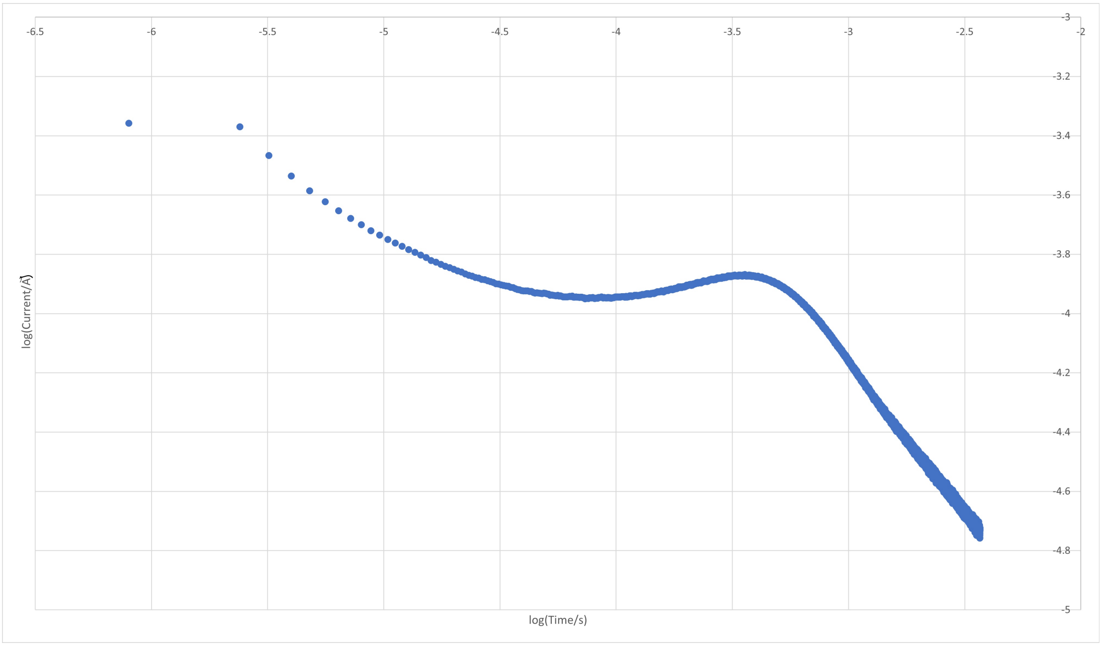


Figure 19: A double logarithmic plot of the hole data for the vacuum sample at a pad voltage of 60V. Notice that the overall shape of the graph is different, involving a localised peak in the distribution caused by the hump seen in Figure 18 which means that using a knee finder algorithm for this dataset would provide poor results.

as for electrons [4]. As we get a negative value for the intercept of the trendline it is unclear whether photogeneration with no external field is possible here. I believe it would be possible to identify the time-of-flight for this trend using an algorithm that searches for peaks in the data but I, unfortunately, did not have the time to create an algorithm that could do this.

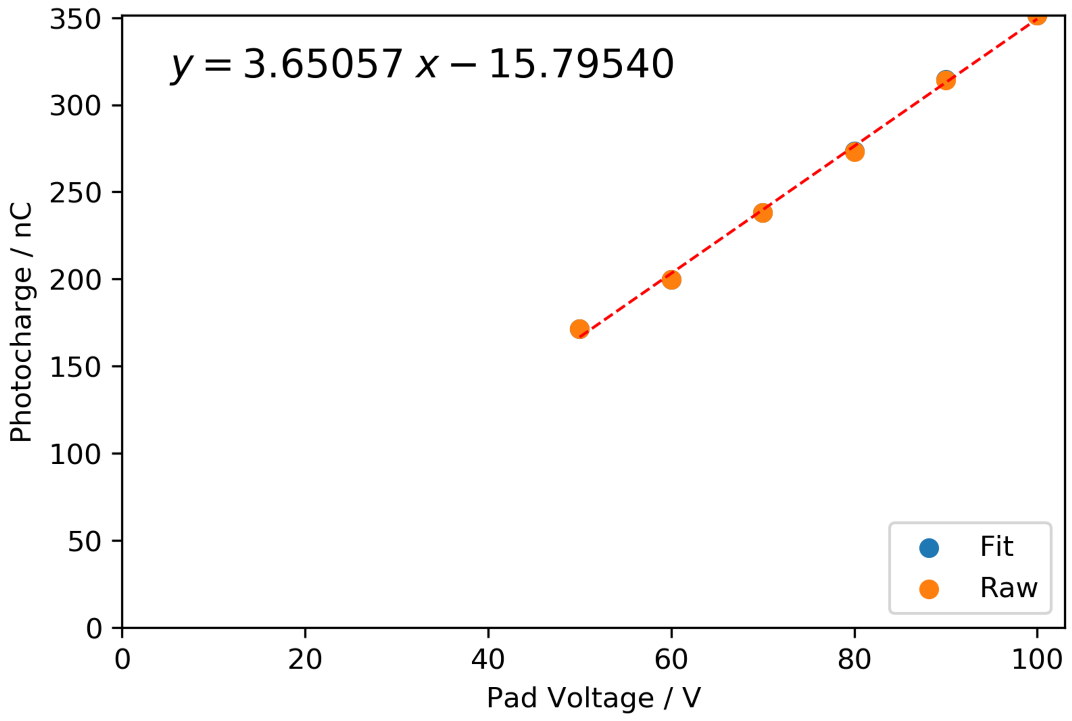


Figure 20: A plot of the photocharge against the voltage. There is a clear positive linear correlation which seems to head through the origin.

5.4 Holes: Exposed Sample

Similar measurements were taken for the sample after it had been exposed to the air, and as is shown in Figure 21, no dark current was detected - indeed the dark current for the holes data was frequently negative due to negative noise causing the average current below $t = 0s$ to be negative. Notice that the hump visible in Figure 18 is no longer visible as a result of hole trapping caused by exposure to air [27].

When we look at the double logarithmic plot for the holes after the sample was exposed to the air in Figure 22 we find that no peak is visible at all as a result of the exposure unlike Figure 19. This makes finding the time-of-flight for this dataset nearly impossible with a peak finding algorithm and I would suggest that the best method for finding a time-of-flight with this data would likely be by hand.

Fortunately, we can still investigate the relationship between the photocharge and the pad voltage in Figure 23 which once again shows us a positive linear relationship with a non-zero intercept similar to Figure 6 once again showing us that the quantum efficiency is dependent on the external field [4]. The trend does not pass through the origin here but the photogeneration at no external field is small at around $3nC$ which suggests that this is an artefact as we do not expect the sample to begin displaying photogeneration at no external field after it has been exposed to air.

Once again we can decide to make a graph of the ratio between the photocharge after exposure and before exposure against the pad voltage and we find an interesting trend in Figure 24 which appears to be a reversed trend than for electrons in Figure 17. It is thought this opposite effect could be a result of the negative values of dark current found by the algorithm as a result of the

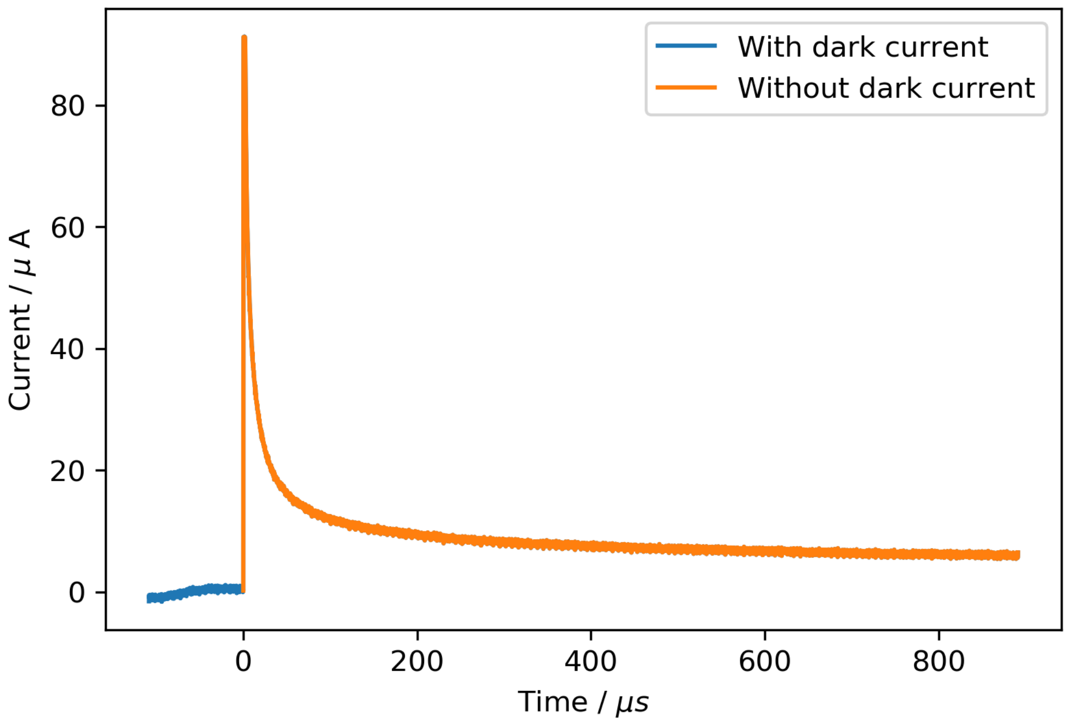


Figure 21: A plot of the raw hole data for the sample after it was exposed to the air at a pad voltage of 60V. Notice that there is a very small (if existent) dark current and that no hump is clearly visible on this graph after the main peak unlike Figure 18.

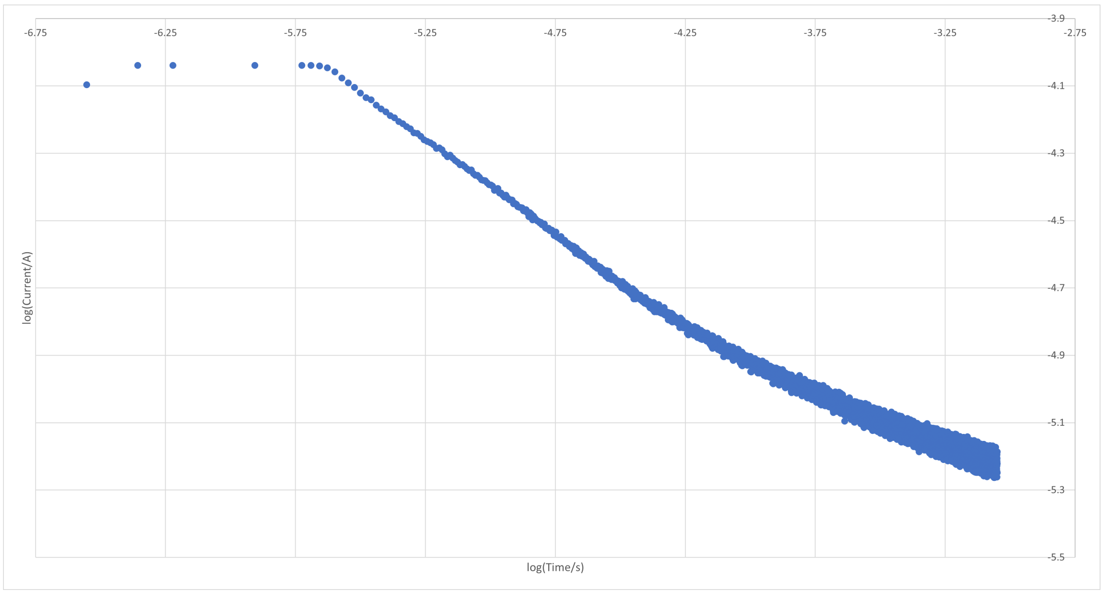


Figure 22: A double logarithmic plot of the hole data for the sample exposed to air at a pad voltage of 60V. There is little evidence of a local peak in the distribution because of the increase in dispersion as a result of exposure to air.

noise in the hole dark current data or this could be an artefact in the data.

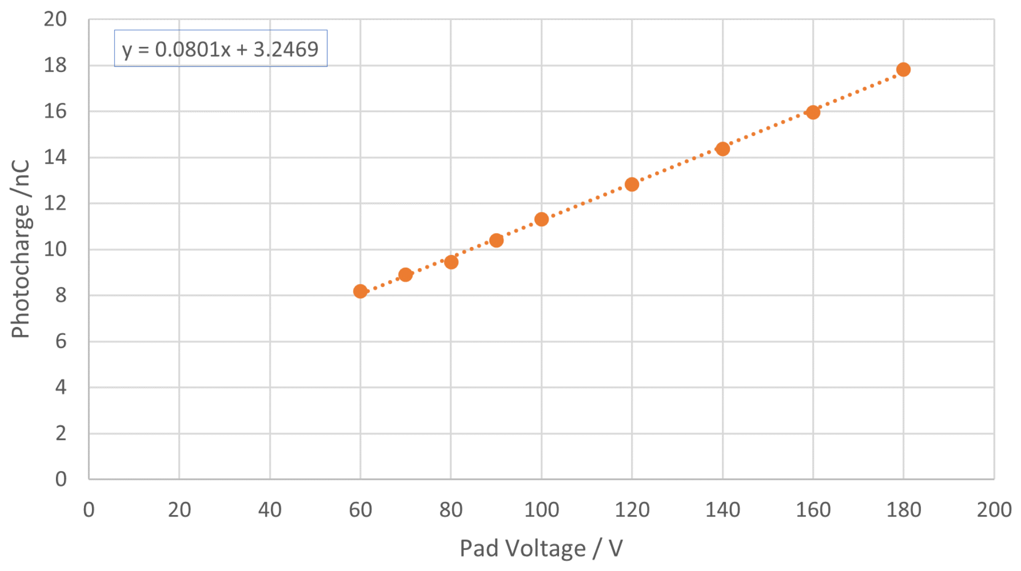


Figure 23: A plot of the photocharge against the pad voltage. There is a clear positive linear correlation.

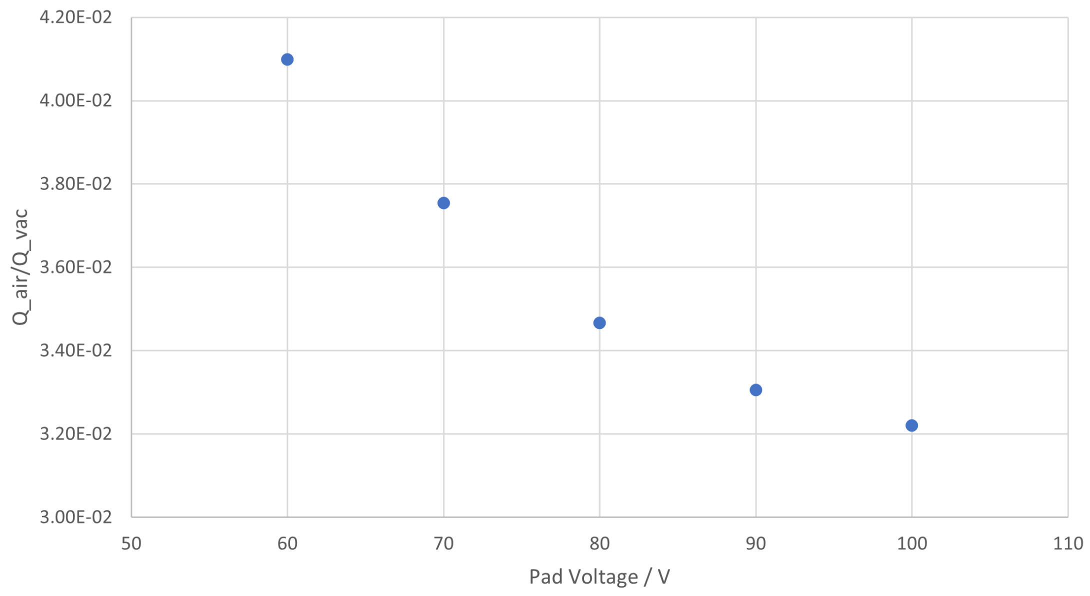


Figure 24: A plot of the ratio between the photocharge of the sample after exposure with the photocharge before exposure against the pad voltage. It shows us a concave upwards trend.

5.5 Dark Current Trends

When we create an IV graph for the vacuum data in Figure 25 of the dark current for each pad voltage allowing the hole data to be positive and the electron data to be negative we find a trend that appears to resemble the Shockley equation which is shown as equation 22 [12]. This is unsurprising as our semiconductor material can act as a diode which means physics akin to the Shockley equation should be observed [12]. The fact that we have an additional tail in the negative area of the graph could simply be due to the slow instrumental response time increasing the size of the

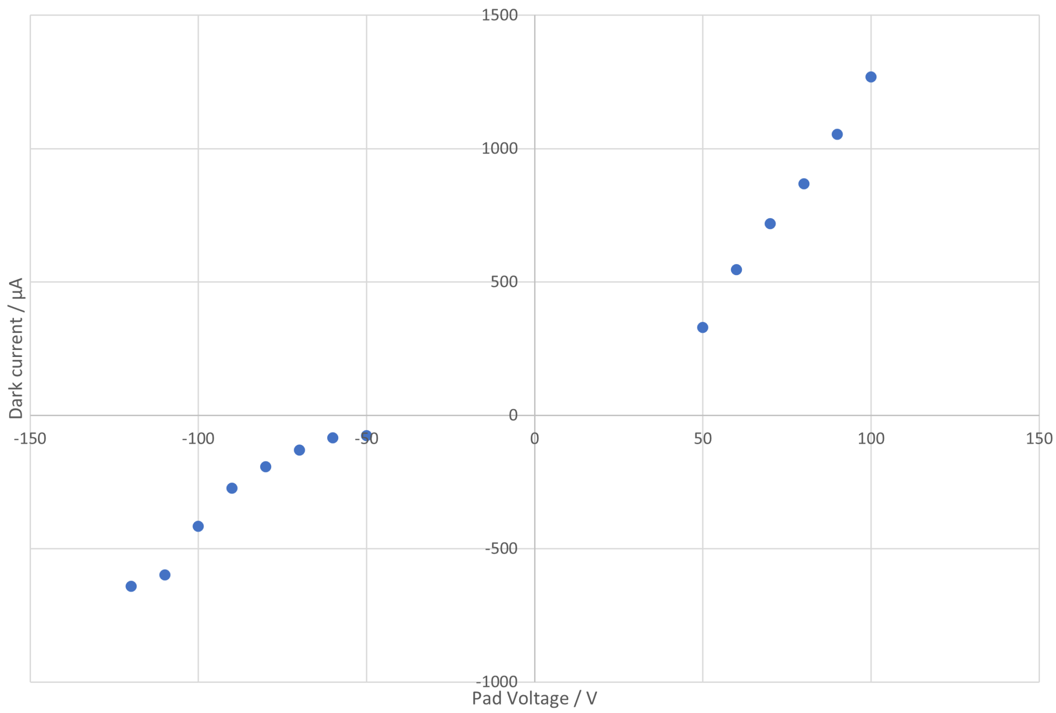


Figure 25: An IV plot of the dark current against the pad voltage for the sample before it was exposed to the air. The negative points correspond to electron data and the positive points correspond to data taken for holes. It is thought that an exponential trend in accordance with Shockley's Law shown in 22 is seen here with a small tail of data not adhering to Shockley's Law at the points with the most negative pad voltages.

integral of the line in the early part of the measurement at higher pad voltages. When I attempted to fit the data to an exponential curve in Microsoft Excel, the trend line provided was not a good fit for the data so this suggests that the data may not match the data as well as expected but this again could have been caused by artefacts in the data. It is unfortunate that the data I had from experiment did not include values below 50V of pad voltage as it makes it impossible to definitively say whether or not this trend is indeed indicative of the Shockley equation.

$$I = I_s(e^{\frac{V_D}{nV_T}} - 1) \tag{22}$$

where I is the diode current, I_s is the reverse saturation current, n is the emission coefficient, V_D is the voltage across the diode, and V_T is the thermal voltage [12].

When we create the same graph for the sample after it was exposed to the air in Figure 26, it becomes clear that no relation to equation 22 applies after the sample has been exposed to air and that the overall trend of the graph seems to have reversed compared to Figure 25. This is likely because the dark currents that were being measured by the algorithm were in fact noise as several of the dark currents for the hole data are negative which is unphysical which suggests that the sample has a negligible dark current and that we are merely picking up noise.

Although I have been unable to perform further analysis on space charge data due to time constraints, a free, open-source tool for performing quick analysis on space charge data does exist [15] and could be used in the future to improve the workflow for working with space charge data. If one wanted to use the algorithm used in this report with that of [15] one can now analyse both dispersive and spacecharge time-of-flight data quickly and easily which opens the possibility of performing

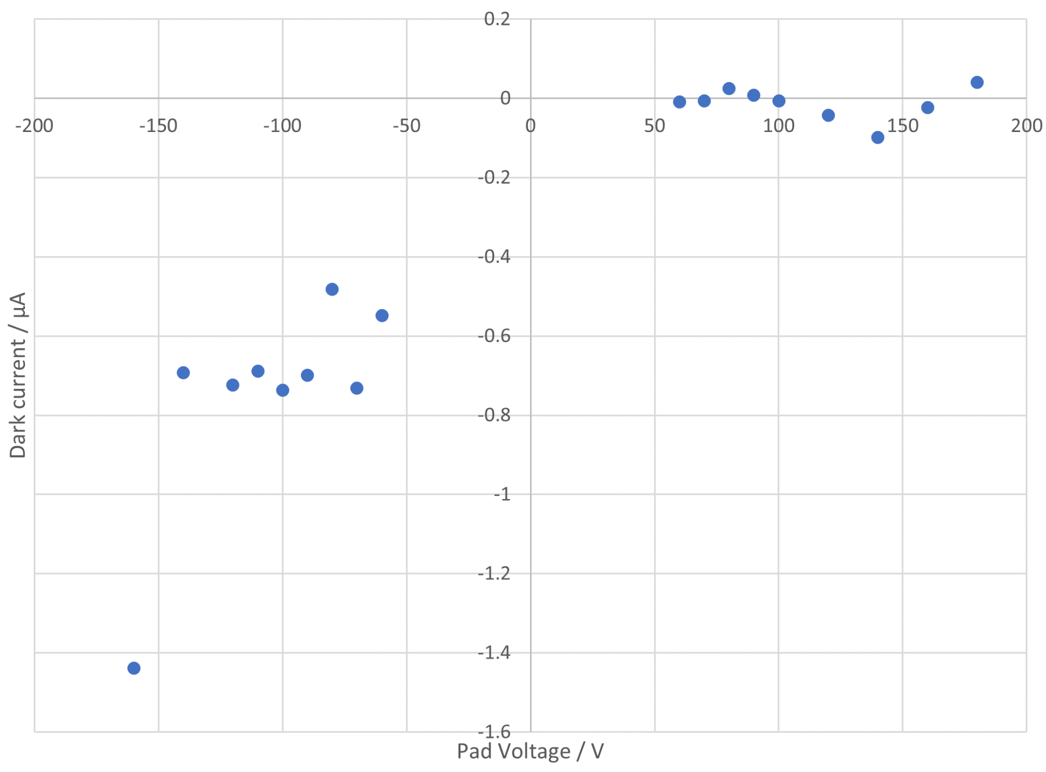


Figure 26: An IV plot of the dark current against the pad voltage for the sample after it was exposed to the air. The negative points correspond to electron data and the positive points correspond to data taken for holes. It is obvious that equation 22 is not adhered to after exposure to air and indeed it looks like the trend may have been reversed compared to the vacuum data Figure 25.

other research instead of slowly analysing time-of-flight data by hand.

5.6 Algorithmic Phenomena

Key decisions in the construction of the algorithm were made based on an approach of testing which methods worked better than others. An important example of this was noticing that the kneedle algorithm [22] would repeatedly underestimate the time-of-flight for the sample before it was exposed (as this was the first set of data I tested my algorithm on) simply by looking at graphs such as Figure 7 and noticing that the black line corresponding to the knee point detected by the kneedle algorithm was typically slightly to the left of the knee I could see with my eye. It was also noted that the mobilities found by the kneedle algorithm would slightly differ from those found by hand in [3]. I decided then to create the bisection algorithm to improve the accuracy of the time-of-flight which seemingly did a better job of finding the knee point and agreeing with [3] so I decided to plot a graph comparing the time-of-flight data found by the two separate methods which is shown in Figure 27.

It is clear that whilst a linear trend does exist between the two methods of finding time the kneedle algorithm tends to underestimate the time-of-flight as the gradient of Figure 27 is less than 1.

The same graph has been generated for the time-of-flight data gathered after the sample had been exposed to the air and a very different trend is observed in Figure 28 where an overall positive trend does still exist but the noise in the trend is greater and the size of the discrepancy between the kneedle algorithm time and the bisection algorithm time has increased. This is believed to be

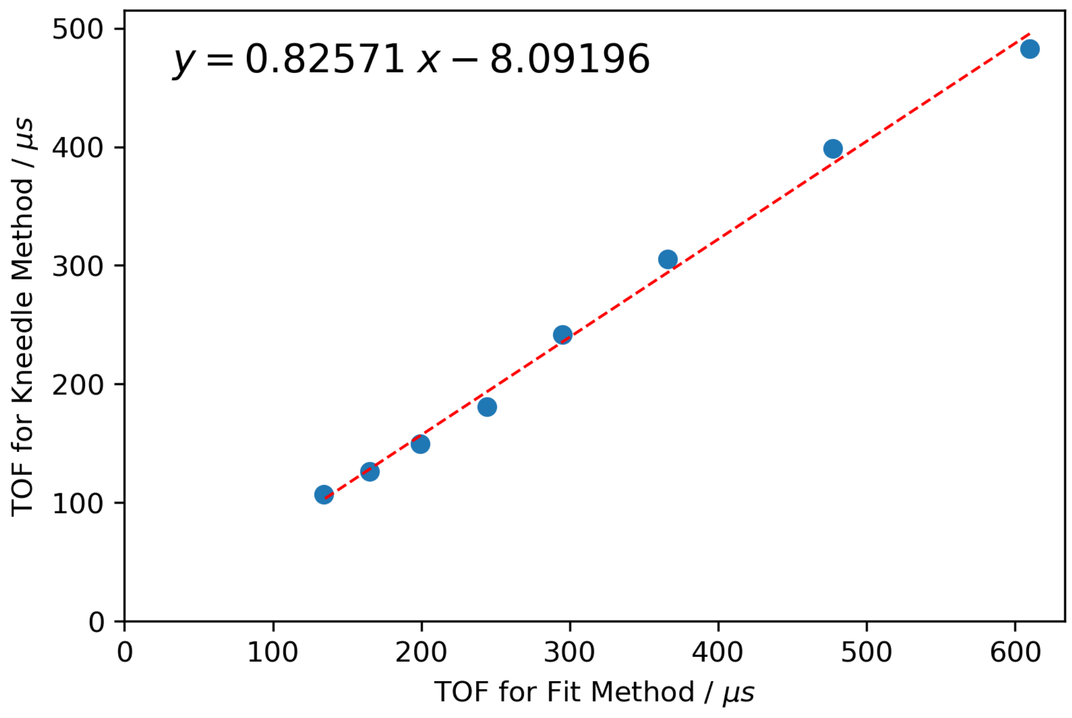


Figure 27: A plot of the time found by the kneedle algorithm against the time found by the bisection method for the vacuum electron data. There is a positive linear relationship between them with the kneedle algorithm frequently underestimating the time-of-flight.

caused by the fact that the knee point in the graphs corresponding to the data for the sample after exposure is far less pronounced than for the data under vacuum which has made it much harder for the kneedle algorithm to find an accurate knee and, as such, the kneedle algorithm tends to underestimate further the time-of-flight for noisy data with a less pronounced knee.

Another important point in selecting how the bisection algorithm would function would be the coefficient used to create the bisection lines. The lines are generated by taking the time found by the kneedle algorithm and using equations 23 and 24 to find t_{min} and t_{max}. The lines are then created by taking linear regressions of the data between t_{min} and t_{max} and the user input cuts that on either side of the knee.

$$t_{min} = t_{kneedle} + log(ct_{kneedle}) \tag{23}$$

$$t_{max} = t_{kneedle} - log(ct_{kneedle}) \tag{24}$$

where c is a coefficient which decides how far in time from $t_{kneedle}$, the time found by the kneedle algorithm, t_{min} and t_{max} should be located. I suspected a value of $c = 0.5$ would work well as the kneedle algorithm never underestimated outside of this range but I decided that the best way to test this would be to create a Poole-Frenkel plot with each different value of c overlaid upon one another. That plot is shown in Figure 29.

Notice that in Figure 29 we find that changing the value of c seems to shift the entire trend of the graph vertically. The scale of the change is not too great but is noticeable. $c = 0.1$ appears lowest on the graph but $c = 0.2$ and onward have a trend that as the value of c increases the graph is shifted downwards. Using the outputted double logarithmic plots similar to Figure 7, I found that this

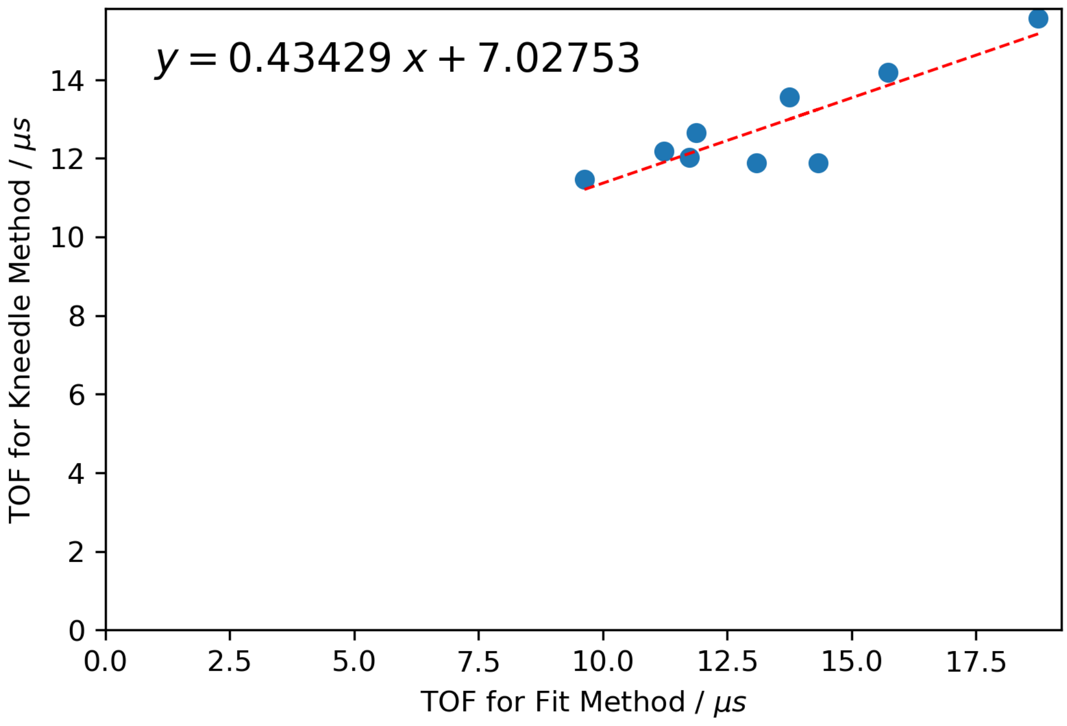


Figure 28: A plot of the time found by the kneedle algorithm against the time found by the bisection method for the exposed electron data. There is a positive linear relationship between them with far more noise present than in Figure 27 with the kneedle algorithm still underestimating the time-of-flight.

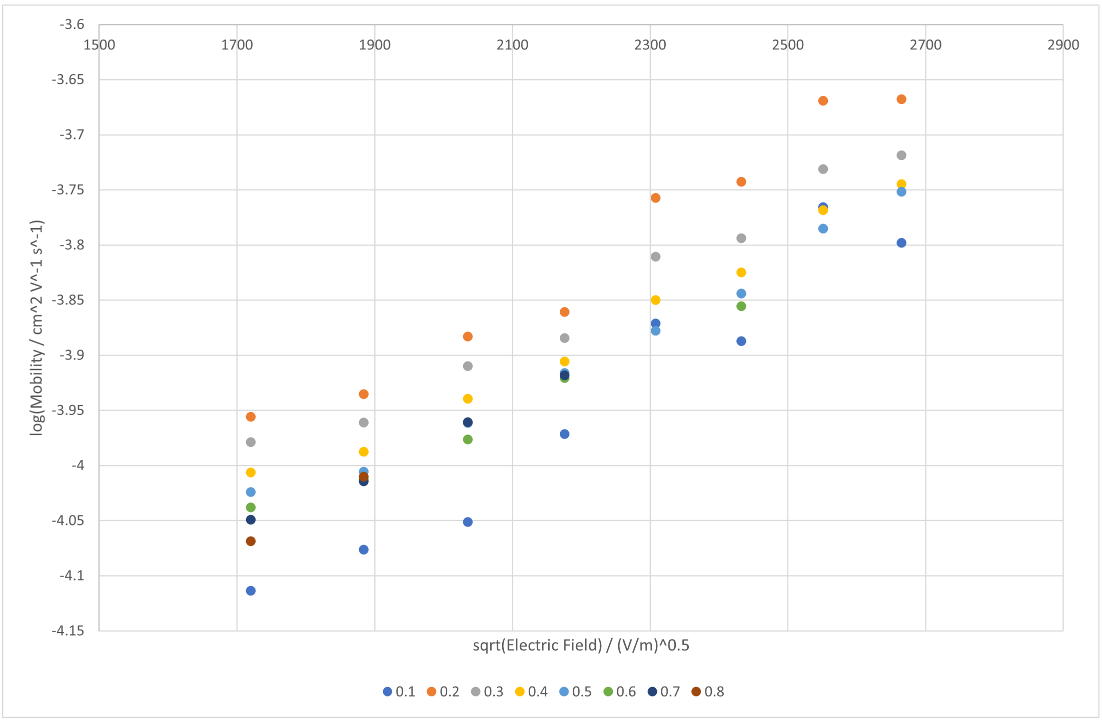


Figure 29: A plot of the differing Poole-Frenkel plots produced by varying the value of the coefficient c. Each value of c appears to shift the entire dataset vertically.

is because at $c = 0.1$ the bisection lines generated were too close to the knee and would provide wildly incorrect lines that did not match the data at all. Another important point to make is that at higher values of c you may notice that some points are not included on the graph as t_{min} and t_{max} would be outside of the user inputted cuts creating an error in the code and preventing you from completing analysis on higher pad voltages which is why some of the higher values of c do not have as many points on the graph as the lower values of c. This ruled out $c > 0.5$ as coefficients above 0.5 were unable to work for all of the data.

When gathering this data I paid close attention to how well the linear regression lines of the bisection algorithm visibly matched the data and it became obvious to me that higher values of c matched the data the best. As I had already decided that values of $c > 0.5$ weren't suitable, I decided to set $c = 0.5$ for the rest of the data analysis. Analysis performed using $c = 0.5$ has been found to replicate the results of the analysis in [3] so I am confident this selection has been successful.

Whilst using equations 23 and 24 to define our bisection lines adds a degree of arbitrariness into our time-of-flight analysis, it allows us to cut out a step whereby the user was forced to input a second set of values for the linear regression which has helped to drastically improve the workflow of the algorithm. As improving the workflow of analysing time-of-flight data was the overall goal of this project I believe that this is a sacrifice worth making.

6 Conclusions

We can be fairly confident the code has worked well as the results found using the code are close to those found in [3] but we could be made more confident if we had the option to test the algorithm against other datasets. I am hopeful the opportunity for this will arrive when members of the time-of-flight research team at Queen Mary University of London's Centre for Condensed Matter Physics get to use the algorithm in their own research. Indeed, it is hoped that this algorithm could be useful for the quick analysis of time-of-flight data in the future and overall improve the workflow for researchers in the field.

We have shown that $P3HT : PC_{60}BM$ behaves very differently before and after exposure to air. It is believed that this is primarily caused by increased electron trapping due to reactions with oxygen that create highly electronegative regions in the samples after exposure [19]. We have also shown that α_1 and α_2 are poor dispersion parameters and are not a reliable way of gauging the processes that are occurring inside of a sample [14].

It was found that the vacuum electron mobility of the sample with no external electric field was about 0.01$cm^2V^{-1}s^{-1}$ with a γ value of $2.9x10^{-5}$, after exposure to air that rose to about 0.1$cm^2V^{-1}s^{-1}$ with a γ value of $-1.4x10^{-4}$. That rise in mobility is a result of the negative field dependence seen in Figure 14 which suggests that the knee for exposed data may not be an arrival time. For the vacuum data, an α_1 range of about 0.02 with an α_1 value of around 0.65 demonstrated that the sample had low dispersion before exposure and after exposure, this increased to 0.25 although α_1 didn't behave well with pad voltage. α_2 was found to have behaved non-physically before (with values between 0.6 and 1.2) and after exposure (with values between -0.06 and 0.08). It was found that the sample had about 36.5nC of photogeneration occurring for electrons without an external field and after exposure this ability to photogenerate without an external field stops. For holes but it was not definitive whether photogeneration occurs without a field before and after exposure.

In terms of the program and its ability to function, I believe we have demonstrated, with a degree of care in the choice of cuts and the value of the coefficient c, that the program can match the accuracy of the measurement of the time-of-flight with the benefit of a far quicker workflow but at the expense of the introduction of a degree of arbitrariness in the creation of the bisection lines. It is worth mentioning that repeating measurements with the program using different choices for the user inputted cuts can have a marked impact on the results of the experiment but this is believed to be a result of a change in the amount of unwanted curvature in the data or as a result of a lack of a gap in time between the cuts and the knee point of the graph as when possible it is better to take measurements over a full generation.

Future versions of the code could include the ability to search for the time-of-flight in hole data before exposure to air using a peak detection algorithm which would allow us to further investigate the time-of-flight properties of the sample in which the user could input whether they wanted the algorithm to study the data using methods for holes or electrons. This change is not completely needed as space charge data can be analysed quickly using the algorithm provided in [15] but having the ability to analyse different types of data in one program could be useful. The code could be widened such that the user doesn't have to repeat each separate value of the path voltage but this would likely involve automating the cuts before the knee algorithm (potentially by using machine learning to identify the best fits for different kinds of trends) is used which could create mistakes in finding the knee and force the user to repeat their analysis which would defeat the purpose of improving the workflow so I am unsure this would actually improve the usability of the algorithm.

The code could include a user input as to the number of photons hitting the sample so that the code could calculate a quantum efficiency for the sample using equation 20 and then explore the behaviour of the quantum efficiency by generating plots involving that efficiency. I would like to further probe the relationship between the photocharge before and after exposure and the behaviour causing the trends shown in Figures 17 and 24 as whilst a suggestion of the cause of this behaviour has been given I would like to see whether this behaviour is repeated in other samples as this would help us to check whether or not this is indeed caused by artefacts in the measurements of the dark current.

Extensions to the experiment could involve comparing the properties of this sample with those of other suitable candidates for radiation detection to find the best possible balance of expense and usability for different industries. We could also test the samples using different lasers to investigate how the frequency of the laser impacts the data. Furthermore, we could also take data for higher and lower values of the pad voltage as that would help us to differentiate whether the IV graphs we have created for the dark current against the pad voltage are indeed following the Shockley equation or whether strange behaviour occurs at larger pad voltages for electrons and holes. It would be interesting to investigate the amounts of trapping occurring at different amounts of time after the sample was exposed to air to see how durable the samples are to the open air and to determine whether the drop in mobility caused by exposure is a sudden reaction or whether it continues long after the first exposure. Repeating measurements and analysis over a set of different temperatures would allow us to investigate the behaviour of the sample under high and low temperatures and let us test to see whether our Poole-Frenkel plots are created well by the algorithm with sets of data at different temperatures.

Overall, I hope that some of the experiments I have suggested will now be able to take place as a result of the improved workflow provided by my algorithm saving researchers precious time during analysis and I am confident that as a result of the research that is being undertaken we will soon see effective and relatively cheap organic semiconductor-based radiation detectors in use in a wide variety of fields and industries.

7 References

[1] Maddalena Binda, D Natali, Marco Sampietro, Tiziano Agostinelli, and L Beverina. Organic Based Photodetectors: Suitability for X-and Γ-rays Sensing Application. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 624(2):443–448, 2010.

[2] Tiziano Agostinelli, M Campoy-Quiles, JC Blakesley, R Speller, DDC Bradley, and J Nelson. A Polymer/Fullerene Based Photodetector with Extremely Low Dark Current for X-ray Medical Imaging Applications. *Applied Physics Letters*, 93(20):419, 2008.

[3] Fani Eirini Taifakou, Muhammad Ali, Joanna Borowiec, Xiaoqi Liu, Peter A Finn, Christian B Nielsen, Cozmin Timis, Tamsin Nooney, Adrian Bevan, and Theo Kreouzis. Solution-Processed Donor–Acceptor Poly (3-hexylthiophene): Phenyl-C61-butyric Acid Methyl Ester Diodes for Low-Voltage α Particle Detection. *ACS Applied Materials & Interfaces*, 13(5):6470–6479, 2021.

[4] Mark Geoghegan and Georges Hadziioannou. *Polymer Electronics*, volume 22. OUP Oxford, 2013.

[5] Nikolay Borodinov, James Giammarco, Neil Patel, Anuradha Agarwal, Katie R O'Donnell, Courtney J Kucera, Luiz G Jacobsohn, and Igor Luzinov. Stability of Grafted Polymer Nanoscale Films Toward Gamma Irradiation. *ACS Applied Materials & Interfaces*, 7(34):19455–19465, 2015.

[6] Akarin Intaniwet, Christopher A Mills, Maxim Shkunov, Heiko Thiem, Joseph L Keddie, and Paul J Sellin. Characterization of Thick Film Poly (triarylamine) Semiconductor Diodes for Direct X-ray Detection. *Journal of applied Physics*, 106(6):064513, 2009.

[7] Beatrice Fraboni, Andrea Ciavatti, Francesco Merlo, Luca Pasquini, Anna Cavallini, Alberto Quaranta, Annalisa Bonfiglio, and Alessandro Fraleoni-Morgera. Organic Semiconducting Single Crystals as Next Generation of Low-Cost, Room-Temperature Electrical X-ray Detectors. *Advanced Materials*, 24(17):2289–2293, 2012.

[8] M Tamura, H Miyata, M Katsumata, K Matsuda, T Ueno, D Ito, and T Suzuki. Beta Particle Detection Efficiency of the Radiation Sensor Made from a Mixture of Polyaniline and Titanium Oxide. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 828:176–180, 2016.

[9] Prodromos Chatzispyroglou, Joseph L Keddie, and Paul J Sellin. Boron-loaded Polymeric Sensor for the Direct Detection of Thermal Neutrons. *ACS Applied Materials & Interfaces*, 12(29):33050–33057, 2020.

[10] Feng Liu, Dian Chen, Cheng Wang, Kaiyuan Luo, Weiyin Gu, Alejandro L Briseno, Julia WP Hsu, and Thomas P Russell. Molecular Weight Dependence of the Morphology in P3HT: PCBM Solar Cells. *ACS applied materials & interfaces*, 6(22):19876–19887, 2014.

[11] Stelios A Choulis, J Nelson, Y Kim, D Poplavskyy, T Kreouzis, JR Durrant, and DDC Bradley. Investigation of Transport Properties in Polymer/Fullerene Blends using Time-of-Flight Photocurrent Measurements. *Applied physics letters*, 83(18):3812–3814, 2003.

[12] Robert B Leighton and Matthew Sands. *The Feynman Lectures on Physics*. Addison-Wesley Boston, MA, USA, 1965.

[13] Harvey Scher and Elliott W Montroll. Anomalous Transit-time Dispersion in Amorphous Solids. *Physical Review B*, 12(6):2455, 1975.

[14] Heinz Bässler. Charge Transport in Disordered Organic Photoconductors. A Monte Carlo Simulation Study. *Physica Status Solidi B (Basic Research);(Germany)*, 175(1), 1993.

[15] Nikolaos Felekidis, Armantas Melianas, and Martijn Kemerink. Automated Open-source Software for Charge Transport Analysis in Single-Carrier Organic Semiconductor Diodes. *Organic Electronics*, 61:318–328, 2018.

[16] Sergei V Novikov, David H Dunlap, Vasudev M Kenkre, Paul Ernest Parris, and Anatoly V Vannikov. Essential Role of Correlations in Governing Charge Transport in Disordered Organic Materials. *Physical Review Letters*, 81(20):4472, 1998.

[17] Jason A Röhr, Davide Moia, Saif A Haque, Thomas Kirchartz, and Jenny Nelson. Exploring the Validity and Limitations of the Mott–Gurney Law for Charge-Carrier Mobility Determination of Semiconducting Thin-films. *Journal of Physics: Condensed Matter*, 30(10):105901, 2018.

[18] Dirk Hertel and Heinz Bässler. Photoconduction in Amorphous Organic Solids. *ChemPhysChem*, 9(5):666–688, 2008.

[19] Davood Abbaszadeh, Alexander Kunz, Naresh B Kotadiya, Anirban Mondal, Denis Andrienko, Jasper J Michels, Gert-Jan AH Wetzelaer, and Paul WM Blom. Electron Trapping in Conjugated Polymers. *Chemistry of Materials*, 31(17):6380–6386, 2019.

[20] Wes McKinney et al. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine Learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[22] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.

[23] John D Hunter. Matplotlib: A 2D Graphics Environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[24] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[25] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane,

Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array Programming with NumPy. *Nature*, 585:357–362, 2020.

[26] Awnish Kumar Tripathi, YN Mohapatra, et al. Mobility with Negative Coefficient in Poole–Frenkel Field Dependence in Conjugated Polymers: Role of Injected Hot Electrons. *Organic Electronics*, 11(11):1753–1758, 2010.

[27] Chen Li, Lian Duan, Haoyuan Li, and Yong Qiu. Universal Trap Effect in Carrier Transport of Disordered Organic Semiconductors: Transition from Shallow Trapping to Deep Trapping. *The Journal of Physical Chemistry C*, 118(20):10651–10660, 2014.

[28] Xiaoqing Guo, Qianxun Gong, Joanna Borowiec, Sijie Zhang, Shuo Han, Meng Zhang, Maureen Willis, Theo Kreouzis, and Kui Yu. Energetics of Nonradiative Surface Trap States in Nanoparticles Monitored by Time-of-Flight Photoconduction Measurements on Nanoparticle–Polymer Blends. *ACS applied materials & interfaces*, 11(40):37184–37192, 2019.

[29] S Barard, M Heeney, L Chen, M Cölle, M Shkunov, I McCulloch, N Stingelin, M Philips, and T Kreouzis. Separate Charge Transport Pathways Determined by the Time of Flight Method in Bimodal Polytriarylamine. *Journal of Applied Physics*, 105(1):013701, 2009.

[30] Weihua Tang, Vijila Chellappan, Minghui Liu, Zhi-Kuan Chen, and Lin Ke. Hole Transport in Poly [2, 7-(9, 9-dihexylfluorene)-alt-bithiophene] and High-efficiency Polymer Solar Cells from its Blends with PCBM. *ACS Applied Materials & Interfaces*, 1(7):1467–1473, 2009.

A Appendix

```python
# Installs kneed if you don’t already have it installed.
import sys
!{sys.executable} -m pip install kneed
```

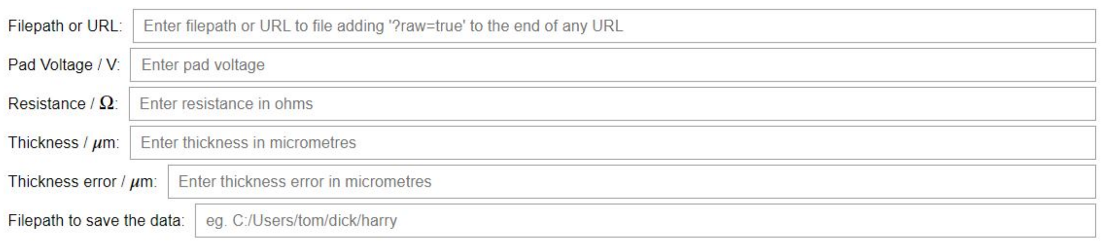
Listing 1: Code used to install the kneed package used for knee finding onto the local computer

```python
# Imports modules which will be used throughout the code.
import pandas as pd
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
import sklearn
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from scipy import stats, integrate
from kneed import KneeLocator
import math
from sympy.solvers import solve
from sympy import Symbol
import os
import ipywidgets as widgets
```

Listing 2: Code used to import the necessary modules for the kernel

```python
# Creates a user interface using the widget module.
widget_width = ’900px’
padding = ’0px 0px 0px 4px’

text0 = widgets.Text(
    value="",
    description="Filepath or URL:",
    placeholder="Enter filepath or URL to file adding ’?raw=true’ to the end of any URL
    ",
    style={"description_width": "initial"},
    layout=widgets.Layout(width=widget_width, padding=padding),
)
text0

text1 = widgets.Text(
    value="",
    description="Pad Voltage / V:",
    placeholder="Enter pad voltage",
    style={"description_width": "initial"},
    layout=widgets.Layout(width=widget_width, padding=padding),
)
text1

text2 = widgets.Text(
    value="",
    description="Resistance / $ \Omega $:",
    placeholder="Enter resistance in ohms",
    style={"description_width": "initial"},
    layout=widgets.Layout(width=widget_width, padding=padding),
)
text2

text3 = widgets.Text(
    value="",
    description="Thickness / $\mu$m:",
    placeholder="Enter thickness in micrometres",
    style={"description_width": "initial"},
    layout=widgets.Layout(width=widget_width, padding=padding),
)
text3

text6 = widgets.Text(
    value="",
    description="Filepath to save the data:",
    placeholder="eg. C:/Users/tom/dick/harry",
    style={"description_width": "initial"},
    layout=widgets.Layout(width=widget_width, padding=padding),
)
text6

text8 = widgets.Text(
    value="",
    description="Thickness error / $\mu$m:",
    placeholder="Enter thickness error in micrometres",
    style={"description_width": "initial"},
    layout=widgets.Layout(width=widget_width, padding=padding),
)
text8

# Sets the output of the user input to variables which can be used in the rest of the
    code.
toolbar_widget = widgets.VBox()
toolbar_widget.children = [
    text0,
    text1,
    text2,
    text3,
    text8,
    text6

]
#Displays the widget.
toolbar_widget
```

Listing 3: Code used to create the user interface used to collect the initial experimental parameters for use later in analysis



Filepath or URL:	Enter filepath or URL to file adding '?raw=true' to the end of any URL
Pad Voltage / V:	Enter pad voltage
Resistance / Ω:	Enter resistance in ohms
Thickness / $\mu$m:	Enter thickness in micrometres
Thickness error / $\mu$m:	Enter thickness error in micrometres
Filepath to save the data:	eg. C:/Users/tom/dick/harry

Figure 30: A screen capture of part of the output of the above cell of code which is the first user input prompt which is shown in its empty form.

```
# Sets variable names for the output of the user interface.
filename = text0.value
Pad_Voltage = float(text1.value)
Resistance = float(text2.value)
```

```python
Thickness = float(text3.value) * 10 ** (-6)
Thickness_error = float(text8.value) * 10 ** (-6)
filepath = text6.value
# Takes output from the user imput filename and adds extra to create new files and
    files.
savename = str(filepath) + "/electron_plate_data.csv"
graph_savename = str(filepath) + "/Graphs"

# If the path does not yet exist on the user's PC it will be created by the code.
if not os.path.exists(graph_savename):
    os.makedirs(graph_savename)

# If a path for files used in the code does not yet exist they will be created.
if not os.path.exists(str(filepath) + "/Intermediate_files"):
    os.makedirs(str(filepath) + "/Intermediate_files")

# By default this reads the csv format data frame (df) ready for analysis and assumes
    there are no headers in the data.
# If there are headers in the data you should delete "headers = None, names = ["Time",
    "Current"]".
df = pd.read_csv(filename, header=None, names=["Time", "Voltage"])

# A new dataframe is created removing the first 11 points to avoid the extra
    information often contained early
# in the csv files. If the code doesn't work here try adding higher numbers as there
    may be even more unwanted
# data at the beginning. Missing a few early datapoints makes a negligible difference
    to the overall results.
df1 = df.drop(labels = [0,1,2,3,4,5,6,7,8,9,10,11], axis = 0)
# Drops any further Not-A-Number values just in case.
df1.dropna()

# The index of the dataframe is reset to ensure the datapoints start at 0 not at 12,
    etc.
df1.reset_index(drop = True, inplace = True)

# Saves the cleaned data into an intermediate file for use later.
df1.to_csv(str(filepath) + "/Intermediate_files/corrected_testdata.csv", index=False)

# Sets the data to datatype 'float64' if it isn't already in this form
df1["Time"] = df1["Time"].astype("float")
df1["Voltage"] = df1["Voltage"].astype("float")
# Resets the index of the dataframe to avoid confusion
df1.reset_index(drop = True, inplace = True)

# Prints the types of data to ensure we definitely don't have any non float64 data
    remaining. This is mainly here for
# troubleshooting.
print(" ")
print("The datatypes of each column:")
print(df1.dtypes)

# Creates a column in the dataframe for the current by dividing each value of the pad
    voltage by the user provided
# resistance.
df1['Current'] = df1["Voltage"] / Resistance
# Removes any infinite values that may have been left in the code by replacing them
    with Not a Number (NaN) values
# and then deleting any NaN values.
```

```
df1.replace([np.inf, -np.inf], np.NaN, inplace=True)
df1.dropna()

# Creates new dataframes for data with times above and below the 0s mark. Works by
    reading the first corrected file and
# writing a new csv file containing no values where the first character is '-' thereby
    removing all negative values.
# This means we can still access the data with negative values whilst also having
    cleaned the initial data.
with open(str(filepath) + "/Intermediate_files/corrected_testdata.csv", 'r') as f:
    with open(str(filepath) + "/Intermediate_files/corrected_testdata_positive.csv", 'w
    ') as g:
        with open(str(filepath) + "/Intermediate_files/correlated_testdata_negative.csv
    ", 'w') as h:
            for row in f:
                if row[0] != '-':
                    g.write(row)
                else:
                    h.write(row)

# Reads the positive and negative dataframes and creates the current values for them
    both.
df2 = pd.read_csv(str(filepath) + "/Intermediate_files/corrected_testdata_positive.csv"
    )
df_negative = pd.read_csv(str(filepath) + "/Intermediate_files/
    correlated_testdata_negative.csv", header=None, names=["Time", "Voltage"])
df2['Current'] = df2["Voltage"] / Resistance
df_negative["Current"] = df_negative["Voltage"] / Resistance

# Removes any infinite values and resets the index of the positive dataframe.
df2.replace([np.inf, -np.inf], np.NaN, inplace=True)
df2.dropna(inplace = True)
df2.reset_index(drop = True, inplace = True)

# The dark current is found by taking the mean value of the current from the negative
    data.
dark_current = df_negative["Current"].mean()
# Each value of the current in the positive dataframe has the dark current subtracted
    from its value.
df2['Current'] -= dark_current
# New columns are created in the dataframe for the logarithms of time and current.
df2['Log Time'] = np.log10(df2["Time"])
df2['Log Current'] = np.log10(df2["Current"])
# Infinite values are once again removed as subtracting the dark current from some
    noise in the positive region leads
# to negative values being put into a logarithm which leads to an error.
df2.replace([np.inf, -np.inf], np.NaN, inplace=True)
df2.dropna(inplace = True)

# A plot of the raw data for current against is produced including the slopes before
    and after the dark current
# has been subtracted and the plot is saved in the specified filepath.
plt.figure()
plt.plot(df1["Time"]*1000000, df1["Current"]*1000000, label = "With dark current")
plt.plot(df2["Time"]*1000000, df2["Current"]*1000000, label = "Without dark current")
#plt.title("Time of Flight Data")
plt.xlabel("Time / $\mu s$")
plt.ylabel(r"Current / $\mu$ A")
plt.legend()
```

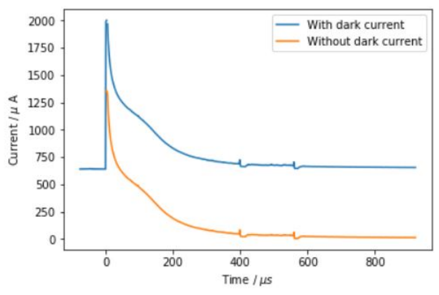
```python
plt.savefig(str(graph_savename) + "/Raw_data" + str(Pad_Voltage) + "V.png", dpi = 300,
    bbox_inches = 'tight')
plt.show()

# A polynomial regression of the raw data is generated. You can change the number of
    free parameters after the
# 2nd comma.
a = np.polyfit(df2["Time"], df2["Current"], 20)
b = np.poly1d(a)

# The polynomial regression is printed out as a linear equation for completeness.
print("The generated polynomial regression:")
print(" ")
print(b)

# This linspace is used to plot the polynomial regression. We can choose how many
    points it plots after the
# 2nd comma.
myline = np.linspace(df2["Time"].min(), df2["Time"].max(), 1000)

# We plot the current v the time again with the polynomial regression displayed to
    check if the regression
# seems sensible.
plt.figure()
plt.plot(df2["Time"]*1000000, df2["Current"]*1000000)
plt.plot(myline*1000000, b(myline)*1000000)
#plt.title("Current against Time")
plt.xlabel(r"Time / $\mu s$")
plt.ylabel(r"Current / $\mu$ A")
plt.show()

# This function uses our polynomial regression to find an integral of the curve and
    finds the error in its
# measurement using Gaussian quadrature.
photocharge, photocharge_error = integrate.quad(b, df2["Time"].min(), df2["Time"].max()
    )
print("Polynomial photocharge:", photocharge, "C with an error of:", photocharge_error,
     "C")
print(" ")
# This function performs a cumulative trapezoidal integral approximation on the raw
    data without the need
# for a polynomial.
y_int = integrate.cumtrapz(df2["Current"], df2["Time"], initial=0)
print("Cumulative trapezoidal photocharge:", y_int[-1], "C")

# A polynomial regression of the double logarithmic data is created ready to be graphed
    .
f = np.polyfit(df2["Log Time"], df2["Log Current"], 10)
p = np.poly1d(f)
myline_1 = np.linspace(df2["Log Time"].min(), df2["Log Time"].max(), 1000)

# The double logarithmic plot is created.
plt.figure()
plt.plot(df2["Log Time"], df2["Log Current"])
plt.plot(myline_1, p(myline_1))
#plt.title("log(Current) against log(Time)")
plt.xlabel("log(Time / s)")
plt.ylabel("log(Current / A)")
plt.show()
```

```python

# The polynomial regression is printed in the form of a linear equation.
print("The polynomial for the double logarithmic plot:")
print(" ")
print(p)
```

Listing 4: Code used to create the Pandas DataFrames necessary for analysis and then to plot a graph of the raw data alongside the raw data minus the dark current. The photocharge of the data is calculated twice using the Gaussian quadrature method for a polynomial regression and the cumulative trapezoidal numerical integration method on the data itself. The double logarithmic plot of the current against the time is then plotted alongside a polynomial regression of that data.

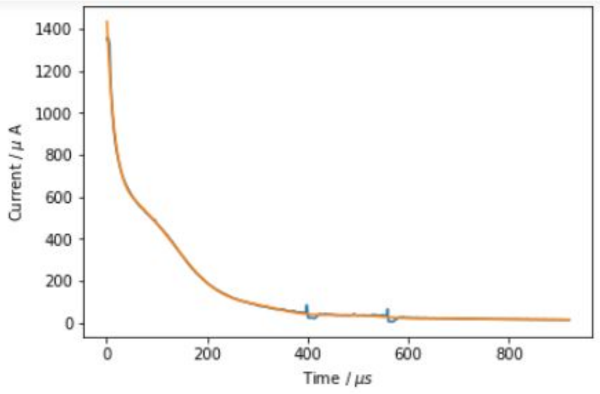
```
The datatypes of each column:
Time       float64
Voltage    float64
dtype: object
```

```
C:\Users\bluse\Anaconda3\lib\site-packages\ipykernel_launcher.py:85: RuntimeWarning: divide by zero encountered in log10
C:\Users\bluse\Anaconda3\lib\site-packages\ipykernel_launcher.py:86: RuntimeWarning: invalid value encountered in log10
```

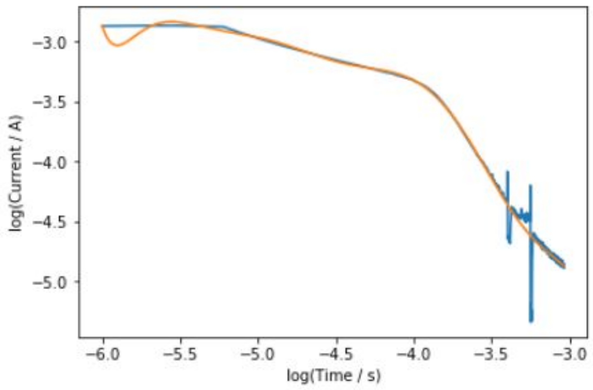


```
The generated polynomial regression:

            20             19             18             17
5.411e+63 x  - 4.755e+61 x  + 1.818e+59 x  - 3.797e+56 x
             16             15            14             13
 + 4.258e+53 x  - 1.315e+50 x  - 2.93e+47 x  + 3.858e+44 x
             12             11            10             9
 - 3.954e+40 x  - 3.679e+38 x  + 4.91e+35 x  - 3.561e+32 x
            8             7             6             5             4
 + 1.72e+29 x - 5.859e+25 x + 1.435e+22 x - 2.533e+18 x + 3.192e+14 x
             3             2
 - 2.796e+10 x + 1.602e+06 x - 55.65 x + 0.001487
```

Figure 31: A screen capture of part of the typical output of the above cell of code. The full output of the code would not fit on to one page so I had to separate the output into several images.



```
Polynomial photocharge: 1.3077269495391782e-07 C with an error of: 1.0740237078041807e-11 C

Cumulative trapezoidal photocharge: 1.3097245394736852e-07 C
```

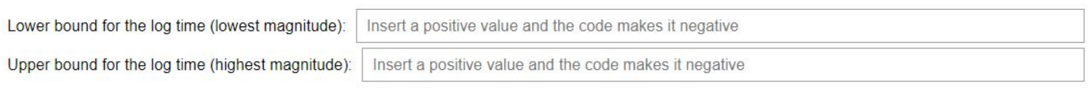


```
The polynomial for the double logarithmic plot:

        10         9          8        7        6              5
0.1287 x  + 5.568 x + 107.5 x + 1219 x + 8987 x + 4.501e+04 x
             4              3            2
 + 1.55e+05 x + 3.623e+05 x + 5.5e+05 x + 4.895e+05 x + 1.939e+05
```

Figure 32: A screen capture of part of the typical output of the above cell of code. The full output of the code would not fit on to one page so I had to separate the output into several images.

```
# A 2nd user interface is created for selecting the cuts.
widget_width = '900px'
padding = '0px 0px 0px 4px'

text4 = widgets.Text(
    value="",
    description="Upper bound for the log time (highest magnitude):",
    placeholder="Insert a positive value and the code makes it negative",
    style={"description_width": "initial"},
    layout=widgets.Layout(width=widget_width, padding=padding),
)
text4

text5 = widgets.Text(
    value="",
    description="Lower bound for the log time (lowest magnitude):",
    placeholder="Insert a positive value and the code makes it negative",
    style={"description_width": "initial"},
    layout=widgets.Layout(width=widget_width, padding=padding),
)
text5

# The inputs are saved as variables to be used later in the code.
toolbar_widget = widgets.VBox()
toolbar_widget.children = [
    text5,
    text4


]
# Displays the widget.
toolbar_widget
```

Listing 5: Code used to generate the second user interface for the cuts



Lower bound for the log time (lowest magnitude): Insert a positive value and the code makes it negative

Upper bound for the log time (highest magnitude): Insert a positive value and the code makes it negative

Figure 33: A screen capture of part of the output of the above cell of code which generates the second user input prompt which is shown in its empty form.

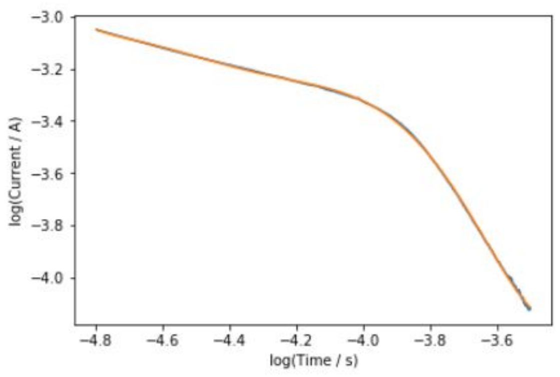
```
# Sets the outputs of the widget to be of float64 type.
cut_max = float(text4.value)
cut_min = float(text5.value)

print("
    ----------------------------------------------------------------------------------------------------
    ")
# A new dataframe is created where values outside of the cuts are disgarded.
df3 = df2[df2["Log Time"] < -cut_min]
df3 = df3[df3["Log Time"] > -cut_max]

# A polynomial regression of the line between the cuts is generated. You can edit the
    number of free
# parameters after the 2nd comma.
f1 = np.polyfit(df3["Log Time"], df3["Log Current"], 6)
p1 = np.poly1d(f1)

```

```python
# A linspace is used to allow us to plot our polynomial regression. You can edit the
    number of points in the
# linspace after the 2nd comma.
myline_2 = np.linspace(df3["Log Time"].min(), df3["Log Time"].max(), 100)

# Plots a double logarithmic graph of the data and of the polynomial between the two
    cuts.
plt.figure()
plt.plot(df3["Log Time"], df3["Log Current"])
plt.plot(myline_2, p1(myline_2))
#plt.title("log(Current) against log(Time)")
plt.xlabel("log(Time / s)")
plt.ylabel("log(Current / A)")
plt.show()

# Prints the linear equation of the polynomial regression between the two cuts.
print("The zoomed-in polynomial for the double logarithmic plot:")
print(" ")
print(p1)

# The kneedle algorithm searches for a knee point on a concave decreasing curve. This
    can be editted
# to be used for a convex and an increasing curve but that is not needed here. The knee
     point is saved
# into a variable known as kneedle_2
kneedle_2 = KneeLocator(myline_2, p1(myline_2), S = 1.0, curve = "concave", direction =
     "decreasing")
# The different components of kneedle_2 are given variable names
knee_x = kneedle_2.knee
knee_y = kneedle_2.knee_y
# The values of the logarithmic time and current for the kneedle algorithm alone are
    printed.
print(" ")
print("Log(Time) value of knee point:", knee_x)
print(" ")

print("Log(Current) value of knee point:", knee_y)
print(" ")
print(kneedle_2.plot_knee())
print(" ")

# Values used as the nearest points to the knee on the graph are chosen arbitrarily by
    adding and
# subtracting +/- 0.5 * the time of flight found using the kneedle method.
t_min = knee_x + math.log10(0.5 * -knee_x)
t_max = knee_x - math.log10(0.5 * -knee_x)

# 2 dataframes are created with only the data between the cuts and t_min or t_max
df4 = df3[df3["Log Time"] > t_min]
df5 = df3[df3["Log Time"] < t_max]

# The logarithmic time and current values are saved in lists which are then reshaped
    ready for use in
# the linear regression algorithm.
v = np.array(df4["Log Time"].values.tolist())
w = df4["Log Current"].tolist()
x = v.reshape((-1, 1))
y = w

```

```
# A linear regression is found with the slope and intercept being printed for both of
    the lines used
# in the bisection.
model1 = LinearRegression().fit(x, y)
intercept_1 = model1.intercept_
slope_alpha_2 = model1.coef_
print('Intercept alpha 2:', intercept_1)
print(" ")

v = np.array(df5["Log Time"].values.tolist())
w = df5["Log Current"].tolist()
x = v.reshape((-1, 1))
y = w

model2 = LinearRegression().fit(x, y)
intercept_2 = model2.intercept_
slope_alpha_1 = model2.coef_
print('Intercept alpha 1:', intercept_2)

# This finds the logarithmic time and current of the bisection of the two lines and
    prints
# them out.
x = Symbol('x')
m = solve(intercept_1 -intercept_2 + (slope_alpha_2 - slope_alpha_1) * x, x)
print(" ")
print("The Log(Time) value of the intersection:", m[x])
n = intercept_1 + slope_alpha_2 * m[x]
o = n[0]
print(" ")
print("The Log(Current) value of the intersection:", o)

# The zoomed-in double logarithmic plot is displayed with the bisection lines.
linspace = np.linspace(df2["Log Time"].min(), df2["Log Time"].max(), num=50)
plt.figure()
plt.plot(df3["Log Time"], df3["Log Current"], label = "Raw Data")
plt.plot(myline_2, p1(myline_2), label = "Polynomial")
plt.plot(linspace, slope_alpha_2*linspace + intercept_1, label = "Fit 2")
plt.plot(linspace, slope_alpha_1*linspace + intercept_2, label = "Fit 1")
plt.vlines(knee_x, df3["Log Current"].min(), df3["Log Current"].max(), label = "Knee
    Point")
#plt.title("log(Current) against log(Time)")
plt.xlabel("log(Time / s)")
plt.ylabel("log(Current / A)")
plt.xlim([df3["Log Time"].min(), df3["Log Time"].max()])
plt.ylim([df3["Log Current"].min(), df3["Log Current"].max()])
plt.legend()
plt.show()

# # The double logarithmic plot is displayed with the cuts, kneedle point, and the
    bisection lines.
plt.figure()
plt.scatter(df2["Log Time"], df2["Log Current"], label = "Raw Data", s = 5)
plt.plot(linspace, slope_alpha_2*linspace + intercept_1, label = "Fit 2", color = "red"
    )
plt.plot(linspace, slope_alpha_1*linspace + intercept_2, label = "Fit 1", color = "
    green")
plt.vlines(knee_x, df2["Log Current"].min(), df2["Log Current"].max(), label = "Knee
    Point")
plt.vlines(-cut_max, df2["Log Current"].min(), df2["Log Current"].max(), label = "Cut 1
    ", linestyles = "dashed")
plt.vlines(-cut_min, df2["Log Current"].min(), df2["Log Current"].max(), label = "Cut 2
    ", linestyles = "dashed")
#plt.title("log(Current) against log(Time)")
plt.xlabel("log(Time / s)")
plt.ylabel("log(Current / A)")
plt.xlim([df2["Log Time"].min(), df2["Log Time"].max()])
plt.ylim([df2["Log Current"].min(), df2["Log Current"].max()])
plt.legend()
plt.savefig(str(graph_savename) + "/Scatter_doublelog" + str(Pad_Voltage) + "V.png",
    dpi = 300, bbox_inches = 'tight')
plt.show()

print("Slope of line corresponding to Alpha 1:", slope_alpha_1[0])
print("Slope of line corresponding to Alpha 2:", slope_alpha_2[0])
```

Listing 6: Code uses the cuts provided by the user to create a Pandas DataFrame using information only between those two cuts. It then creates a polynomial regresssion for the double logarithmic plot and uses the kneedle program on this regression to find a knee. This then uses equations 23 and 24 where c is 0.5 to find where to take two linear regressions of the polynomial which then bisect to provide us with values for the time-of-flight. Many plots are generated for analysis.

```
------------------------------------------------------------------------------------------
```



```
The zoomed-in polynomial for the double logarithmic plot:

       6         5        4        3              2
4.226 x + 107.9 x + 1145 x + 6457 x + 2.042e+04 x + 3.431e+04 x + 2.393e+04

Log(Time) value of knee point: -3.9452551537323464

Log(Current) value of knee point: -3.3624155765028263

None

Intercept alpha 2: -11.113720085635098

Intercept alpha 1: -4.630520786958941

The Log(Time) value of the intersection: -3.88670666657823

The Log(Current) value of the intersection: -3.35368854043147
```

Figure 34: A screen capture of part of the typical output of the above cell of code. The full output of the code would not fit on to one page so I had to separate the output into several images.

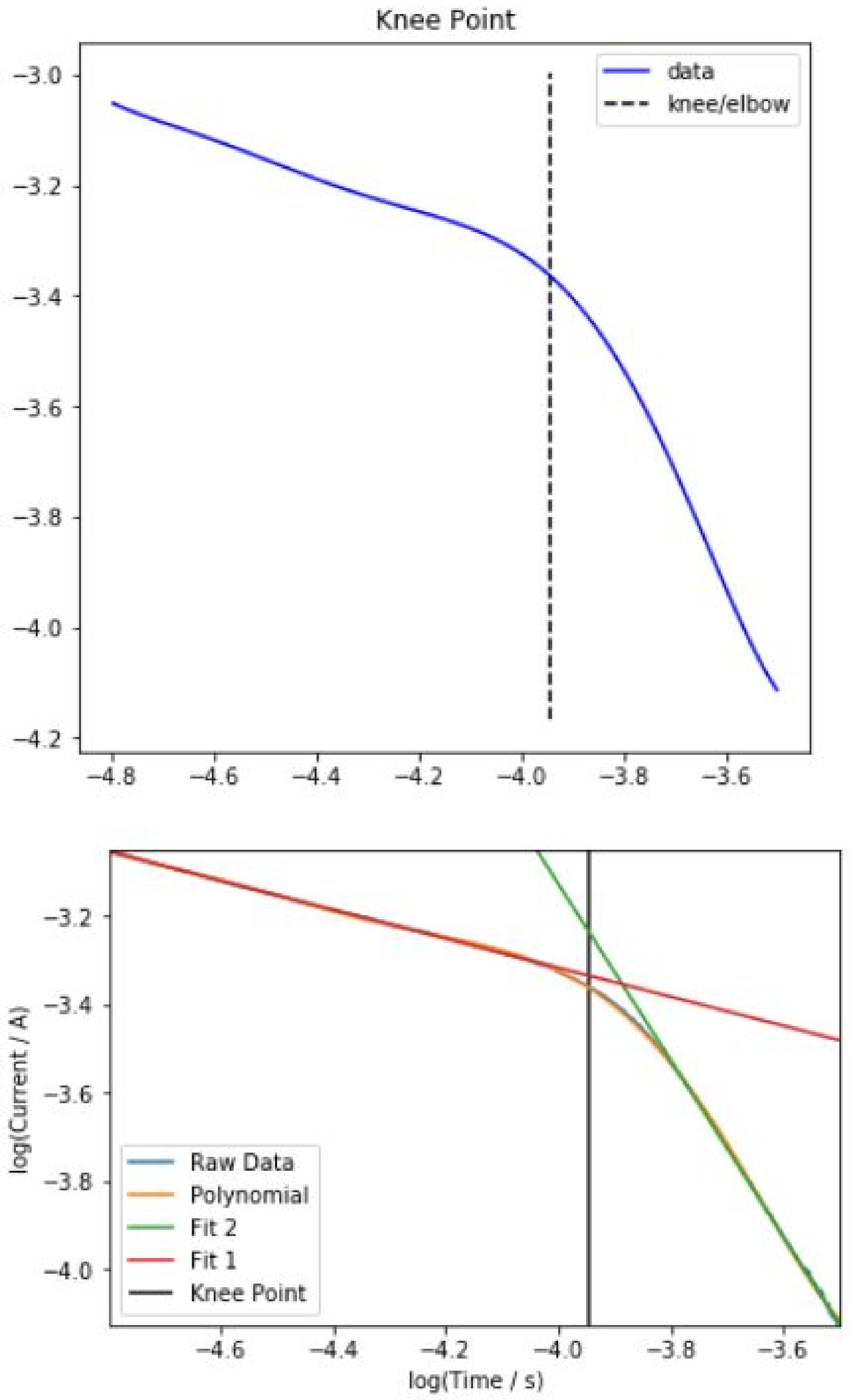
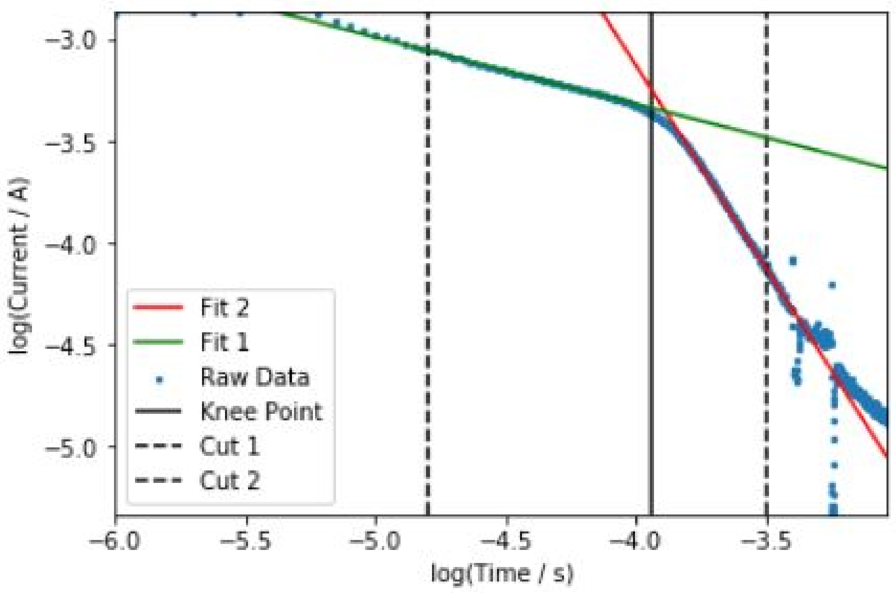


Figure 35: A screen capture of part of the typical output of the above cell of code. The full output of the code would not fit on to one page so I had to separate the output into several images.



```
Slope of line corresponding to Alpha 1: -0.32851263448999946
Slope of line corresponding to Alpha 2: -1.9965570368178518
```

Figure 36: A screen capture of part of the typical output of the above cell of code. The full output of the code would not fit on to one page so I had to separate the output into several images.

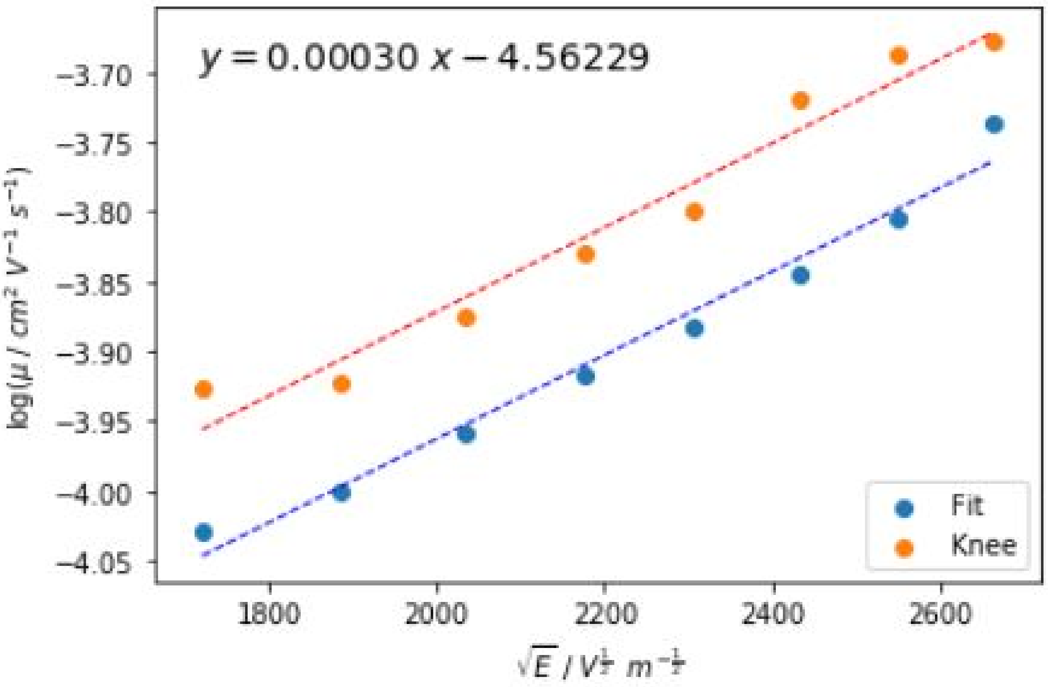
```python
# A new dataframe is created including many different data columns as shown below.
data = {'Pad Voltage/V': [Pad_Voltage], 'Resistance/Ohms': [Resistance], 'Thickness/m': [Thickness], 'Time [Fit]/microsecs': [10**6 * 10**(m[x])], 'Time [Knee]/microsecs': [10**6 * 10**(knee_x)], 'Dark current/ micro A': [10**6 * dark_current], 'Photocharge [Raw]/C': [y_int[-1]], 'Photocharge [Fit]/C': [photocharge], 'Photocharge error [Fit]/C': [photocharge_error], 'Alpha1': [1 + slope_alpha_1[0]], 'Alpha2': [-1 - slope_alpha_2[0]], 'Electric Field / (V/m)': [Pad_Voltage / Thickness], 'Mobility [Fit] / cm^2 V^-1 s^-1': [(10000 * (Thickness)**2) / (Pad_Voltage * 10**((m[x])))], 'Mobility [Knee] / cm^2 V^-1 s^-1': [(10000 * (Thickness)**2) / (Pad_Voltage * 10**((knee_x)))], 'sqrt(Electric Field) / (V/m)^0.5': [math.sqrt(Pad_Voltage / Thickness)]}
main_dataframe = pd.DataFrame(data)

# The code checks if a prexisting file exists and appends the data to that dataframe if it does. If it does not
# exist a new csv file is created with generated headers.
if os.path.isfile(savename) == True:
    main_dataframe.to_csv(savename, mode = 'a', header = False, index = False)
else:
    main_dataframe.to_csv(savename, header = True, index = False)
```

Listing 7: Code checks whether a specified CSV file already exists and either saves the data to a new file or appends it to an existing file specified by the user inputted file path.

```python
# A new dataframe for use in generating plots is created by reading the dataframe created above
# and a Poole-Frenkel plot is created and saved.
# A new dataframe for use in generating plots is created by reading the dataframe created above
# and a Poole-Frenkel plot is created and saved.
new_df = pd.read_csv(savename)

# The logarithm of the mobility is found and saved into the dataframe.
new_df["LogMobilityFit"] = np.log10(new_df["Mobility [Fit] / cm^2 V^-1 s^-1"])
new_df["LogMobilityKnee"] = np.log10(new_df["Mobility [Knee] / cm^2 V^-1 s^-1"])

# A Poole-Frenkel plot is created and saved.
plt.figure()
plt.scatter(new_df["sqrt(Electric Field) / (V/m)^0.5"], np.log10(new_df["Mobility [Fit] / cm^2 V^-1 s^-1"]), label = "Fit")
plt.scatter(new_df["sqrt(Electric Field) / (V/m)^0.5"], np.log10(new_df["Mobility [Knee] / cm^2 V^-1 s^-1"]), label = "Knee")
plt.xlabel(r"$\sqrt{E}$ / $V^{\frac{1}{2}}$  $m^{-\frac{1}{2}}$")
plt.ylabel(r"log($\mu$ / $cm^2$ $V^{-1}$ $s^{-1}$)")

# A linear regression is used to generate a trendline.
z = np.polyfit(new_df["sqrt(Electric Field) / (V/m)^0.5"], new_df["LogMobilityFit"], 1)

y_hat = np.poly1d(z)(new_df["sqrt(Electric Field) / (V/m)^0.5"])

# The equation of the linear regressions for the bisection fit only
# is printed in the corner of the graph.
text = f"$y={z[0]:0.5f}\;x{z[1]:+0.5f}$"
plt.gca().text(0.05, 0.95, text,transform=plt.gca().transAxes,
     fontsize=14, verticalalignment='top')
plt.plot(new_df["sqrt(Electric Field) / (V/m)^0.5"], y_hat, "b--", lw=1)
z = np.polyfit(new_df["sqrt(Electric Field) / (V/m)^0.5"], new_df["LogMobilityKnee"], 1)
y_hat = np.poly1d(z)(new_df["sqrt(Electric Field) / (V/m)^0.5"])
```

```python
plt.plot(new_df["sqrt(Electric Field) / (V/m)^0.5"], y_hat, "r--", lw=1)
plt.legend(loc = "lower right")
plt.savefig(str(graph_savename) + "/mobility_field.png", dpi = 300, bbox_inches = '
    tight')
plt.show()

# The photocharge is converted to nC from C and then a graph of photocharge against pad
    voltage
# is created.
new_df["Photocharge [Fit]/nC"] = new_df["Photocharge [Fit]/C"] * 1000000000
new_df["Photocharge [Raw]/nC"] = new_df["Photocharge [Raw]/C"] * 1000000000
plt.figure()
plt.scatter(new_df["Pad Voltage/V"], new_df["Photocharge [Fit]/nC"], label = "Fit")
plt.scatter(new_df["Pad Voltage/V"], new_df["Photocharge [Raw]/nC"], label = "Raw")
plt.xlabel("Pad Voltage / V")
plt.ylabel("Photocharge / nC")
plt.ylim([0, new_df["Photocharge [Raw]/nC"].max()])
plt.xlim(0,)

# A trendline is plotted and its equation is put in the corner of the graph.
z = np.polyfit(new_df["Pad Voltage/V"], new_df["Photocharge [Raw]/nC"], 1)
y_hat = np.poly1d(z)(new_df["Pad Voltage/V"])
text = f"$y={z[0]:0.5f}\;x{z[1]:+0.5f}$"
plt.gca().text(0.05, 0.95, text,transform=plt.gca().transAxes,
     fontsize=14, verticalalignment='top')
plt.plot(new_df["Pad Voltage/V"], y_hat, "r--", lw=1)
plt.legend(loc = "lower right")
plt.savefig(str(graph_savename) + "/photocharge_voltage.png", dpi = 300, bbox_inches =
    'tight')
plt.show()

# A graph of knee time v bisection time is plotted with a trendline and its equation.
plt.figure()
plt.scatter(new_df["Time [Fit]/microsecs"], new_df["Time [Knee]/microsecs"])
plt.xlabel(r"TOF for Fit Method / $\mu s$")
plt.ylabel(r"TOF for Kneedle Method / $\mu s$")
z = np.polyfit(new_df["Time [Fit]/microsecs"], new_df["Time [Knee]/microsecs"], 1)
y_hat = np.poly1d(z)(new_df["Time [Fit]/microsecs"])
text = f"$y={z[0]:0.5f}\;x{z[1]:+0.5f}$"
plt.gca().text(0.05, 0.95, text,transform=plt.gca().transAxes,
     fontsize=14, verticalalignment='top')
plt.plot(new_df["Time [Fit]/microsecs"], y_hat, "r--", lw=1)
plt.ylim(0,)
plt.xlim(0,)
plt.savefig(str(graph_savename) + "/kneedletime_fittime.png", dpi = 300, bbox_inches =
    'tight')
plt.show()

# Creates a plot of alpha1 against pad voltage.
plt.figure()
plt.scatter(new_df["Pad Voltage/V"], new_df["Alpha1"])
plt.xlabel("Pad Voltage / V")
plt.ylabel(r"$\alpha_{1}$")
z = np.polyfit(new_df["Pad Voltage/V"], new_df["Alpha1"], 1)
y_hat = np.poly1d(z)(new_df["Pad Voltage/V"])
text = f"$y={z[0]:0.5f}\;x{z[1]:+0.5f}$"
plt.gca().text(0.05, 0.95, text,transform=plt.gca().transAxes,
     fontsize=14, verticalalignment='top')
plt.plot(new_df["Pad Voltage/V"], y_hat, "r--", lw=1)
plt.savefig(str(graph_savename) + "/alpha1_voltage.png", dpi = 300, bbox_inches = '
    tight')
plt.show()

# Creates a plot of alpha2 against pad voltage.
plt.figure()
plt.scatter(new_df["Pad Voltage/V"], new_df["Alpha2"])
plt.xlabel("Pad Voltage / V")
plt.ylabel(r"$\alpha_{2}$")
z = np.polyfit(new_df["Pad Voltage/V"], new_df["Alpha2"], 1)
y_hat = np.poly1d(z)(new_df["Pad Voltage/V"])
text = f"$y={z[0]:0.5f}\;x{z[1]:+0.5f}$"
plt.gca().text(0.05, 0.95, text,transform=plt.gca().transAxes,
     fontsize=14, verticalalignment='top')
plt.plot(new_df["Pad Voltage/V"], y_hat, "r--", lw=1)
plt.savefig(str(graph_savename) + "/alpha2_voltage.png", dpi = 300, bbox_inches = '
    tight')
plt.show()
```

Listing 8: Code reads the Pandas DataFrame created in the previous step and creates various different graphs for quick and easy analysis which are saved in the user selected file path.



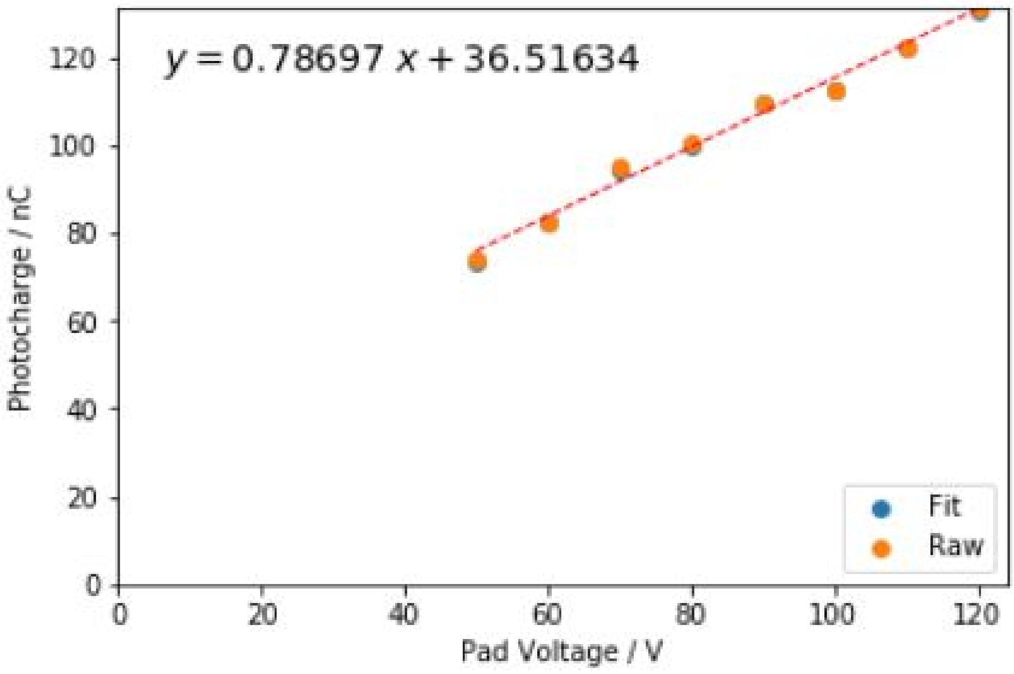
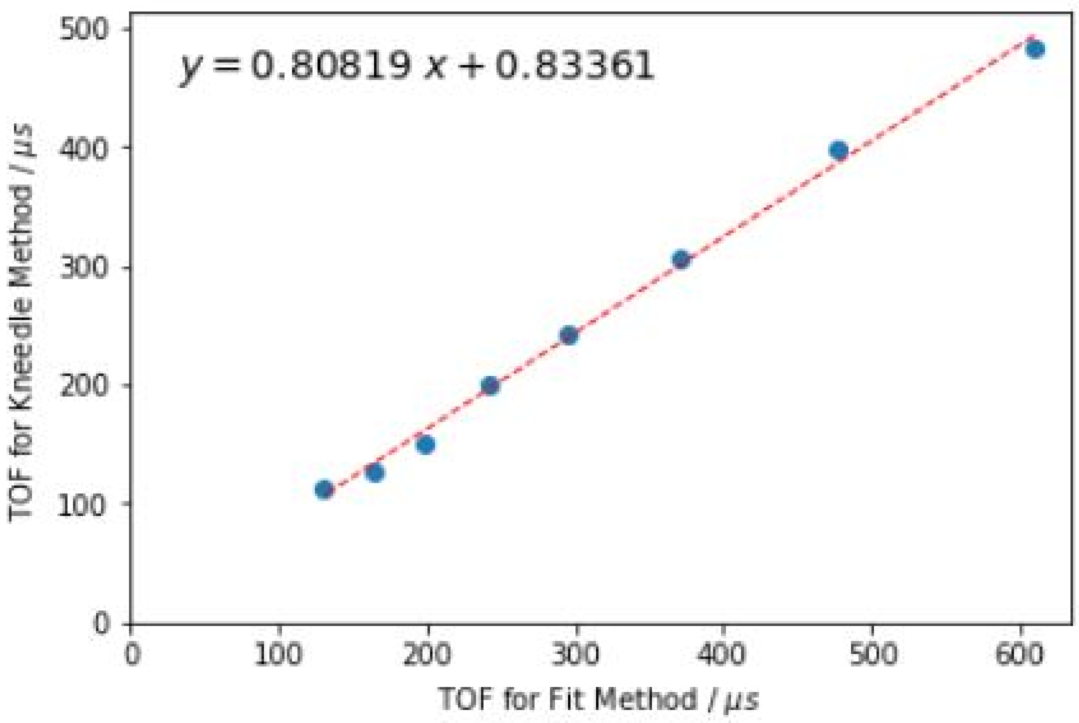


Figure 37: A screen capture of part of the typical output of the above cell of code. The full output of the code would not fit on to one page so I had to separate the output into several images.



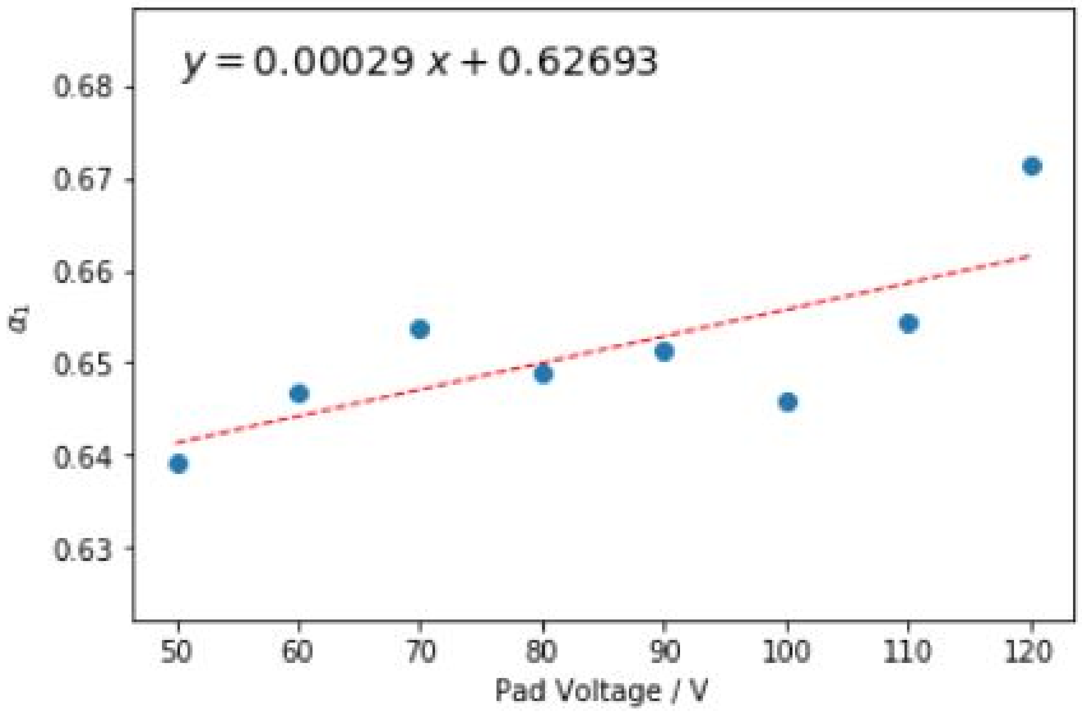


Figure 38: A screen capture of part of the typical output of the above cell of code. The full output of the code would not fit on to one page so I had to separate the output into several images.

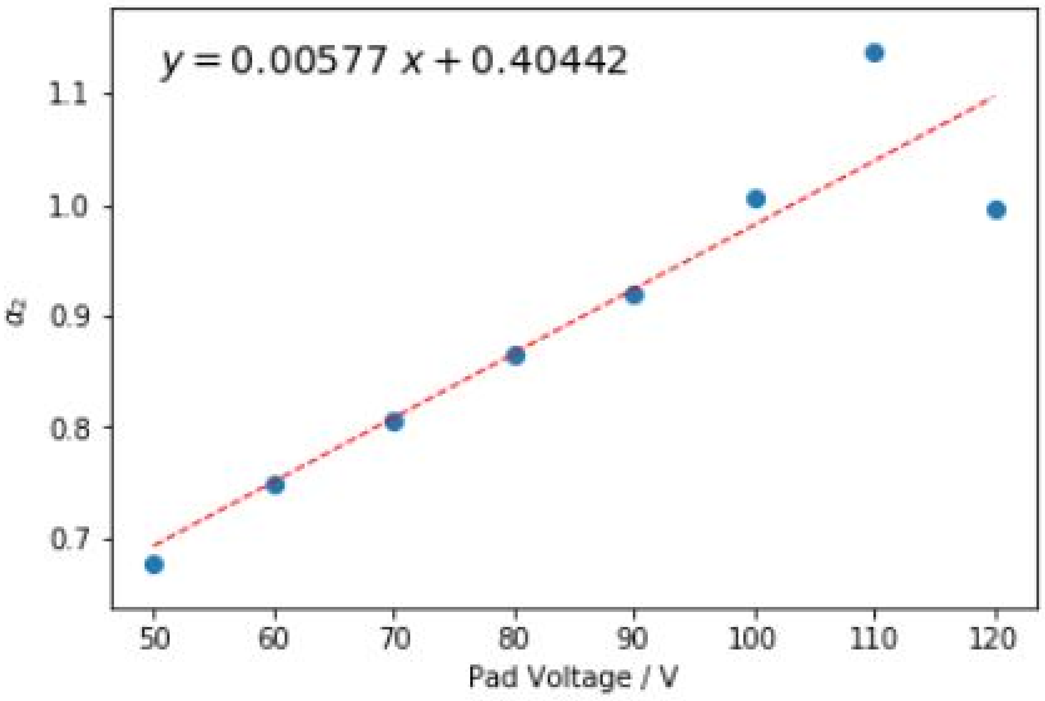


Figure 39: A screen capture of part of the typical output of the above cell of code. The full output of the code would not fit on to one page so I had to separate the output into several images.