REPORT SUBMISSION FORM

Name	:	TAN WEi LIANG
Partner's Name	:	AINA IMANINA BINTI MOHB KHOZIKIN
Group	:	M5B
Experiment Code	:	1MP1
Experiment Title	:	RADIOACTIVITY
Lecturer's/Examiner's Name	:	Dr. Dian Alwani Zainuri
Starting Date (1st session)	:	10/06/2024
Ending Date (2nd session)	:	24/06/2024
Submission Date	:	01/07/2024

DECLARATION OF ORIGINALITY

I, **TAN WEI LIANG 22302889** hereby declare that this laboratory report is my own work. I further declare that:

1. The references/bibliography reflect the sources I have consulted, and
2. I also certify that this report has not previously been submitted for assessment in this or any other units, and that I have not copied in part or whole or otherwise plagiarized the work of other students and/or persons.
3. Sections with no source referrals are my own ideas, arguments, and/or conclusions.

Signature: ______________________ Date: 01/07/2024

DYNAMICS

By

TAN WEI LIANG

June 2024

First Year Laboratory Report

RADIOACTIVITY

ABSTRACT

The research paper titled "Radioactivity". This physics report investigates three key aspects of radioactive decay using a Geiger-Müller (G-M) tube: determining the operating voltage, estimating the standard deviation of count rates, and measuring the range of beta particles. The objectives were to find the optimal operating voltage for the G-M tube by analyzing the count rate versus applied voltage, to verify that the standard deviation for a single count rate can be approximated as $\sqrt{R/t}$ within a 68% confidence interval, and to determine the range of beta particles using an aluminum absorber. The study found the operating voltage for the G-M tube to be 1010 V, with a plateau slope of 0.07% per V. The standard deviation of count rates was consistent with theoretical predictions, showing a 0.0081% percentage discrepancy. The range of beta particles in aluminum was determined to be 678.61 mg cm^{-2}, with the absorption coefficient calculated as 0.0048 cm^{-1}. The half-thickness value ($X_{1/2}$) for beta particles in the aluminum absorber is 143.37 mg cm^{-2}. Overall, this study confirms the principles of radioactivity and enhances our understanding of radiation detection and measurement.

ACKNOWLEDGEMENTS

First and foremost, I express my deepest gratitude to Dr. Dian Alwani Zainuri, our distinguished lecturer and examiner, for his invaluable guidance and unwavering support throughout our scientific exploration. I am truly thankful for his mentorship and the foundation he laid for our scientific understanding. I extend my sincere gratitude to my experiment partner, Aina Imanina Binti Mohb Khozikin. Her invaluable cooperation and dedication throughout both experiments were instrumental to the success of this project. I appreciate her commitment, expertise, and teamwork, which made these scientific endeavours both productive and enjoyable. A heartfelt acknowledgment is also extended to Dr. John Soo Yue Han for his dedicated efforts in revising and standardizing the manual in 2021, elevating its clarity and educational significance. This collective endeavor has significantly enhanced our scientific learning journey, and I extend genuine gratitude to everyone mentioned for their noteworthy contributions.

CONTENTS

List of Tables

List of Figures

INTRODUCTION

This experiment investigates three main objectives: determining the operating voltage for a Geiger-Müller (G-M) tube, estimating the standard deviation of count rates, and measuring the range of beta particles. Radioactivity involves the emission of particles or electromagnetic radiation from unstable atomic nuclei, with alpha, beta, and gamma decay being the primary types. The G-M tube detects ionizing radiation by producing electrical pulses when gas inside the tube is ionized. Statistical analysis is essential in radioactive measurements due to the random nature of decay, with the count rate following a Poisson distribution. The range of beta particles is measured using an aluminum absorber to determine the thickness required to reduce the count rate to background levels. This experiment aims to bridge theoretical concepts and practical applications, enhancing understanding of radiation detection and measurement.

THEORY

Radioactivity

Our present state of knowledge indicates that an atom is composed of a central core (the nucleus) and various groupings of electrons in rapid motion around it. The nuclei consist of neutrons and protons, it can exist only in certain definite energy states. Transitions from higher to lower energy states are accompanied by the emission of either electromagnetic radiation or subatomic particles. This phenomenon called radioactivity, and when exhibited by naturally occurring isotopes (e.g. uranium, radium or polonium) it is termed natural radioactivity.

Artificial radioactivity is related to man-made isotopes. In this experiment, we shall consider the three most important modes of radioactive disintegration, characterised according to the emission as either alpha, beta or gamma decay.

Alpha decay

The alpha particle is identical with the helium nucleus (^{4}He). When ejecting an alpha particle, the original nucleus loses four unit masses (two protons, two neutrons) and two units of charge. Hence, the resulting daughter nucleus is that of a different element. This new isotope may also be radioactive, and may decay again via the emission of an alpha or beta particle. Alpha particles are highly ionising, and they lose energy over a short distance, thus they cannot travel far in most medium. Alpha particles are commonly emitted by the larger radioactive nuclei such as polonium-210, radon-222, radium-226 and americium-241.

Beta decay

Beta particles may be either negative (electrons, e^-) or positive (positrons, e^+), the former being by far the more common type. The daughter isotopes will have the same mass number as the parent (since the mass of the ejected beta particle is negligible in comparison with the mass of the nucleus). Emitted simultaneously with the beta particle is an electrically neutral particle of negligible rest mass called a neutrino. Beta particles have the moderate penetrating and ionising power. Although the beta particles given off by different radioactive materials vary in energy, most beta particles can be stopped by a few millimetres of aluminium. Examples of radioactive materials that give off beta particles are hydrogen-3 (tritium), carbon-14, phosphorus-32, sulfur-35 and strontium-90.

Gamma decay

Transitions from higher to lower nuclear energy states of the same isotope are accompanied by the emission of gamma rays. These rays are similar in nature with X-rays, radio waves and other electromagnetic radiation, but are of much higher energy. These waves can travel a considerable range in air and have greater penetrating power (can travel further) than either alpha or beta particles. Gamma rays are generally blocked by thick blocks of lead or other heavy materials. Examples of common radionuclides that emit gamma rays are technetium-99m, iodine-125, iodine-131, cobalt-57 and cesium-137.

Absorption of Radiation

The absorption of beta and gamma radiation may be described by an exponential equation,

$$R = R_0 e^{-\mu x}, \tag{1}$$

where R is the radiation intensity, R_0 the radiation intensity without an absorber, μ the linear absorption coefficient and x the absorber's thickness. μ is dependent on the material of which the absorber is made and has a dimension of $[L^{-1}]$.

Equation 1 is most frequently written in the form of

$$R = R_0 e^{\frac{\mu}{\rho}\rho x} = R_0 e^{-\mu_m \rho x}, \tag{2}$$

where ρ is the density of the absorber. μ_m is the mass absorption coefficient with dimension $[L^2 M^{-1}]$, and ρx is the mass area density. This expression has an advantage such that μ_m is practically independent of the nature of the absorber.

Let $\rho x = X$. In logarithmic form, **Equation 2** becomes

$$\ln R = \ln R_0 - \mu_m X. \tag{3}$$

Thus, by plotting $\ln R$ vs. X, we will obtain a straight line with a slope of $-\mu_m$ and a y-intercept of $\ln R_0$.

When passing through matter, charged particles ionise and thus lose energy in many steps, until their energy is (almost) zero. The distance to this point is called the range (r) of the particle. The range depends on the type of particle, its initial energy, and on the material through which it passes. The extrapolated range is the point where the absorption curve meets the background, as shown in **Figure 1**.

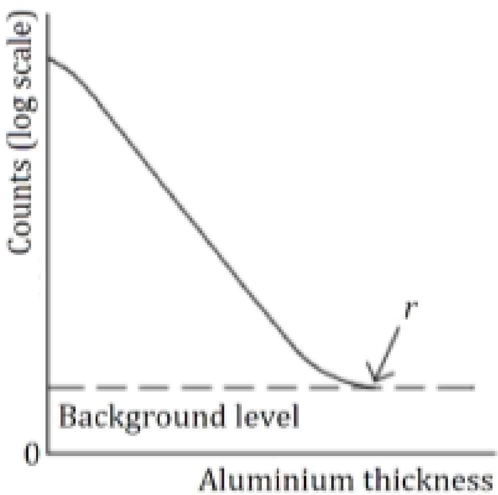


Figure 1: Beta decay absorption curve.

A useful measure of the penetrating power is the half-value thickness $X_{1/2}$ defined as the thickness of the absorber necessary to reduce the radiation intensity by a factor of two ($R/R_0 = 1/2$). Thus, from **Equation 3**,

$$\ln(\frac{R_o}{2}) = \ln R_0 - \mu_m X_{\frac{1}{2}}. \tag{4}$$

$$X_{1/2} = -\frac{\ln\left(\frac{1}{2}\right)}{\mu_m}. \tag{5}$$

In fact, only gamma radiation actually obeys the above relationship exactly, provided that all secondary radiation is excluded from a beam arriving at the detector. However, you will find in this experiment that the equations provide quite a good quantitative description of the total absorption of the beta radiation as well.

Uncertainty in the Count Rate

Radioactive decay and most other nuclear reactions are random events; therefore they must be described quantitatively in statistical terms. Not only is there a continuous change in the activity within a specific measurement (due to the half-life of the radionuclide), but there is also a fluctuation in the decay rate between measurements due to the random nature of radioactive decay. Thus the radiation count N from a single measurement can be expressed as

$$N \pm \sigma = N \pm \sqrt{N}, \tag{6}$$

where $\sigma = \sqrt{N}$ represents one standard deviation using Poisson statistics. Since a sample is counted for a specified period of time (t), the results are reported in units of inverse time, i.e. counts per minute (cpm) or counts per second (cps). Thus, the equation for count rate is

$$\frac{N}{t} \pm \frac{\sqrt{N}}{t} = R \pm \sqrt{\frac{R}{t}}, \tag{7}$$

where $R = N/t$ is the count rate, or counts per unit time.

The range of values $N \pm \sigma$ will contain the true mean N_{mean} within 68% probability. We can also say that the interval $N_{mean} \pm \sigma_{mean}$ has 68% probability of containing our single measurement N. Thus, we can interchange N_{mean} and N in the statement.

Geiger-Müller Tube

A Geiger-Müller (G-M) tube is a device used for the detection and measurement of all types of radiation: alpha, beta and gamma radiation. Basically, it consists of a pair of electrodes surrounded by a gas, usually helium or argon. The electrodes have high voltages across them. When radiation enters the tube, it ionises the gas, the ions and electrons are then attracted to the electrodes and an electric current is produced. A scaler counts the current pulses, and one obtains a count whenever radiation ionises the gas. **Figure 2** shows a simplified detector circuit with a G-M tube.

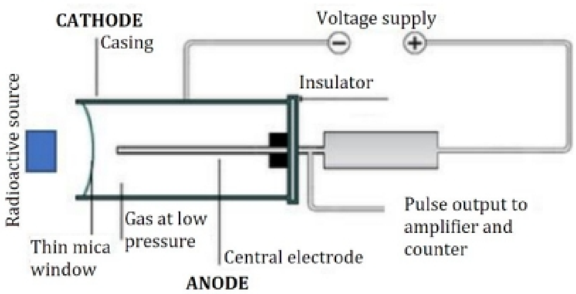


Figure 2: A simplified detector circuit with a G-M tube.

The characteristic curve of a G-M tube is obtained by plotting the count rate as a function of supply voltage in a constant radiation field. The main features of these characteristics are given in **Figure 3** below.

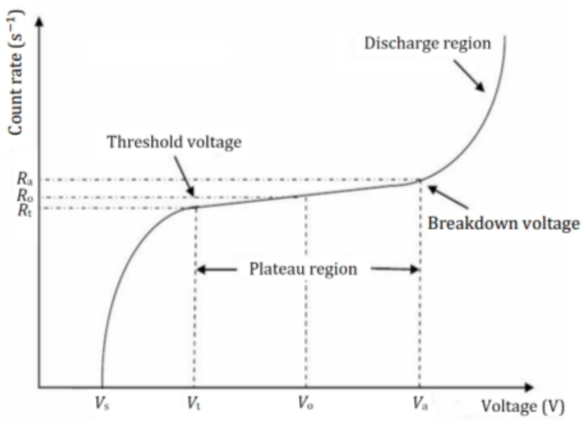


Figure 3: The characteristic curve of a Geiger-Müller tube.

At a very low voltage, the count rate is insignificant, so the tubes cannot generally be operated usefully in this region. The starting voltage (V_s) is defined as the lowest voltage applied to a counter tube at which pulses can be detected. Above the starting voltage, the count rate increases rapidly until it reaches the threshold voltage (V_t), which marks the beginning of the G-M tube plateau region (or Geiger region) for the conditions under which the circuit should be operating.

Beyond the threshold, further increase in voltage will result in a negligible increase in the count rate. An operating voltage (V_o) is selected to be used within this plateau. If the voltage is increased further past the plateau, another rapid rise in count rate takes place. This region is called the discharge region, where the voltage is large enough to cause the atoms to self-ionise. Operating a G-M tube in this region will quickly ruin the tube.

In this experiment, we will investigate the operating principles of the Geiger-Müller tube, validate the uncertainty analysis for a radioactive decay experiment and study some characteristics of β particles.

EXPERIMENTAL METHODOLOGY

In the **Part A** experiment to find operating Voltage for a Geiger-Müller Tube. The experiment commenced with setting up the Geiger-Müller (G-M) tube connected to the counter. The radioactive beta source (Sr-90) was placed at a suitable distance from the G-M tube window using tweezers to avoid contamination. The counter was switched on and allowed to warm up for a few minutes.

Starting with a low applied voltage, the voltage was increased in increments of approximately 20 V until the first detection of radiation counts was observed. This voltage was recorded as the starting voltage (V_s). The voltage was then increased further in 20 V increments, and the count rate was recorded at each increment, ensuring the count rate stabilized around 10^3 by adjusting the distance between the source and the G-M tube. The threshold voltage (V_t) its corresponding count rate (R_t) was identified where the count rate began to plateau. Recording continued until a rapid increase in count rate was observed, marking the breakdown voltage (V_a) and its corresponding count rate (R_a).

The data, including count rates and corresponding voltages, were accurately documented. A graph of count rate against applied voltage was plotted to visualize the characteristic curve of the G-M tube. The Geiger plateau region between R_t and R_a corresponding to the voltages V_t and V_a was identified, and the slope of the plateau was computed using the formula:

$$\text{Slope} = \frac{R_a - R_t}{0.5(R_a + R_t) \times (V_a - V_t)} \times 100\%$$

In the **Part B** experiment, the applied voltage was set to the operating voltage determined in Part A. Twenty separate measurements of the count rate were taken, and each count rate along with the total time taken for these measurements was recorded. The standard deviation (σ) for the 20 measurements was calculated and compared with $\sqrt{R/t}$. The percentage discrepancy was computed, and conditions under which Poisson distribution approaches a Gaussian distribution were discussed.

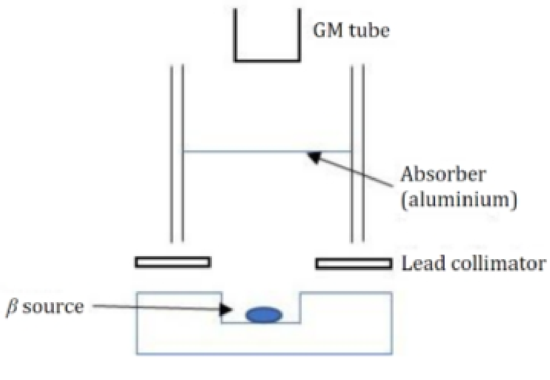


Figure 4: Experimental setup for measuring the range of β radiation.

In **Part C** experiment, the applied voltage was set to the operating voltage determined in Part A, and the background count rate without any absorber was measured and recorded. The experimental setup was arranged as illustrated in the **Figure 4**, ensuring the Sr-90 source was correctly positioned. An initial aluminum foil layer was placed between the source and the G-M tube, and the count rate was measured for 30 seconds.

This process was repeated with additional layers of aluminum foil added in pairs until the recorded activity dropped to the background radiation level.

Graphs of the count rate (R) against the thickness of the absorber (X) were plotted, including the background count rate. Additionally, a graph of the logarithm of the count rate ($\ln R$) against the thickness of the absorber (X) was plotted. The range of beta particles in aluminum was determined from these graphs, and the absorption coefficient (μ_m) was calculated. The values obtained from different graphs were compared to verify if the count rate satisfied the exponential absorption equation $R = R_0 e^{-\mu_m X}$. Finally, the half-thickness value ($X_{1/2}$) for beta particles in the aluminum absorber was computed. The value of $X_{1/2}$ can be obtained by setting $\frac{R}{R_o} = 1$.

DATA ANALYSIS

PART A

Voltage(V)	n_1	n_2	n_3	$n_{average}$	Count Rate (s^{-1})
780	0	0	0	0	0
800	198	195	202	198	6.611111
820	872	865	892	876	29.211111
840	984	970	983	979	32.633333
860	1000	1014	1002	1005	33.511111
880	1035	1031	1037	1034	34.477778
900	1041	1041	1047	1043	34.766667
920	1076	1061	1064	1067	35.566667
940	1079	1061	1102	1081	36.022222
960	1056	1040	1125	1074	35.788889
980	1069	1114	1101	1095	36.488889
1000	1074	1138	1110	1107	36.911111
1020	1137	1087	1138	1121	37.355556
1040	1059	1122	1156	1112	37.077778
1060	1164	1168	1165	1166	38.855556
1080	1131	1149	1148	1143	38.088889
1100	1122	1213	1256	1197	39.900000
1120	1207	1215	1203	1208	40.277778
1140	1244	1243	1240	1242	41.411111
1160	1209	1256	1234	1233	41.100000
1180	1285	1260	1223	1256	41.866667
1200	1322	1334	1347	1334	44.477778
1220	2030	2031	2036	2032	67.744444

Table 1: Data table for PART A

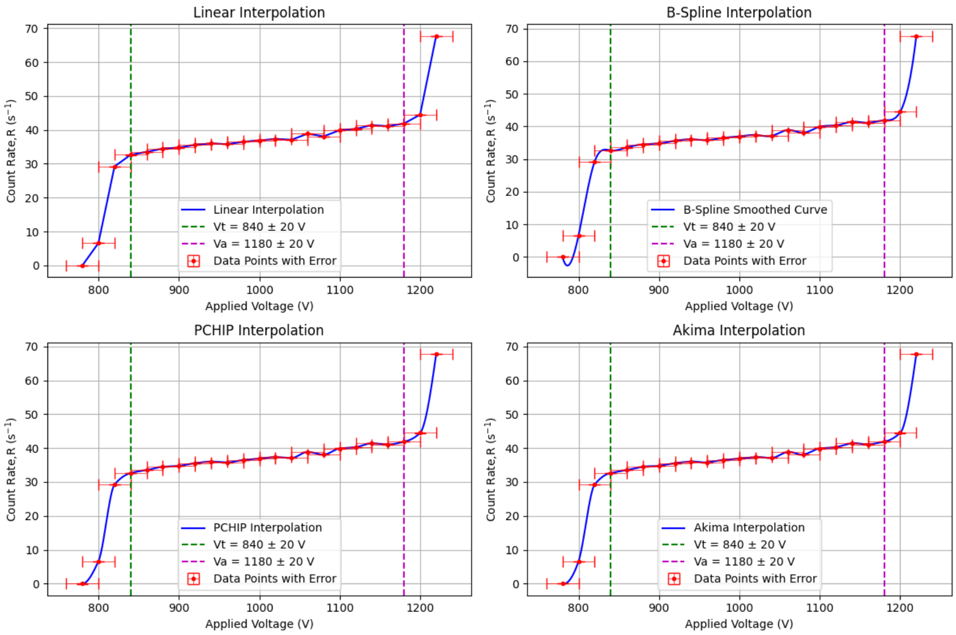


Figure 5: Graphs of count rate against applied voltage with various method.

By comparing different graphing method, Akima Interpolation show the best data visualisation,

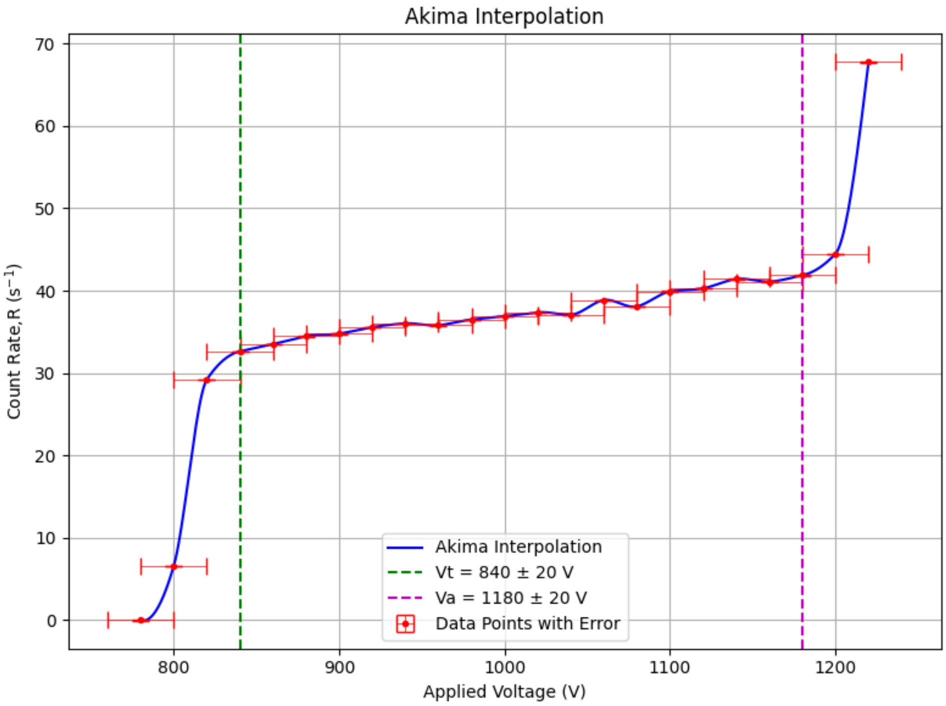


Figure 6: Graph of count rate against applied voltage with Akima Interpolation.

Calculation by using Python,

Starting Voltage = 780 V

Threshold Voltage , $V_t = 840 \pm 20$ V; Count Rate , $R_t = 32.63 \pm 0.03\,\mathrm{s}^{-1}$

Breakdown Voltage , $V_a = 1180 \pm 20$ V; Count Rate , $R_a = 41.87 \pm 0.03\,\mathrm{s}^{-1}$

Operating Voltage, $V_0 = 1010\,\mathrm{V}$

Slope of the Geiger plateau, $m = 0.07\,\%\,per\,V$

Percentage difference between standard value, $m_0 = 0.10\,\%\,per\,V$ and experiment value, m = 27.10%

PART B

No.	N	Rate (s^{-1})	Standard deviation
1	1228	40.933333	1.168094
2	1186	39.533333	1.147945
3	1190	39.666667	1.149879
4	1210	40.333333	1.159502
5	1223	40.766667	1.165714
6	1164	38.800000	1.137248
7	1169	38.966667	1.139688
8	1211	40.366667	1.159981
9	1164	38.800000	1.137248
10	1182	39.400000	1.146008
11	1184	39.466667	1.146977
12	1146	38.200000	1.128421
13	1134	37.800000	1.122497
14	1168	38.933333	1.139200
15	1137	37.900000	1.123981
16	1189	39.633333	1.149396
17	1158	38.600000	1.134313
18	1151	38.366667	1.130880
19	1124	37.466667	1.117537
20	1132	37.733333	1.121507

Table 2: Data table for PART B

Calculation by using Python,

$$\text{Average Rate} = 39.08\, s^{-1}$$

$$\text{Average standard deviation}, \sqrt{\frac{R}{T}} = 1.1413008$$

$$\text{Overall standard deviation}, \sigma = 1.1413929$$

$$\text{Percentage Discrepancy between } \sqrt{\frac{R}{T}} \text{ and } \sigma = 0.0081\%$$

$$\text{Standard Deviation of Average standard deviation}, \sqrt{\frac{R}{T}} = 0.0148767$$

$$\text{Confidence Interval for standard deviation}, \sqrt{\frac{R}{T}} = [1.12652, 1.15627]$$

$$\text{Values within Confidence Interval} = 12$$

$$\text{Confidence Percentage of values within Confidence Interval} = 60.0\%$$

PART C

Type	Density (mg/cm^2)	1	2	3	4	Mean	Count Rate (s^{-1})
Al	4.5	1076	1090	1113	1075	1088.50	36.28
Al	6.5	1052	1073	1101	1008	1058.50	35.28
Poly	9.6	1085	1130	1117	1064	1099.00	36.63
Poly	19.2	1064	1038	1060	1058	1055.00	35.17
Plastic	59.1	833	852	826	825	834.00	27.80
Plastic	102.0	775	755	735	752	754.25	25.14
Al	141.0	577	589	585	578	582.25	19.41
Al	170.0	470	472	461	458	465.25	15.51
Al	216.0	326	332	337	328	330.75	11.02
Al	258.0	251	252	254	251	252.00	8.40
Al	328.0	134	140	142	142	139.50	4.65
Al	425.0	85	79	85	93	85.50	2.85
Al	522.0	46	49	50	47	48.00	1.60
Al	645.0	41	38	39	37	38.75	1.29
Al	655.0	35	35	34	36	35.00	1.17
Al	840.0	31	32	32	30	31.25	1.04
Lead	1120.0	29	28	29	29	28.75	0.96
Lead	2066.0	26	28	27	28	27.25	0.91
Lead	3448.0	24	26	23	26	24.75	0.82
Lead	7367.0	17	19	21	16	18.25	0.61

Table 3: Data table for PART C

Type	Density (mg/cm^2)	1	2	3	4	Mean	Count Rate (s^{-1})
Al	4.5	1076	1090	1113	1075	1088.50	36.28
Al	6.5	1052	1073	1101	1008	1058.50	35.28
Al	141.0	577	589	585	578	582.25	19.41
Al	170.0	470	472	461	458	465.25	15.51
Al	216.0	326	332	337	328	330.75	11.02
Al	258.0	251	252	254	251	252.00	8.40
Al	328.0	134	140	142	142	139.50	4.65
Al	425.0	85	79	85	93	85.50	2.85
Al	522.0	46	49	50	47	48.00	1.60
Al	645.0	41	38	39	37	38.75	1.29
Al	655.0	35	35	34	36	35.00	1.17
Al	840.0	31	32	32	30	31.25	1.04

Table 4: Data table for Aluminium

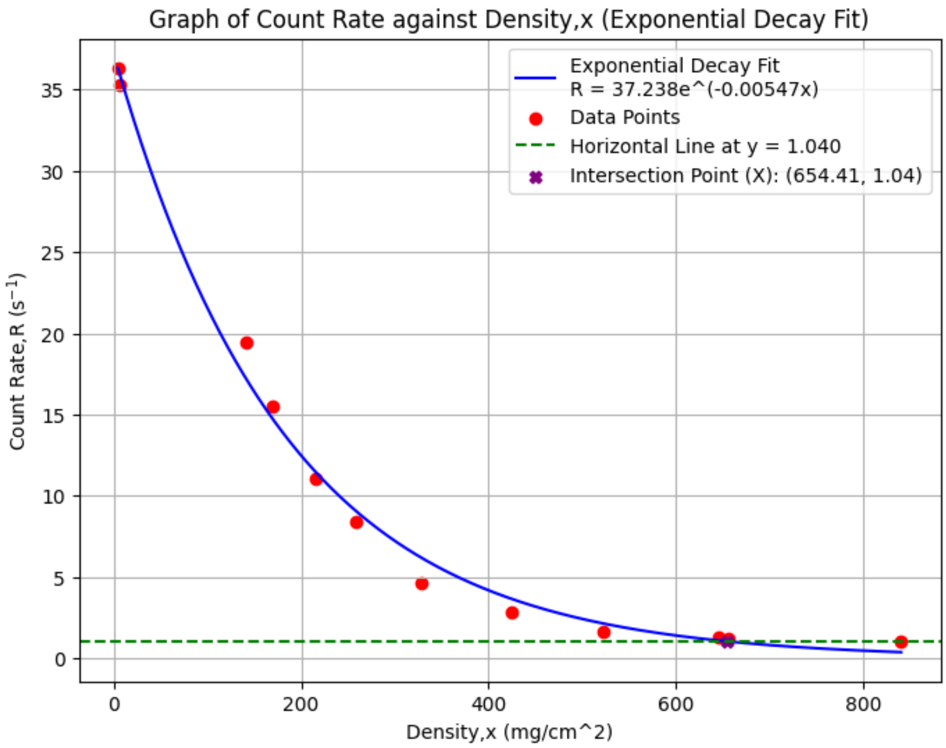


Figure 7: Graph of count rate against density.

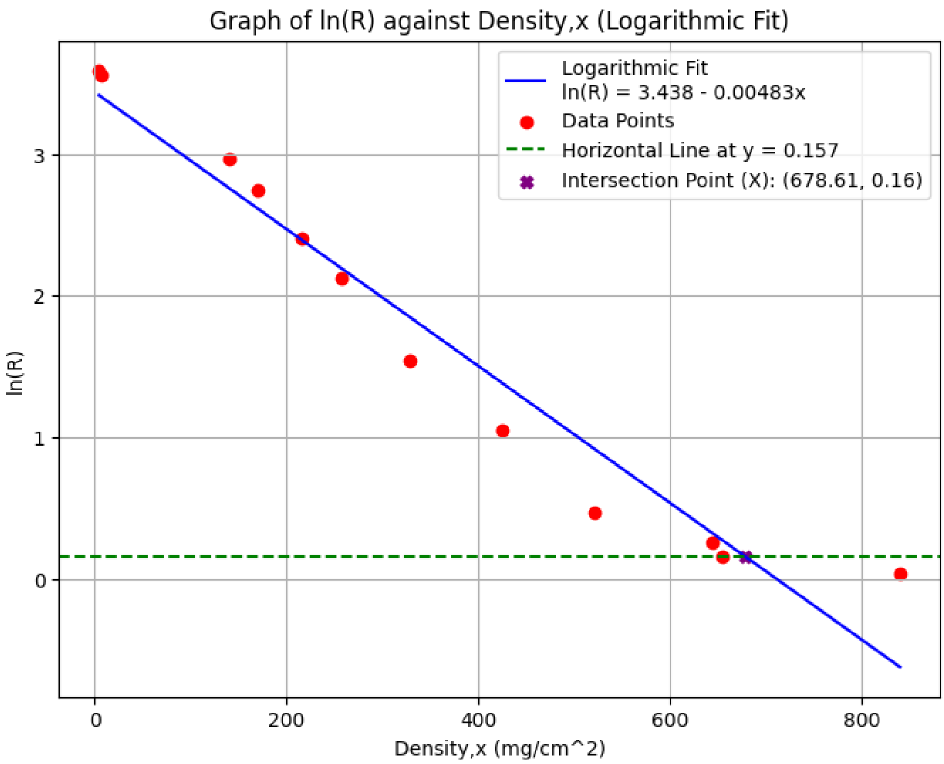


Figure 8: Graph of ln(R) against density.

Calculation by using Python,

The range of β particles in aluminium (β_1) is the x-value of intersection points of first graph
$= 654.41$ mg cm^{-2}
The range of β particles in aluminium (β_2) is the x-value of intersection points of second graph
$= 678.61$ mg cm^{-2}
Percentage difference between β_1 and β_2
$= 3.63\%$
mass absorption coefficient,$\mu_{m,\text{exp}}$
$= 0.00546768$ cm^{2}mg^{-1}
≈ 0.0055 cm^{2}mg^{-1}
mass absorption coefficient,$\mu_{m,\text{log}}$
$= 0.00483452$ cm^{2}mg^{-1}
≈ 0.0048 cm^{2}mg^{-1}
Percentage difference between μ from two different methods
$= 12.29\%$
The half-thickness value ($X_{1/2}$) for β particles in the aluminium absorber (β_2) is 143.37 mg cm^{-2}

DISCUSSION

In Part A of the experiment, the operating voltage V_0 determined was 1010 V. This value was essential for completing Parts B and C. The measured slope of the plateau for the Geiger-Müller (G-M) tube was 0.07% per volt, which is very close to the theoretical value of 0.10% per volt. The percentage discrepancy between theoretical and experiment is 27.10%, which indicates that our experimental results are accurate.

For Part B, the percentage discrepancy between the standard deviation σ and the calculated value $\sqrt{R/t}$ was found to be 0.0081%. This minimal discrepancy suggests that $\sqrt{R/t}$ is an excellent approximation for the standard deviation of count rates. Additionally, 12 out of 20 measurements fell within the range of the confidence interval. The standard deviation σ for a single count rate R can be estimated as $\sqrt{R/t}$ within a 68% confidence interval for nuclear decay, with a confidence percentage of 60%. This result aligns with the Poisson distribution approaching a Gaussian distribution as the sample size increases.

In Part C, the range of beta particles in aluminum (β_1) was determined to be 654.41 mg cm^{-2} from the graph of count rate R against absorber thickness X. Similarly, the range (β_2) from the graph of $\ln R$ against absorber thickness X was 678.61 mg cm^{-2}. The percentage difference between β_1 and β_2 was calculated to be 3.63%. The absorption coefficient μ_m was calculated as 0.0055 cm^2 mg^{-1} using the equation $R = R_0 e^{-\mu_m X}$ and as 0.0048 cm^2 mg^{-1} using the equation $\ln R = \ln R_0 - \mu_m X$. The percentage difference between the values of μ_m from these equations was 12.29%. The percentage differences indicate that the ranges of beta particles in aluminum obtained from the two different graphs are reliable. Furthermore, the count rate curve of beta particles follows the equation $R = R_0 e^{-\mu_m X}$, given the small percentage difference. The half-thickness value $X_{1/2}$ for beta particles in the aluminum absorber was found to be 143.37 mg cm^{-2}.

CONCLUSION

The value of operating voltage V_0 obtained is 1010 V. The slope of the plateau of the G-M tube is 0.07% per volt. The percentage discrepancy between σ and $\sqrt{R/t}$ is 0.0081%. The confidence percentage of $\sqrt{R/t}$ is 60%, proving that it can be estimated as $\sqrt{R/t}$ within a 68% confidence interval for nuclear decay. The range of beta particles in aluminum (β_1) from the graph of count rate R against the thickness of the absorber X is 654.41 mg cm^{-2}, and the range (β_2) from the graph of $\ln R$ against the thickness of the absorber X is 678.61 mg cm^{-2}. The absorption coefficient μ_m obtained is 0.0055 cm^2 mg^{-1} from the equation $R = R_0 e^{-\mu_m X}$ and 0.0048 cm^2 mg^{-1} from the equation $\ln R = \ln R_0 - \mu_m X$. The half-thickness value ($X_{1/2}$) for beta particles in the aluminum absorber is 143.37 mg cm^{-2}.

REFERENCES

1. Turner, J. E. (2008). *Atoms, Radiation and Radiation Protection* (3rd ed.). John Wiley & Sons.

2. Leo, W. R. (2012). *Techniques for Nuclear and Particle Physics Experiments* (2nd ed.). Springer Science & Business Media.

3. Knoll, G. F. (2010). *Radiation Detection and Measurement* (4th ed.). John Wiley & Sons.

APPENDICES

Python code of PART A

Data processing of PART A:

```
import pandas as pd

# Creating the DataFrame from the provided data
data = {
    "v": [780, 800, 820, 840, 860, 880, 900, 920, 940, 960, 980, 1000, 1020, 1040, 1060, 1080, 1100,
     1120, 1140, 1160, 1180, 1200, 1220],
    "n1": [0, 198, 872, 984, 1000, 1035, 1041, 1076, 1079, 1056, 1069, 1074, 1137, 1059, 1164, 1131,
     1122, 1207, 1244, 1209, 1285, 1322, 2030],
    "n2": [0, 195, 865, 970, 1014, 1031, 1041, 1061, 1061, 1040, 1114, 1138, 1087, 1122, 1168, 1149,
     1213, 1215, 1243, 1256, 1260, 1334, 2031],
    "n3": [0, 202, 892, 983, 1002, 1037, 1047, 1064, 1102, 1125, 1101, 1110, 1138, 1156, 1165, 1148,
     1256, 1203, 1240, 1234, 1223, 1347, 2036]
}

df = pd.DataFrame(data)

# Calculating n_sum and count_rate
df['n_sum'] = (df['n1'] + df['n2'] + df['n3']) / 3
df['n_sum'] = df['n_sum']
df['count_rate'] = df['n_sum'] / 30

# Displaying the DataFrame
print(df)
```

Output:

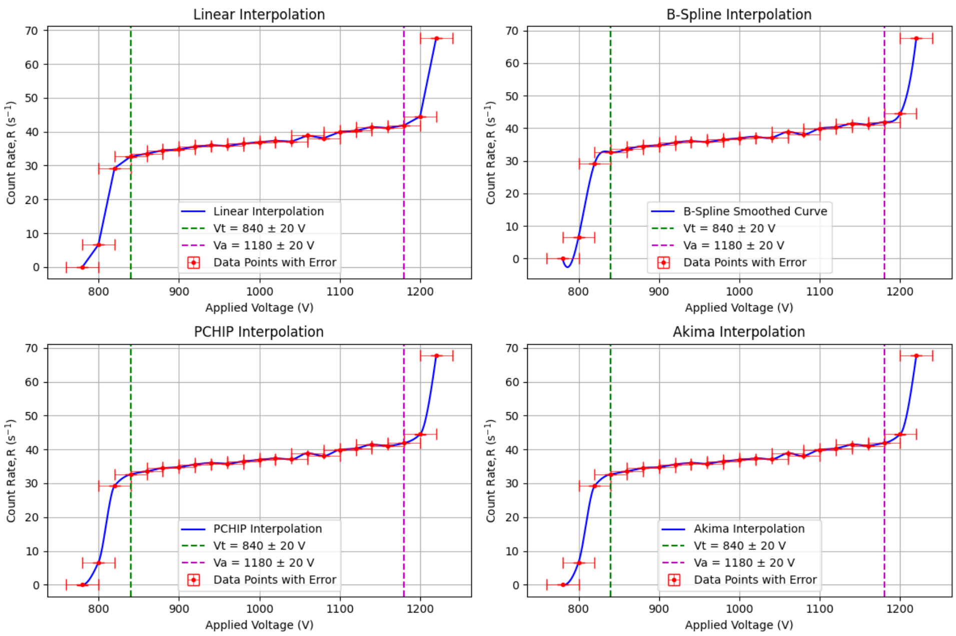
```
        v    n1    n2    n3        n_sum  count_rate
0     780     0     0     0     0.000000    0.000000
1     800   198   195   202   198.333333    6.611111
2     820   872   865   892   876.333333   29.211111
3     840   984   970   983   979.000000   32.633333
4     860  1000  1014  1002  1005.333333   33.511111
5     880  1035  1031  1037  1034.333333   34.477778
6     900  1041  1041  1047  1043.000000   34.766667
7     920  1076  1061  1064  1067.000000   35.566667
8     940  1079  1061  1102  1080.666667   36.022222
9     960  1056  1040  1125  1073.666667   35.788889
10    980  1069  1114  1101  1094.666667   36.488889
11   1000  1074  1138  1110  1107.333333   36.911111
12   1020  1137  1087  1138  1120.666667   37.355556
13   1040  1059  1122  1156  1112.333333   37.077778
14   1060  1164  1168  1165  1165.666667   38.855556
15   1080  1131  1149  1148  1142.666667   38.088889
16   1100  1122  1213  1256  1197.000000   39.900000
17   1120  1207  1215  1203  1208.333333   40.277778
18   1140  1244  1243  1240  1242.333333   41.411111
19   1160  1209  1256  1234  1233.000000   41.100000
20   1180  1285  1260  1223  1256.000000   41.866667
21   1200  1322  1334  1347  1334.333333   44.477778
22   1220  2030  2031  2036  2032.333333   67.744444
```

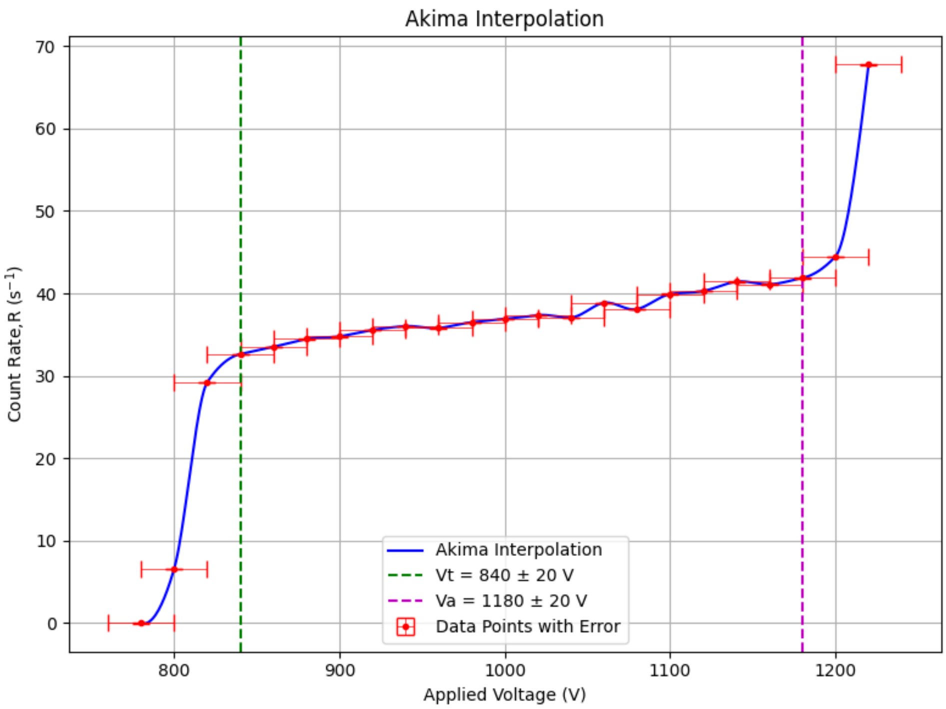
Graph plot of count rate against applied voltage and calculation:

```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import interp1d, splrep, splev, Akima1DInterpolator, PchipInterpolator

# Data from the image
v = np.array([780, 800, 820, 840, 860, 880, 900, 920, 940, 960, 980, 1000, 1020, 1040, 1060, 1080,
    1100, 1120, 1140, 1160, 1180, 1200, 1220])
rate = np.array([0, 6.61111111, 29.2111111, 32.6333333, 33.5111111, 34.4777778, 34.7777778,
    35.5666667, 36.0222222, 35.7888889, 36.4888889, 36.9111111,
                 37.3555556, 37.0777778, 38.8555556, 38.0888889, 39.9, 40.2777778, 41.4111111, 41.1,
    41.8666667, 44.4777778, 67.7444444])

# Error specifications
v_err = np.full_like(v, 20)  # Error of voltage is 20V
rate_err = np.full_like(rate, 0.03)  # Error of counting rate is 0.03

# Create a range of values for a smooth curve
v_smooth = np.linspace(v.min(), v.max(), 500)

# Linear Interpolation
linear_interp = interp1d(v, rate)
rate_smooth_linear = linear_interp(v_smooth)

# B-Spline Interpolation
spl = splrep(v, rate)
rate_smooth_bspline = splev(v_smooth, spl)

# PCHIP Interpolation
pchip_interp = PchipInterpolator(v, rate)
rate_smooth_pchip = pchip_interp(v_smooth)

# Akima Interpolation
akima_interp = Akima1DInterpolator(v, rate)
rate_smooth_akima = akima_interp(v_smooth)

def compute_geiger_plateau_slope(v, rate):
    # Calculate gradient
    gradients = np.gradient(rate, v)

    # Calculate the second derivative (change in gradients)
    second_gradients = np.gradient(gradients, v)

    # Identify the threshold voltage (V_t) where the gradient suddenly decreases to a small value
    V_t_index = np.argmin(second_gradients)
    V_t_index_real = V_t_index + 1  # Adjust to the next point after the minimum gradient

    # Identify the breakdown voltage (V_a) where the gradient suddenly increases to a large value
    V_a_index = np.argmax(second_gradients)
    V_a_index_real = V_a_index - 1  # Adjust to the point before the maximum gradient

    V_t = v[V_t_index_real]
    V_a = v[V_a_index_real]
    R_t = rate[V_t_index_real]
    R_a = rate[V_a_index_real]

    # Compute the slope using the given formula
    slope = ((R_a - R_t) / (0.5 * (R_a + R_t) * (V_a - V_t))) * 100

    return slope, V_t, V_a, R_t, R_a, V_t_index_real, V_a_index_real

# Compute the slope of the Geiger plateau
slope, V_t, V_a, R_t, R_a, V_t_index_real, V_a_index_real = compute_geiger_plateau_slope(v, rate)

# Calculate operating voltage
V_operating = (V_t + V_a) / 2
V_start = v[0]  # Starting voltage

# Plotting all data in one figure
plt.figure(figsize=(12, 8))

# Linear Interpolation
plt.subplot(2, 2, 1)
plt.plot(v_smooth, rate_smooth_linear, color='b', label='Linear Interpolation')
plt.errorbar(v, rate, xerr=v_err, yerr=rate_err, fmt='r.', label='Data Points with Error', capsize=5,
    linewidth=0.5)
plt.axvline(V_t, color='g', linestyle='--', label=f'Vt = {V_t} $\pm$ {v_err[V_t_index_real]} V')
plt.axvline(V_a, color='m', linestyle='--', label=f'Va = {V_a} $\pm$ {v_err[V_a_index_real]} V')
plt.title('Linear Interpolation')
plt.xlabel('Applied Voltage (V)')
plt.ylabel('Count Rate,R (s$^{-1}$)')
plt.legend()
plt.grid(True)

# B-Spline Interpolation
plt.subplot(2, 2, 2)
plt.plot(v_smooth, rate_smooth_bspline, color='b', label='B-Spline Smoothed Curve')
plt.errorbar(v, rate, xerr=v_err, yerr=rate_err, fmt='r.', label='Data Points with Error', capsize=5,
    linewidth=0.5)
plt.axvline(V_t, color='g', linestyle='--', label=f'Vt = {V_t} $\pm$ {v_err[V_t_index_real]} V')
plt.axvline(V_a, color='m', linestyle='--', label=f'Va = {V_a} $\pm$ {v_err[V_a_index_real]} V')
plt.title('B-Spline Interpolation')
plt.xlabel('Applied Voltage (V)')
plt.ylabel('Count Rate,R (s$^{-1}$)')
plt.legend()
plt.grid(True)

# PCHIP Interpolation
```

```python
plt.subplot(2, 2, 3)
plt.plot(v_smooth, rate_smooth_pchip, color='b', label='PCHIP Interpolation')
plt.errorbar(v, rate, xerr=v_err, yerr=rate_err, fmt='r.', label='Data Points with Error', capsize=5,
    linewidth=0.5)
plt.axvline(V_t, color='g', linestyle='--', label=f'Vt = {V_t} $\pm$ {v_err[V_t_index_real]} V')
plt.axvline(V_a, color='m', linestyle='--', label=f'Va = {V_a} $\pm$ {v_err[V_a_index_real]} V')
plt.title('PCHIP Interpolation')
plt.xlabel('Applied Voltage (V)')
plt.ylabel('Count Rate,R (s$^{-1}$)')
plt.legend()
plt.grid(True)

# Akima Interpolation
plt.subplot(2, 2, 4)
plt.plot(v_smooth, rate_smooth_akima, color='b', label='Akima Interpolation')
plt.errorbar(v, rate, xerr=v_err, yerr=rate_err, fmt='r.', label='Data Points with Error', capsize=5,
    linewidth=0.5)
plt.axvline(V_t, color='g', linestyle='--', label=f'Vt = {V_t} $\pm$ {v_err[V_t_index_real]} V')
plt.axvline(V_a, color='m', linestyle='--', label=f'Va = {V_a} $\pm$ {v_err[V_a_index_real]} V')
plt.title('Akima Interpolation')
plt.xlabel('Applied Voltage (V)')
plt.ylabel('Count Rate,R (s$^{-1}$)')
plt.legend()
plt.grid(True)

plt.tight_layout()

# Separate plot for Akima Interpolation
plt.figure(figsize=(8, 6))
plt.plot(v_smooth, rate_smooth_akima, color='b', label='Akima Interpolation')
plt.errorbar(v, rate, xerr=v_err, yerr=rate_err, fmt='r.', label='Data Points with Error', capsize=5,
    linewidth=0.5)
plt.axvline(V_t, color='g', linestyle='--', label=f'Vt = {V_t} $\pm$ {v_err[V_t_index_real]} V')
plt.axvline(V_a, color='m', linestyle='--', label=f'Va = {V_a} $\pm$ {v_err[V_a_index_real]} V')
plt.title('Akima Interpolation')
plt.xlabel('Applied Voltage (V)')
plt.ylabel('Count Rate,R (s$^{-1}$)')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()

# Compute the percentage difference between the standard value and the experimental value
standard_slope = 0.1  # standard slope value in % per volt
percentage_discrepancy = abs((slope - standard_slope) / standard_slope) * 100

print(f"Starting Voltage: {V_start} V")
print(f"Threshold Voltage (V_t): {V_t} $\pm$ {v_err[V_t_index_real]} V, Count Rate (R_t): {R_t:.2f} $\
    pm$ {rate_err[V_t_index_real]} 1/s")
print(f"Breakdown Voltage (V_a): {V_a} $\pm$ {v_err[V_t_index_real]} V, Count Rate (R_a): {R_a:.2f} $\
    pm$ {rate_err[V_a_index_real]} 1/s")
print(f"Operating Voltage: {V_operating:.0f} V")
print(f"Slope of the Geiger plateau: {slope:.2f}% per volt")
print(f"Percentage discrepancy between standard value (0.1% per volt) and experimental value: {
    percentage_discrepancy:.2f}%")
```

Output:






```
Starting Voltage: 780 V
Threshold Voltage (V_t): 840 $\pm$ 20 V, Count Rate (R_t): 32.63 $\pm$ 0.03 1/s
Breakdown Voltage (V_a): 1180 $\pm$ 20 V, Count Rate (R_a): 41.87 $\pm$ 0.03 1/s
Operating Voltage: 1010 V
Slope of the Geiger plateau: 0.07% per volt
Percentage discrepancy between standard value (0.1% per volt) and experimental value: 27.10%
```

Python code of PART B

Data processing and calculation of PART B:

```python
import pandas as pd
import numpy as np

# Load the data
data = {
    "n": list(range(1, 21)),
    "N": [1228, 1186, 1190, 1210, 1223, 1164, 1169, 1211, 1164, 1182, 1184, 1146, 1134, 1168, 1137,
     1189, 1158, 1151, 1124, 1132]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Calculate the Rate
df['Rate'] = df['N'] / 30

# Calculate the standard deviation
df['std_dev'] = (df['Rate'] / 30) ** 0.5

# Calculate the average Rate
average_rate = df['Rate'].mean()

# Calculate the average std dev
average_std_dev = df['std_dev'].mean()

# Calculate the overall std dev
overall_std_dev = (average_rate / 30) ** 0.5

# Calculate the percentage discrepancy between std_dev and overall_std_dev
percentage_discrepancy = abs(overall_std_dev - average_std_dev) / average_std_dev * 100

# Calculate the standard deviation of the overall standard deviation using statistical methods
std_dev_overall_std_dev = np.std(df['std_dev'], ddof=1)

# Calculate the confidence interval
confidence_interval_lower = overall_std_dev - std_dev_overall_std_dev
confidence_interval_upper = overall_std_dev + std_dev_overall_std_dev

# Count values within the confidence interval
values_within_confidence_interval = ((df['std_dev'] >= confidence_interval_lower) & (df['std_dev'] <=
     confidence_interval_upper)).sum()

# Calculate the confidence percentage
confidence_percentage = (values_within_confidence_interval / 20) * 100

# Print the DataFrame and the results
print("Calculated Rate and Standard Deviation:\n", df)
print("\nAverage Rate:", average_rate)
print("Average Standard Deviation:", average_std_dev)
print("Overall Standard Deviation:", overall_std_dev)
print("Percentage Discrepancy:", percentage_discrepancy)
print("Standard Deviation of Overall Std Dev:", std_dev_overall_std_dev)
print("Confidence Interval: [{:.5f}, {:.5f}]".format(confidence_interval_lower,
     confidence_interval_upper))
print("Values within Confidence Interval:", values_within_confidence_interval)
print("Confidence Percentage:", confidence_percentage)
```

Output:

```
Calculated Rate and Standard Deviation:
      n     N       Rate   std_dev
0    1  1228  40.933333  1.168094
1    2  1186  39.533333  1.147945
2    3  1190  39.666667  1.149879
3    4  1210  40.333333  1.159502
4    5  1223  40.766667  1.165714
5    6  1164  38.800000  1.137248
6    7  1169  38.966667  1.139688
7    8  1211  40.366667  1.159981
8    9  1164  38.800000  1.137248
9   10  1182  39.400000  1.146008
10  11  1184  39.466667  1.146977
11  12  1146  38.200000  1.128421
12  13  1134  37.800000  1.122497
13  14  1168  38.933333  1.139200
14  15  1137  37.900000  1.123981
15  16  1189  39.633333  1.149396
16  17  1158  38.600000  1.134313
17  18  1151  38.366667  1.130880
18  19  1124  37.466667  1.117537
19  20  1132  37.733333  1.121507

Average Rate: 39.083333333333336
Average Standard Deviation: 1.1413008051434625
Overall Standard Deviation: 1.1413929112175956
Percentage Discrepancy: 0.008070271546109698
Standard Deviation of Overall Std Dev: 0.014876684571997062
Confidence Interval: [1.12652, 1.15627]
Values within Confidence Interval: 12
Confidence Percentage: 60.0
```

Python code of PART C

Data processing of PART C:

```python
import pandas as pd

# Define the data
data = {
    'Type': ['Al', 'Al', 'Poly', 'Poly', 'Plastic', 'Plastic', 'Al', 'Al', 'Al', 'Al', 'Al', 'Al', 'Al', 'Al', 'Al', 'Al', 'Lead', 'Lead', 'Lead', 'Lead'],
    'Density(mg/cm^2)': [4.5, 6.5, 9.6, 19.2, 59.1, 102, 141, 170, 216, 258, 328, 425, 522, 645, 655, 840, 1120, 2066, 3448, 7367],
    '1': [1076, 1052, 1085, 1064, 833, 775, 577, 470, 326, 251, 134, 85, 46, 41, 35, 31, 29, 26, 24, 17],
    '2': [1090, 1073, 1130, 1038, 852, 755, 589, 472, 332, 252, 140, 79, 49, 38, 35, 32, 28, 28, 26, 19],
    '3': [1113, 1101, 1117, 1060, 826, 735, 585, 461, 337, 254, 142, 85, 50, 39, 34, 32, 29, 27, 23, 21],
    '4': [1075, 1008, 1064, 1058, 825, 752, 578, 458, 328, 251, 142, 93, 47, 37, 36, 30, 29, 28, 26, 16]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Calculate the mean and count rate
df['Mean'] = df[['1', '2', '3', '4']].mean(axis=1)
df['Count Rate'] = df['Mean'] / 30

# Format the count rate to 2 decimal places
df['Count Rate'] = df['Count Rate'].round(2)

# Display the results
print(df[['Type', 'Density(mg/cm^2)', '1', '2', '3', '4', 'Mean', 'Count Rate']])
```

Output:

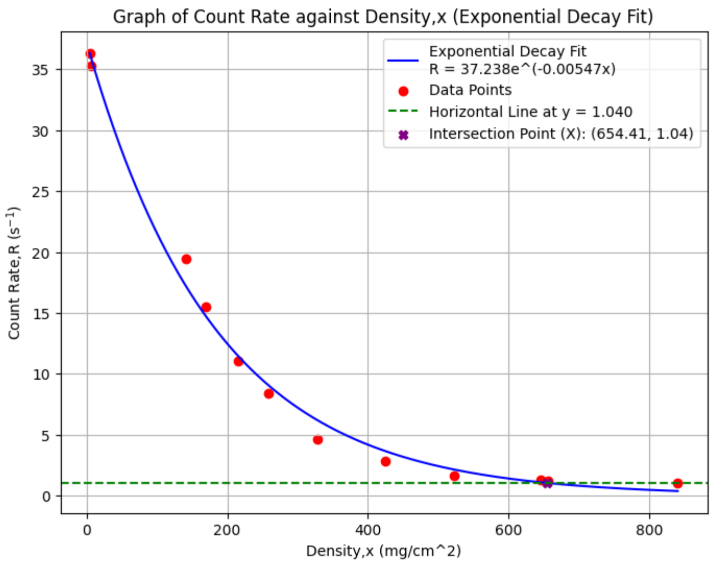
```
       Type  Density(mg/cm^2)     1     2     3     4     Mean  Count Rate
0        Al               4.5  1076  1090  1113  1075  1088.50       36.28
1        Al               6.5  1052  1073  1101  1008  1058.50       35.28
2      Poly               9.6  1085  1130  1117  1064  1099.00       36.63
3      Poly              19.2  1064  1038  1060  1058  1055.00       35.17
4   Plastic              59.1   833   852   826   825   834.00       27.80
5   Plastic             102.0   775   755   735   752   754.25       25.14
6        Al             141.0   577   589   585   578   582.25       19.41
7        Al             170.0   470   472   461   458   465.25       15.51
8        Al             216.0   326   332   337   328   330.75       11.02
9        Al             258.0   251   252   254   251   252.00        8.40
10       Al             328.0   134   140   142   142   139.50        4.65
11       Al             425.0    85    79    85    93    85.50        2.85
12       Al             522.0    46    49    50    47    48.00        1.60
13       Al             645.0    41    38    39    37    38.75        1.29
14       Al             655.0    35    35    34    36    35.00        1.17
15       Al             840.0    31    32    32    30    31.25        1.04
16     Lead            1120.0    29    28    29    29    28.75        0.96
17     Lead            2066.0    26    28    27    28    27.25        0.91
18     Lead            3448.0    24    26    23    26    24.75        0.82
19     Lead            7367.0    17    19    21    16    18.25        0.61
```

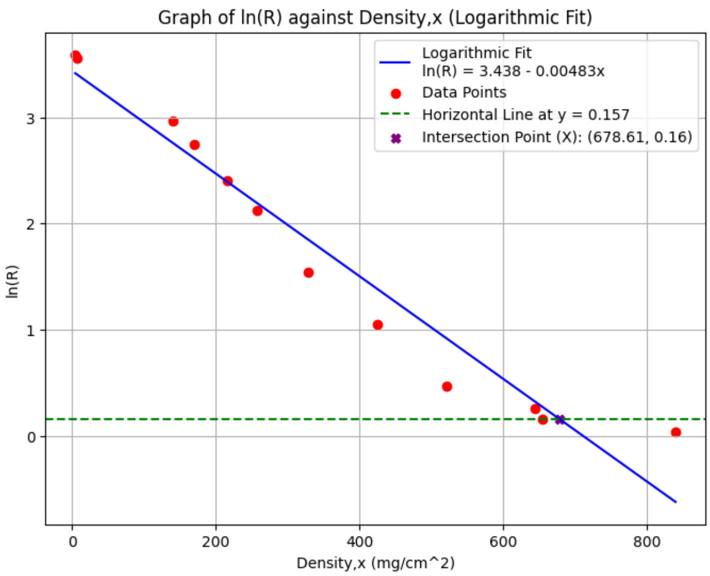
Graph ploting and calculation:

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
from sklearn.linear_model import LinearRegression

# Data
data = {
    'Density (mg/cm^2)': [4.5, 6.5, 141.0, 170.0, 216.0, 258.0, 328.0, 425.0, 522.0, 645.0, 655.0, 840.0],
    'Count Rate': [36.28, 35.28, 19.41, 15.51, 11.02, 8.40, 4.65, 2.85, 1.60, 1.29, 1.17, 1.04]
}

df = pd.DataFrame(data)
density = np.array(df['Density (mg/cm^2)'])
count_rate = np.array(df['Count Rate'])
log_count_rate = np.log(count_rate)

# Exponential decay function for curve fitting
def exponential_decay(x, R0, mu):
    return R0 * np.exp(-mu * x)

# Perform curve fitting with bounds to avoid overflow issues
params, covariance = curve_fit(exponential_decay, density, count_rate, bounds=(0, [100, 0.01]))
R0, mu = params

# Generate data for the fitted curve
density_smooth = np.linspace(density.min(), density.max(), 500)
count_rate_fitted = exponential_decay(density_smooth, R0, mu)

# Find the intersection point of the horizontal line with the fitted curve
y_horizontal_exp = count_rate[-1]
x_intersection_exp = (np.log(y_horizontal_exp / R0)) / (-mu)
y_intersection_exp = exponential_decay(x_intersection_exp, R0, mu)

# Plotting the fitted exponential decay curve
plt.figure(figsize=(8, 6))
plt.plot(density_smooth, count_rate_fitted, color='b', label=f'Exponential Decay Fit\nR = {R0:.3f}e^(-{mu:.5f}x)')
plt.scatter(density, count_rate, color='r', label='Data Points')
plt.axhline(y=y_horizontal_exp, color='g', linestyle='--', label=f'Horizontal Line at y = {y_horizontal_exp:.3f}')
plt.scatter(x_intersection_exp, y_intersection_exp, color='purple', marker='X', label=f'Intersection Point (X): ({x_intersection_exp:.2f}, {y_intersection_exp:.2f})')
plt.xlabel('Density,x (mg/cm^2)')
plt.ylabel('Count Rate,R (s$^{-1}$)')
plt.title('Graph of Count Rate against Density,x (Exponential Decay Fit)')
plt.legend()
plt.grid(True)
plt.show()

# Linear regression for the logarithmic data
model = LinearRegression()
density_reshaped = density.reshape(-1, 1)
model.fit(density_reshaped, log_count_rate)
lnR0 = model.intercept_
mu_log = -model.coef_[0]

# Generate data for the fitted log curve
log_count_rate_fitted = model.predict(density_smooth.reshape(-1, 1))

# Find the intersection point of the horizontal line with the fitted log curve
y_horizontal_log = log_count_rate[-2]
x_intersection_log = (lnR0 - y_horizontal_log) / mu_log
y_intersection_log = lnR0 - mu_log * x_intersection_log

# Plotting the logarithmic fit
plt.figure(figsize=(8, 6))
plt.plot(density_smooth, log_count_rate_fitted, color='b', label=f'Logarithmic Fit\nln(R) = {lnR0:.3f} - {mu_log:.5f}x')
plt.scatter(density, log_count_rate, color='r', label='Data Points')
plt.axhline(y=y_horizontal_log, color='g', linestyle='--', label=f'Horizontal Line at y = {y_horizontal_log:.3f}')
plt.scatter(x_intersection_log, y_intersection_log, color='purple', marker='X', label=f'Intersection Point (X): ({x_intersection_log:.2f}, {y_intersection_log:.2f})')
plt.xlabel('Density,x (mg/cm^2)')
plt.ylabel('ln(R)')
plt.title('Graph of ln(R) against Density,x (Logarithmic Fit)')
plt.legend()
plt.grid(True)
plt.show()

# Printing the results
print(f"The range of \Beta particles in aluminium (\Beta1) is the x-value of intersection points of first graph: {x_intersection_exp:.2f} mg/cm^2")
print(f"The range of \Beta particles in aluminium (\Beta2) is the x-value of intersection points of second graph: {x_intersection_log:.2f} mg/cm^2")

# Calculating the percentage difference
percentage_difference = abs(x_intersection_exp - x_intersection_log) / ((x_intersection_exp + x_intersection_log) / 2) * 100
print(f"Percentage difference between \Beta1 and \Beta2 = {percentage_difference:.2f} %")

# Given equations and calculations
print(f"\mu_m_exp = {mu:.8f} cm^2/mg")

print(f"\mu_m_log = {mu_log:.8f} cm^2/mg")
```

```
# Calculating the percentage difference for \mu
percentage_difference_mu = abs(mu - mu_log) / ((mu + mu_log) / 2) * 100
print(f"Percentage difference between \mu from two formulas = {percentage_difference_mu:.2f} %")

# Calculating the half-thickness value (X1/2)
X_half2 = -np.log(1/2) / mu_log
print(f"The half-thickness value (X_1/2) for \Beta particles in the aluminium absorber (\Beta2) is {X_half2:.2f} mg/cm^2")
```

Output:






```
The range of $\beta$ particles in aluminium ($\beta$1) is the x-value of intersection points of first graph: 654.41 mg/cm^2
The range of $\beta$ particles in aluminium ($\beta$2) is the x-value of intersection points of second graph: 678.61 mg/cm^2
Percentage difference between $\beta$1 and $\beta$2 = 3.63 %
$\mu$_m_exp = 0.00546768 cm^2/mg
$\mu$_m_log = 0.00483452 cm^2/mg
Percentage difference between $\mu$ from two formulas = 12.29 %
The half-thickness value (X_1/2) for $\beta$ particles in the aluminium absorber ($\beta$2) is 143.37 mg/cm^2
```