Poisson Statistics

Şükrü Çağlar, Ertuğrul Yavuz, Eren Önen, Ali Oğuzhan Kumrulu

PHYS442 Spring 2025 Experiment Report

Abstract:
In this experiment, we explored the Poisson distribution as a special case of the binomial distribution. We also investigated the differences between the Poisson and Gaussian distributions, focusing on their behavior under varying mean values. The primary objective is to assess how well the Gaussian distribution approximates the Poisson distribution as the mean increases—a convergence expected from statistical theory-. Additionally, the study aimed to evaluate the suitability of the Poisson distribution for modeling the time distribution between independent random events. Using a GM counter and a chart recorder, we investigated gamma radiation emission events from different sources with varying mean emission rates over a fixed time interval. We also recorded a continuous sequence of 103 emissions to obtain the time distribution between events. Altogether, these datasets allowed us to successfully demonstrate the relationship between variance and mean in the Poisson distribution, the convergence behavior between the Poisson and Gaussian distributions as the mean increases, and finally, the effectiveness of the time-dependent Poisson distribution in modeling inter-event timing.

I Introduction and Theory

History: The study of the Poisson distribution has its roots in the early 19th century, introduced by Siméon Denis Poisson to model the probability of rare events over fixed intervals. Its relevance grew significantly with the development of nuclear physics, where it provided a mathematical foundation for describing random processes such as radioactive decay. In particular, early experiments using simple detectors like the Wulf electroscope laid the groundwork for modern understanding of discrete stochastic events. The motivation behind this experiment is to explore how well the Poisson distribution models real-world random processes, such as gamma radiation detection, and to examine under what conditions it can be approximated by the Gaussian distribution. This investigation not only reinforces key concepts in probability theory but also provides insight into the statistical nature of nuclear phenomena.

Part A: Poisson Distribution

Poisson distribution is a type of discrete probability distribution. It is a special case of binomial distribution:

$$B(n;m,p) = \binom{m}{n} p^n (q)^{m-n} \tag{1}$$

where the number fo trials m is infinitely large and probability of a success p is infinitesimally small. In that case, above distribution can be expressed as:

$$P(\lambda, n) = \frac{\lambda^n e^{-\lambda}}{n!} \tag{2}$$

Where λ is the mean of the distribution and n is the number of desired counts of an event in m number of trials. One can observe by taking the limit as $m \to \infty$, this result comes up naturally from the binomial distribution. Detailed derivation is given in **Appendix 1**. Since poisson distribution is a kind of binomial distribution, it is normalized to the unity:

$$\sum_n P(\lambda, n) = 1 \tag{3}$$

with its variance being:

$$\sigma^2 = \lambda$$

This distribution conveniently approximates to gaussian distribution as λ gets large. Gaussian distribution is another special case of binomial distribution where m becomes large and probability distribution becomes continuous. This kind of distribution is of great importance in probability theory since central limit theorem suggests that the sum (or average) of a large number of independent, identically distributed random variables tends to follow a Normal distribution, even if the original variables themselves are not normally distributed. As long as each variable has finite mean and variance, a discrete distribution like poisson distribution approaches to a normal distribution when the number of variables is large. This phenomenon constitutes our main objective in this experiment which is to observe differences between two distributions with different λ values.

Part B: Distribution of Successive events

However, poisson distribution behaves more differently when it comes to distribution of successive events. If we express the poisson distribution in a time dependent way:

$$P(\lambda, n) = P(\alpha, t, n) = \frac{(\alpha t)^n e^{-\alpha t}}{n!} \tag{4}$$

where α represents the rate of events per unit time, we can find out the probability of having an event occurring in a time interval dt.

$$P(\alpha, dt, 1) = \frac{(\alpha dt) e^{-\alpha dt}}{1!} \tag{5}$$

one more useful step would be calculating the probability of having n events in a time t followed by another event in time interval dt:

$$\begin{aligned} P_q(n+1, t)\, dt &= P(\alpha, t, n) P(\alpha, dt, 1) \\ &= \frac{(\alpha t)^n e^{-\alpha t}}{n!} \cdot \frac{(\alpha dt)^1 e^{-\alpha dt}}{1!} \\ &\simeq \frac{(\alpha t)^n e^{-\alpha t} \alpha dt}{n!} \end{aligned}$$

if we assume $e^{-\alpha dt} \simeq 1$. Finally, when dt is reduced from both sides of the equation, we end up with:

$$P_q(n+1, t) = \frac{(\alpha t)^n e^{-\alpha t} \alpha}{n!} \tag{6}$$

This provides us with the poisson probability for having n successive events in a time interval t which also approximates to normal distribution as n gets larger. In our experiment however, we shall investigate the cases for $n = 0$ and $n = 1$.

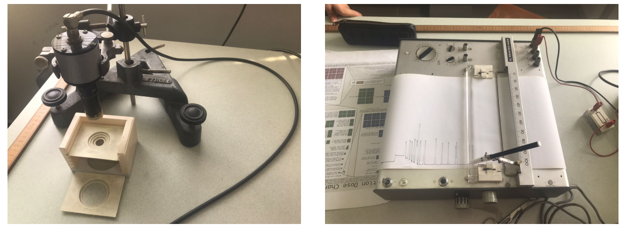
II Setup and Method

Setup:

The experimental setup consists of:

- Geiger Counter
- Sample tray whit adjustable positions
- Gamma-ray sources (Cesium-137 and Barium-133)
- Chart recorder

Geiger Counter is set 5 to 10cm above the tray, catching gamma radiation from the samples. Chart recorder is also connected to the counter for a time series record of detections spanning around 100 seconds.



(a) Geiger-Muller Tube (b) Chart recorder

Figure 1: Geiger-tube and chart recorder

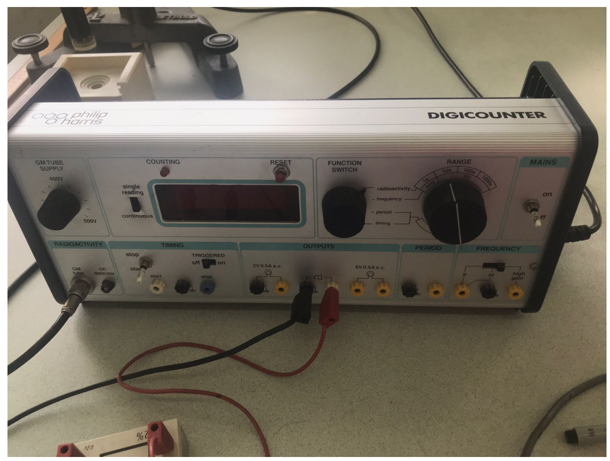


Figure 2: Counter for the Geiger-Tube and chart recorder

Figure 3: Barium-133 and Cesium-137 as gamma-ray sources

Method:

The experiment consist of 2 parts and was conducted following these steps:

Part A: In this part of the experiment, we investigate how does the distribution of gamma-ray detection event behaves. specifically, we hope to observe how does the distribution changes from poisson to normal as the mean of the distribution increases.

1. A radioactive source was selected and placed on the tray beneath the GM tube. The Geiger counter was set to operate in radioactivity and single mode with a counting interval of 100 seconds. The applied high voltage (HV) was varied in 20 V increments, and the voltage range at which the count rate seems to plateau is determined. A voltage value within this plateau region was selected as the operating voltage of the detector to ensure stable performance.

2. The GM tube was then set to operate in continuous mode using the previously determined operating voltage. The counting interval was reduced to 10 seconds.

3. C-137 gamma-ray source and was positioned to yield approximately 100 counts per 10 seconds. The number of counts was recorded over 100 10-second intervals.

4. The counting interval was reduced to 1 second, and the measurements were repeated under the same conditions to achieve higher time resolution.

5. Previous two step is repeated for Ba-133 gamma-ray source to achieve a count rate of 20–30 counts per 10-second interval. The same data collection procedure as described in steps 3 and 4 was followed.

6. 4 datasets with 100 data points each is thus obtained for gamma radiation event. Each dataset contributes to analysis of gaussian and poisson behaviour for different λ values. Expected differences or similarities are explained in Theory.

Part B: This part of the experiment focuses on the distribution fo time intervals between counts. To achieve that, a continuous stream of detections is listed using a chart recorder.

1. Ba-133 is placed on to the tray in a position to give about 1 count per second under the GM tube.

2. Chart recorder is connected to the counter and detected gamma-rays are recorded as pulses in the recorder paper.

3. Chart recorder runs for about 2 minutes and obtains approximately 100 pulses.

4. Since the chart recorder runs with constant speed, distance between each peak gives us a measure in terms of absolute time between the detections.

5. Paper is removed from the chart recorder and distance between each pulse is measured and recorded. As mentioned before this values will be used to observe the time distribution between successive pulses(n=0 case). By adding the two successive intervals, n=1 case is also obtained from the dataset.

III Data

There are 5 datasets in this experiment used in the analysis of poisson statistics. 4 datasets of 100 data points from 2 gamma-ray sources are used to study the effects of mean value. Mean value and deviation of those datasets as well as a simple histogram are presented below. Additionally, full list of those datasets are presented in **data list**.

Dataset	Mean	Standard Deviation
Cs-137 10s	192.94	13.76
Cs-137 1s	19.19	4.29
Ba-133 10s	22.54	4.82
Ba-133 1s	2.46	1.58

Table 1: Mean and Standard Deviation of the Datasets

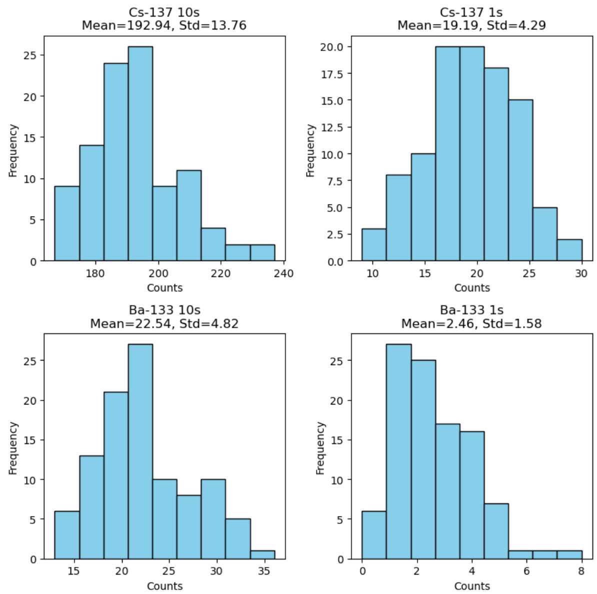


Figure 4: 4 distribution obtained from the experiment with 100 samples each

The other dataset is the timeseries output from the chart recorder used in the Part B of the experiment. It contains 107 data points, spanning around 2 minutes. Some part of that chart is presented below. n=o case is obtained directly from reading the chart and n=1 case is calculated by concatenating two data points in n=0 case. Full list datasets in both cases can be found in **data list**.

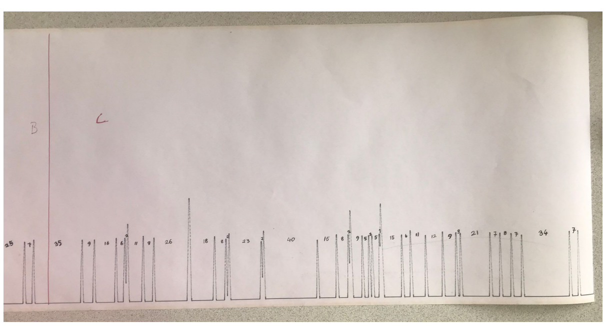


Figure 5: Some part of the chart recorder output

IV Analysis

Operating voltage:

Gm counter operates in a range form 300 to 500 Volts. The optimal operating voltage where potential difference is enough to allow a complete discharge along the anode for each detected radiation count and not high enough to create cascading electron avalanches and continuous discharge where the tube cannot detect radiation, and may be damaged. This region is called the "Geiger Plateau" and Gm-Tube operates in a voltage around the middle of that plateau for reliability. We have taken counts of the same gamma-ray source under 20V intervals from 300-500V region to determine the optimal operating voltage which is found to be around 440V.

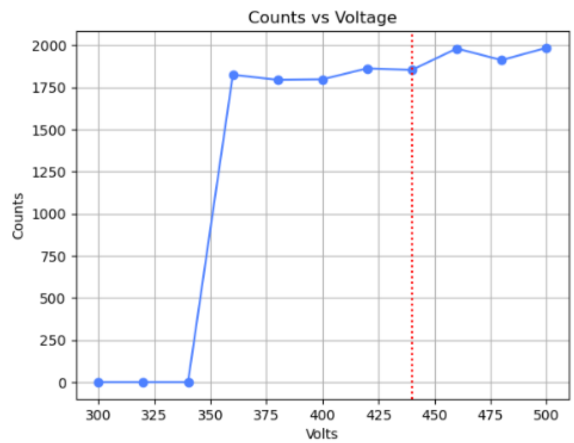


Figure 6: Geiger Region and operating voltage

Part A:

4 datasets recorded in the first part of the experiment with Cesium-137 and Barium-133 are represented in histograms below. Their mean and deviation values are also known. A poisson and Gaussian fit is applied to each distribution and their fitted mean and deviation values are compared with each other as well as the actual value.

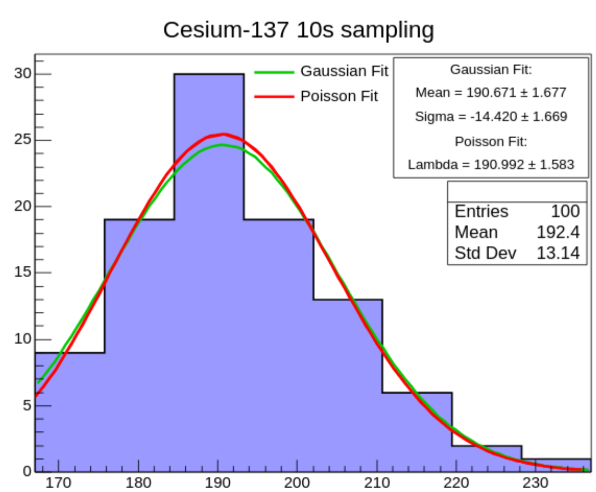


Figure 7: Cesium-137 (about 190 counts per 10s)

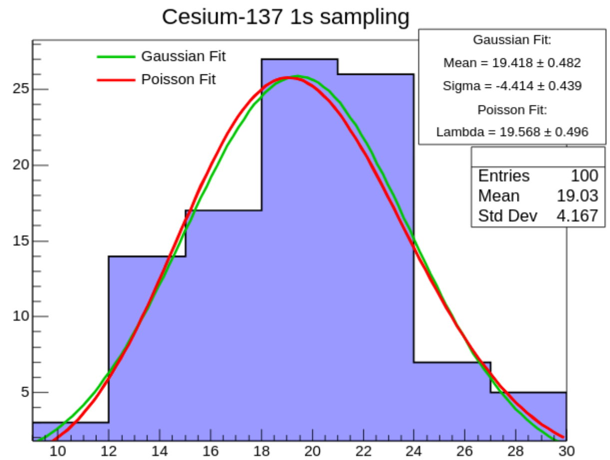


Figure 8: Cesium-137 (about 19 counts per 1s)

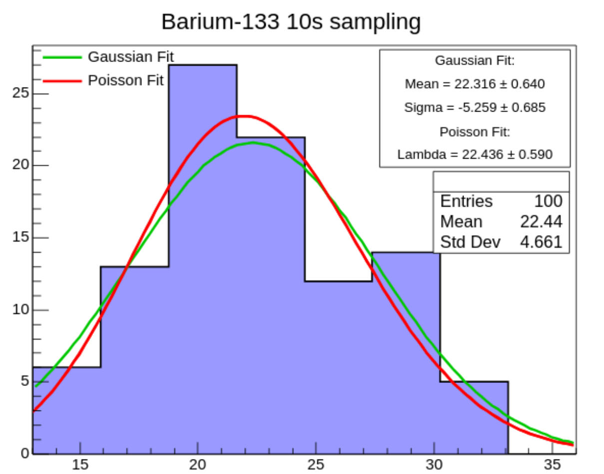


Figure 9: Barrium-133 (about 22 counts per 10s)

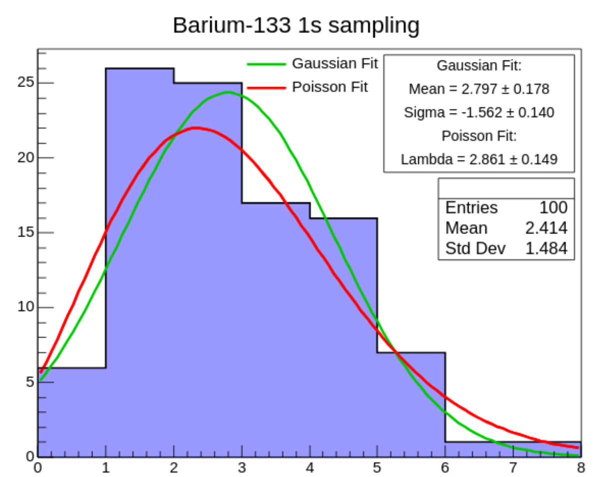


Figure 10: Barrium-133 (about 3 counts per 1s)

It is also possible to calculate a probability value for the fitness of gaussian and poisson distributions using χ^2 value.

Mean Value	$\boldsymbol{\chi^2}$/ndf	**Probability**
190.67	$2.08706/5 = 0.417$	0.837
22.32	$5.39116/4 = 1.348$	0.249
19.42	$5.06427/4 = 1.266$	0.281
2.80	$6.83144/5 = 1.366$	0.233

Table 2: Gaussian fit probabilities for each mean value

Mean Value	$\boldsymbol{\chi^2}$/ndf	**Probability**
190.99	$1.91326\ /\ 6 = 0.319$	0.927
22.44	$5.15318\ /\ 5 = 1.031$	0.397
19.57	$5.40501\ /\ 5 = 1.081$	0.368
2.86	$9.03362\ /\ 6 = 1.506$	0.172

Table 3: Poisson fit probabilities for each mean value

One can observe that the Poisson distribution achieves a lower χ^2/ndf score and has a higher probability compared to the Gaussian distribution. Another important behavior to note is the degradation of probability as the mean value decreases. Although this trend is present in both the Poisson and Gaussian distributions, the Poisson distribution appears to represent the data more accurately than the Gaussian up to a mean value of approximately 2.5.

The inconsistency observed in the last batch of the dataset may be due to an inadequate sample size. Since counting gamma radiation using a GM counter is a discrete process, many regions in the probability distribution remain unfilled (or result in empty bins, one might say) when the sample size is small. These empty bins affect the χ^2 calculations, as the test assumes a continuous distribution, leading to low probabilities for both Poisson and Gaussian estimates.

However, a visual assessment can still be made to distinguish the separation between the Gaussian and Poisson distributions for small mean values, regardless of the calculated probabilities.

Plotting $\sqrt{\lambda}/\sigma$ vs λ:

We deduced in the **Appendix 1** that the variance of poisson distribution is equal to square root of its mean. We can validate this finding by making a plot of $\sqrt{\lambda}/\sigma$ vs λ which in ideal case, should give a slope of 0 and intercept at 1. To remind the reader, the means and variances of the datasets are given in the table below:

μ	σ	$\sqrt{\lambda}/\sigma$
192.4	13.14	1.055
22.44	4.661	1.016
19.03	4.167	1.046
2.414	1.484	1.047

Table 4: Values of μ and σ^2 used in the analysis.

If we apply a linear fit to this 4 data points we obtain a line with slope 9.5×10^{-5} and intercept 1.036 which coincides with the expectation.

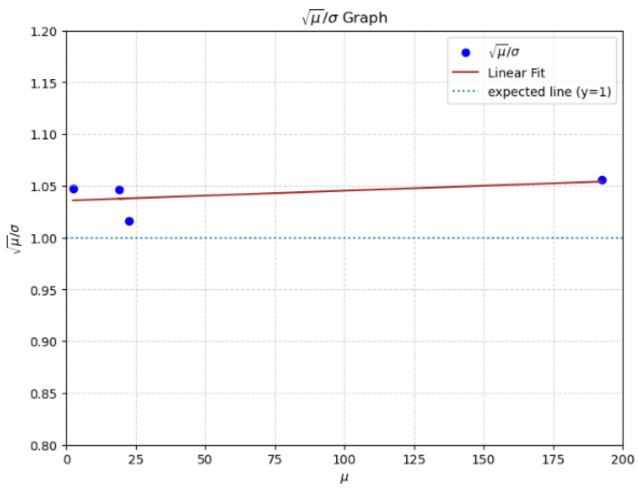


Figure 11: Plot for the relation between variance and mean

Part B:

As explained in theory, poisson distribution can be useful to calculate the probability distribution of time intervals between events. In order to use 6 as a template for n=0 and n=1 cases, value for α needs to be calculated. This is possible by simply dividing the total number of counts to the interval of the measurement T (in millimeters) since chart recorder records with constant speed. For the case of this experiment:

$$\alpha = \frac{counts}{T} = \frac{103}{1119} = 0.09205$$

This result is used to create theoretical distributions which are shown with red in histograms below. Moreover, a function fit with the template in accordance with 6 is used to see if actual distribution resembles what we expect.

for n=0: The version of 6 for the n=0 case is

$$P_q(1,t) = e^{-\alpha t}\alpha$$

We found the experimental α value to be 0.1096 ± 0.0138 in this case. Distribution is in well agreement with the theory with the probability of representing the distribution for the equation above is measured to be 0.7884.

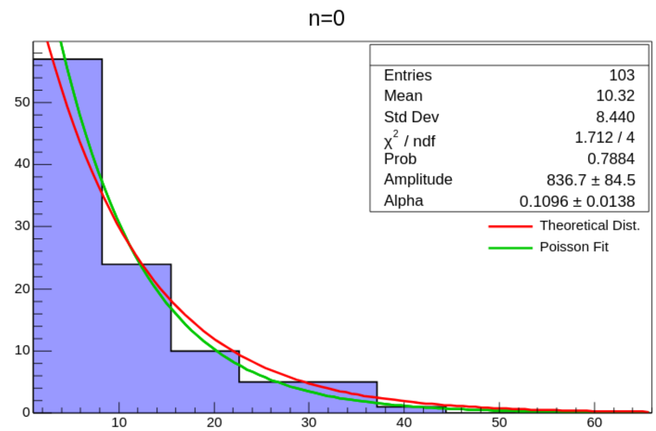


Figure 12: n=0 case for time dependent poisson dist.

for n=1: The version of 6 for the n=1 case is

$$P_q(1,t) = e^{-\alpha t}\alpha^2 t$$

We found the experimental α value to be 0.09657 ± 0.01138 in this case. Distribution is once more in well agreement with the theory and the shapes of the fitted and theoretical poisson distribution matches to a good extend. Probability of representing the distribution for the equation above is measured to be 0.8330.

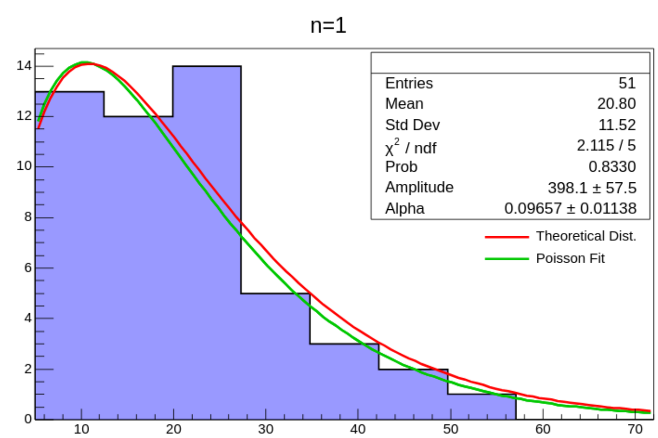


Figure 13: n=1 case for time dependent poisson dist.

V Conclusion

The experiment successfully demonstrated the effectiveness of the Poisson distribution in representing discrete Bernoulli events that occur independently. The conditions under which the Poisson distribution approximates the Gaussian distribution were also verified, thanks to the decreasing difference between the two distributions as the number of Bernoulli events increased. In Part A, datasets with higher mean values (e.g., 190 counts per 10 seconds) showed strong agreement with both distributions; however, the Poisson fit consistently yielded lower χ/ndf values and higher fit probabilities, indicating a more accurate representation of the discrete nature of the events. As the mean decreased, the accuracy of the Gaussian approximation declined significantly, and for the lowest mean (2.8), the fit quality of both distributions degraded—though the Poisson distribution remained superior. One potential improvement would be to use larger sample sizes for each mean value, allowing for more precise histogram binning.

In Part B, the time-dependent Poisson probability equations for successive events (n = 0 and n = 1) were tested against experimental data obtained from the chart recorder. The theoretical expressions derived from the Poisson process matched the observed distributions remarkably well, with high agreement probabilities (0.7884 for n = 0 and 0.8330 for n = 1). The α values derived from the fits were also reasonably close—within 1 to 2 standard deviations—of the actual value $\alpha = 0.09205$ (0.1096 ± 0.0138 for n = 0 and 0.09657 ± 0.01138 for n = 1). These results confirm the applicability of the Poisson process in modeling inter-arrival times of independent random events and validate the theoretical framework used in time-based event analysis

References

[1] E. Gülmez, *Advanced Physics Experiments.* Boğaziçi University, 1997.

[2] GitHub repository for codes and scripts used in this experiment: https://github.com/capta1Nemo/PHYS442-experiments.git

[3] Wikipedia contributors. (n.d.). *Poisson distribution.* In Wikipedia. `https://en.wikipedia.org/wiki/Poisson_distribution`

Appendix

A.1 Derivation of Poisson Distribution

We can express the binomial distribution as:

$$P(X = k) = \binom{m}{n} p^n (q)^{m-n} = \frac{m!}{n!(m-n)!} p^n (q)^{m-n}$$

Before we take limit as $m \to \infty$, we will make a substitution using the fact $\lambda = mp$. This simply represents the mean of distribution since mean value of a binomial distribution is given as the probability of desired event (p) times the number of trials (m). Then we obtain:

$$P(\lambda, n) = \lim_{m\to\infty} \frac{m!}{n!(m-n)!} \left(\frac{\lambda}{m}\right)^n \left(1 - \frac{\lambda}{m}\right)^{m-n}$$

If we simplify further:

$$P(\lambda, n) = \lim_{m\to\infty} \left(\frac{\lambda^n}{n!}\right) \frac{m!}{m^n(m-n)!} \left(1 - \frac{\lambda}{m}\right)^{m-n}$$

we should observe some of the terms in limit reduce to 1:

$$\lim_{m\to\infty} \frac{m!}{m^n(m-n)!} = 1$$

$$\lim_{m\to\infty} \left(1 - \frac{\lambda}{m}\right)^{-n} = 1$$

and using the approximation for

$$e = \lim_{x\to\infty} \left(1 + \frac{1}{x}\right)^x$$

we have:

$$\left(1 - \frac{\lambda}{m}\right)^m \to e^{-\lambda} \quad \text{as } m \to \infty$$

So we get:

$$P(\lambda, n) \approx \frac{\lambda^n}{k!} \cdot e^{-\lambda} = \frac{\lambda^k e^{-\lambda}}{k!}$$

Which is exactly the formula for poisson distribution.

Variance of poisson distribution can also be approximated as:

$$\sigma^2 = \lim_{m\to\infty} mp(1-p) = m\left(\frac{\lambda}{m}\right)\left(1 - \frac{\lambda}{m}\right) = \lambda$$

Additional Material

Full list of datasets

The dataset in text format can also be found in [2].

Full dataset with 100 samples each							
Cs-137 10s		**Ba-133 10s**		**Cs-137 1s**		**Ba-133 1s**	
196	193	31	20	23	22	5	4
235	205	19	20	23	22	1	1
184	170	19	25	27	16	1	4
199	182	17	20	17	20	1	1
189	186	28	22	19	12	4	3
186	189	22	23	17	21	3	1
194	186	23	17	22	12	6	2
186	215	16	19	20	19	2	0
196	191	27	29	21	27	1	1
210	199	20	20	23	25	5	2
197	181	18	25	20	19	1	4
181	186	21	21	16	24	4	4
177	193	19	28	11	21	5	0
182	192	15	20	23	21	2	7
193	194	21	19	21	21	1	3
182	183	22	16	21	23	3	2
182	182	27	22	13	20	3	2
184	174	22	20	15	19	2	1
186	197	29	30	17	21	2	1
185	187	21	22	24	22	3	5
174	173	18	26	27	16	3	1
185	210	29	32	13	20	2	4
192	196	21	24	14	20	1	2
176	194	18	18	29	19	2	2
213	224	25	19	25	22	8	2
192	185	17	24	12	18	3	0
180	196	29	18	13	12	4	0
186	218	25	30	9	14	3	4
193	189	36	20	15	17	4	5
214	193	23	25	22	16	1	2
189	203	20	25	17	16	1	4
212	195	18	23	18	17	1	1
196	210	21	19	20	18	2	4
206	182	20	23	14	18	2	2
188	195	16	13	21	27	4	2
207	223	33	22	20	20	1	3
191	173	23	15	22	17	2	2
202	237	29	23	30	22	3	1
183	183	23	20	14	18	3	1
178	178	15	13	19	19	1	2
194	201	33	25	16	19	2	5
194	167	19	22	18	14	0	1
173	169	23	27	15	24	1	2
190	209	30	25	13	10	3	2
182	209	30	29	22	18	3	0
194	173	28	23	26	19	2	1
187	200	28	31	23	23	3	4
204	206	17	23	19	15	4	1
210	200	20	19	24	24	5	1

Table 5: Dataset for Part A

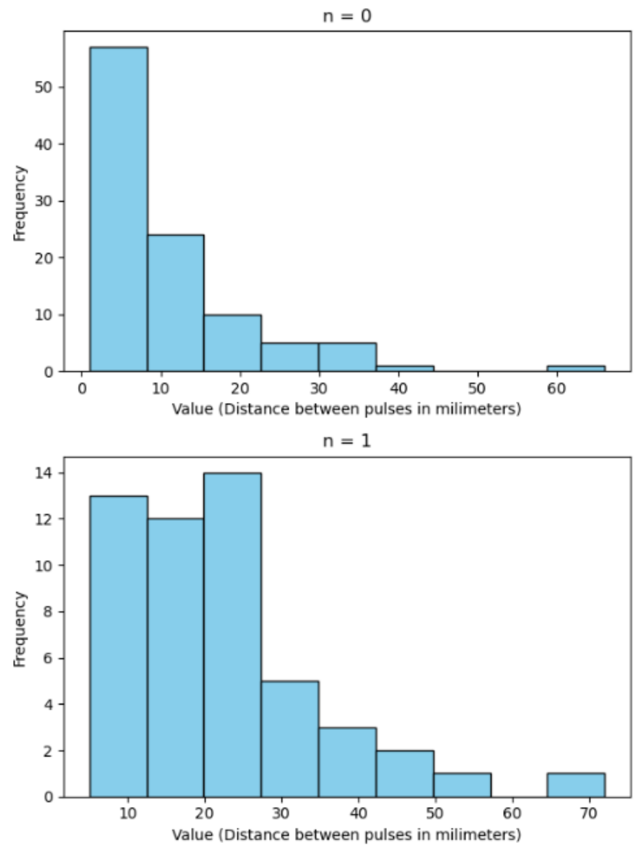


Figure 14: Distribution of unit time between successive pulses obtained from the chart recorder for n=0 and n=1 cases

Codes

All the codes here and more can be found in the GitHub repository [2].

```python

from ROOT import TH1F, TF1, TMath
data = np.loadtxt("outputa.txt", delimiter="\t", skiprows=1)
hist_list=[]
bin_list=[8,7,8,8]
for i in range(4):
    col1_data = data[:, i]

    hist = TH1F("", f"Dataset{i+1}", bin_list[i], min(col1_data), max(col1_data))
    hist.SetDirectory(0)
    for value in col1_data:
        hist.Fill(value)

    gauss_fit = TF1("gauss_fit", "gaus", min(col1_data), max(col1_data))
    gauss_fit.SetParameters(1, (min(col1_data) + max(col1_data)) / 2.0, 4)
    gauss_fit.SetParName(0, "GAmplitude")
    gauss_fit.SetParName(1, "GMean")
    gauss_fit.SetParName(2, "GSigma")
    gauss_fit.SetLineColor(TColor.GetColor(0, 200, 0))
    gauss_fit.SetLineWidth(3)
    result=hist.Fit(gauss_fit,"S")
    chi2_gauss = result.Chi2()
    ndf_gauss = result.Ndf()
    prob_gauss = TMath.Prob(chi2_gauss, ndf_gauss)
    print(f"GaussianFit:mean={gauss_fit.GetParameter(1):.5g},sigma={gauss_fit.GetParameter(2):.5g},prob={prob_gauss:.5f}")
    poisson_fit = TF1("poisson_fit", "[0]*TMath::Poisson(x,[1])", 0, max(col1_data))
    poisson_fit.SetParameters(1, (min(col1_data) + max(col1_data)) / 2.0)
    poisson_fit.SetParName(0, "PAmplitude")
    poisson_fit.SetParName(1, "PLambda")
    poisson_fit.SetLineColor(TColor.GetColor(255, 0, 0))
    poisson_fit.SetLineWidth(3)
    result=hist.Fit(poisson_fit,"S")
    chi2_poss = result.Chi2()
    ndf_poss = result.Ndf()
    prob_poss = TMath.Prob(chi2_poss, ndf_poss)
    print(f"PoissonFit:mean={poisson_fit.GetParameter(1):.5g},sigma={np.sqrt(poisson_fit.GetParameter(1)):.5g},prob={prob_poss:.5f}")
    hist_list.append(hist)
```

Code 1: PyRoot code for calculating the probabilties and best parameters for gaussian and poisson

```python

import numpy as np
from ROOT import TH1F, TF1, TCanvas, gStyle, TColor, TPaveText, TLegend
gStyle.SetOptStat(1)   # No default statistics box
gStyle.SetOptFit(0)    # No automatic fit results
gStyle.SetTitleFont(42, "xyz") # Nicer font
gStyle.SetLabelFont(42, "xyz")
gStyle.SetTitleSize(0.04, "xyz")
gStyle.SetLabelSize(0.03, "xyz")
gStyle.SetPadTopMargin(0.08)
gStyle.SetPadBottomMargin(0.12)
gStyle.SetPadLeftMargin(0.12)
gStyle.SetPadRightMargin(0.08)
c = TCanvas("", "", 800, 600)
data = np.loadtxt("outputa.txt", delimiter="\t", skiprows=1)
hist_list=[]
bin_list=[8,7,8,8]
title_list=["Cesium-137-10","Cesium-137-1","Barium-133-10","Barium-133-1"]
# adjust here [0..3]
for i in [1]:
    col1_data = data[:, i]

```

```
    hist = TH1F(f"", f"{title_list[i]}s-sampling", bin_list[i], min(col1_data), max(
        col1_data))
    hist.SetDirectory(0)
    hist.SetFillColorAlpha(4, 0.4)  # Blue fill with transparency
    hist.SetLineColor(1)
    hist.SetLineWidth(2)
    hist.SetFillStyle(1001)

    for value in col1_data:
        hist.Fill(value)

    # --- Gaussian fit ---
    gauss_fit = TF1(f"gauss_fit_{i}", "gaus", min(col1_data), max(col1_data))
    gauss_fit.SetParameters(1, (min(col1_data) + max(col1_data)) / 2.0, 4)
    gauss_fit.SetParName(0, "GAmplitude")
    gauss_fit.SetParName(1, "GMean")
    gauss_fit.SetParName(2, "GSigma")
    gauss_fit.SetLineColor(TColor.GetColor(0, 200, 0))  # Green
    gauss_fit.SetLineWidth(3)
    hist.Fit(gauss_fit,"Q")

    gauss_params = [gauss_fit.GetParameter(j) for j in range(3)]
    gauss_errors = [gauss_fit.GetParError(j) for j in range(3)]

    # --- Poisson fit ---
    poisson_fit = TF1(f"poisson_fit_{i}", "[0]*TMath::Poisson(x,[1])", 0, max(col1_data))
    poisson_fit.SetParameters(1, (min(col1_data) + max(col1_data)) / 2.0)
    poisson_fit.SetParName(0, "PAmplitude")
    poisson_fit.SetParName(1, "PLambda")
    poisson_fit.SetLineColor(TColor.GetColor(255, 0, 0))  # Red
    poisson_fit.SetLineWidth(3)
    hist.Fit(poisson_fit,"Q")

    poisson_params = [poisson_fit.GetParameter(j) for j in range(2)]
    poisson_errors = [poisson_fit.GetParError(j) for j in range(2)]
    hist_list.append(hist)

    hist.Draw()
    gauss_fit.Draw("same")
    poisson_fit.Draw("same")

    stats_box = TPaveText(0.6, 0.6, 0.88, 0.83, "NDC")
    stats_box.SetFillColor(0)
    stats_box.SetTextFont(42)
    stats_box.SetTextSize(0.03)
    stats_box.SetBorderSize(1)

    stats_box.AddText("GaussianFit:")
    stats_box.AddText(f"Mean={gauss_params[1]:.3f}+-{gauss_errors[1]:.3f}")
    stats_box.AddText(f"Sigma={-gauss_params[2]:.3f}+-{gauss_errors[2]:.3f}")
    stats_box.AddText("PoissonFit:")
    stats_box.AddText(f"Lambda={poisson_params[1]:.3f}+-{poisson_errors[1]:.3f}")
    stats_box.Draw()

    legend = TLegend(0.55, 0.4, 0.85, 0.5)
    legend.AddEntry(gauss_fit, "GaussianFit", "l")
    legend.AddEntry(poisson_fit, "PoissonFit", "l")
    legend.SetBorderSize(0)
    legend.SetFillStyle(0)
    legend.SetTextSize(0.03)
    legend.Draw()

    c.Update()
```

Code 2: PyRoot code for drawing the histograms gaussian and poisson distributions

```python

gStyle.SetOptStat(1)   # No default statistics box
gStyle.SetOptFit(1111)    # No automatic fit results
gStyle.SetTitleFont(42, "xyz")
gStyle.SetLabelFont(42, "xyz")
gStyle.SetTitleSize(0.04, "xyz")
gStyle.SetLabelSize(0.03, "xyz")
gStyle.SetPadTopMargin(0.08)
gStyle.SetPadBottomMargin(0.12)
gStyle.SetPadLeftMargin(0.12)
gStyle.SetPadRightMargin(0.08)

c = TCanvas("", "", 1000, 600)

hist_list=[]
#adjust the for loop in the list [0,1]
for i in [1]:
    col1_data = L2[i]
    hist = TH1F(f"", f"n={i}", 9, min(col1_data), max(col1_data))
    hist.SetDirectory(0)
    hist.SetFillColorAlpha(4, 0.4)  # Blue fill with transparency
    hist.SetLineColor(1)
    hist.SetLineWidth(2)
    hist.SetFillStyle(1001)

    for value in col1_data:
        hist.Fill(value)

    if i == 0:

        func = TF1("n0", "[0]*[1]*exp(-[1]*x)", min(col1_data), max(col1_data))
        func.SetParameter(0, 1.0)
        func.SetParameter(1, alpha)
        bin_min = hist.FindBin(min(col1_data))
        bin_max = hist.FindBin(max(col1_data))
        integral1 = hist.Integral(bin_min, bin_max, "width")
        integral2 = func.Integral(min(col1_data), max(col1_data))
        scale = integral1 / integral2
        func.SetParameter(0, func.GetParameter(0) * scale),min(col1_data), max(col1_data)
        func.SetLineWidth(3)
        n0_fit = TF1(f"n0_fit_{i}", "[0]*[1]*exp(-[1]*x)", min(col1_data), max(col1_data))
        n0_fit.SetParameters(10,0.1)
        n0_fit.SetParName(0, "Amplitude")
        n0_fit.SetParName(1, "Alpha")
        n0_fit.SetLineColor(TColor.GetColor(0, 200, 0))  # Green
        n0_fit.SetLineWidth(3)
        hist.Fit(n0_fit, "SQ")
        fit=n0_fit
        n0_params = [n0_fit.GetParameter(j) for j in range(3)]
        n0_errors = [n0_fit.GetParError(j) for j in range(3)]
        hist_list.append(hist)
        hist.Draw()
        n0_fit.Draw("same")
        func.Draw("same")
    if i==1:
        func = TF1("n0", "[0]*x*([1]**2)*exp(-[1]*x)", min(col1_data), max(col1_data))
        func.SetParameter(0, 1.0)
        func.SetParameter(1, alpha)
        bin_min = hist.FindBin(min(col1_data))
        bin_max = hist.FindBin(max(col1_data))
        integral1 = hist.Integral(bin_min, bin_max, "width")
        integral2 = func.Integral(min(col1_data), max(col1_data))
        scale = integral1 / integral2
        func.SetParameter(0, func.GetParameter(0) * scale),min(col1_data), max(col1_data)
        func.SetLineWidth(3)
        n1_fit = TF1(f"n1_fit_{i}", "[0]*x*([1]**2)*exp(-[1]*x)", min(col1_data), max(
            col1_data))
```

```
        n1_fit.SetParameters(10,0.1)
        n1_fit.SetParName(0, "Amplitude")
        n1_fit.SetParName(1, "Alpha")
        n1_fit.SetLineColor(TColor.GetColor(0, 200, 0))  # Red
        n1_fit.SetLineWidth(3)
        hist.Fit(n1_fit, "SQ")
        fit=n1_fit
        n1_params = [n1_fit.GetParameter(j) for j in range(2)]
        n1_errors = [n1_fit.GetParError(j) for j in range(2)]
        hist_list.append(hist)

        hist.Draw()
        n1_fit.Draw("same")
        func.Draw("same")

    legend = TLegend(0.55, 0.4, 0.85, 0.5)
    legend.AddEntry(func, "TheoreticalDist.", "l")
    legend.AddEntry(fit, "PoissonFit", "l")
    legend.SetBorderSize(0)
    legend.SetFillStyle(0)
    legend.SetTextSize(0.03)
    legend.Draw()

    c.Update()
```

Code 3: PyRoot code for Part B

```

means=[192.4,19.03,22.44,2.414]
var=[13.14,4.167,4.661,1.484]
def func2(means,var):
    return [np.sqrt(means[i])/var[i] for i in range(len(means))]

x_values=means
y_values=func2(means,var)
plt.figure(figsize=(8, 6))
plt.scatter(means, func2(means,var), color='blue', label='$\sqrt{\mu}/\sigma$')

# Optional: Fit a line (like in the image)
fit = np.polyfit(x_values, y_values, 1)  # linear fit
fit_fn = np.poly1d(fit)

# Print slope and intercept
slope, intercept = fit
print(f"Slope:{slope:.6f}")
print(f"Intercept:{intercept:.6f}")
plt.plot(x_values, fit_fn(x_values), color='brown', label='LinearFit')
plt.axhline(y=1, linestyle=':', linewidth=1.5, label=r'expectedline(y=1)')
# Labels and Title
plt.title(r'$\sqrt{\mu}/\sigma$Graph')
plt.xlabel(r'$\mu$')
plt.ylabel(r'$\sqrt{\mu}/\sigma$')

# Axes limits and grid
plt.ylim(0.8, 1.2)
plt.xlim(0, 200)
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()

# Show
plt.show()
```

Code 4: Mean value and variance comparison plot in python