Poisson Statistics Experiment

Zekeriya Gökhan Evelek
Boğaziçi University • Physics Department

***Abstract*—The experiment studies the Poisson distribution, which can be used for any counting process. In this experiment, we compared it with Gaussian distribution, have tested the validity of the equation $\sigma^2 = \mu$ for Possion distribution, and used its modified version for counting processes. The experiment was in agreement with those theories and confirmed them.**

I. Theory

For a succession of events, if each trial has a finite probability of succes and independent of any other trial, then any result would have a probability which is equal to the product of the possibilities:

$$P = p^n.(1-p)^{m-n} \tag{1}$$

Where, m is the number of successes and n is the number of trials and p is the probability of success.

However, if we do not seek any order in that then the result can be represented by the so-called Binomial Distribution:

$$B(n;m,p) = \frac{m!}{m!(n-m)!} p^n (1-p)^{m-n} \tag{2}$$

Poisson Distribution was first introduced by Simeon Denis Poisson in 1837. His work was focused on the number of wrongful convictions in a given country in a time interval t, provided that this number N is discrete.[1] On the other hand, this discrete probability distribution is actually a special case of binomial distribution, in which number of trials goes to infinitely large while number of success is infinitesimally small(in the limit m-¿inf and p-¿0)[2]. Then, binomial distribution becomes:

$$P(\lambda, n) = \frac{\lambda^n}{n!} e^{-\lambda} \tag{3}$$

where λ is poison parameter and n can be any positive integer up to infinity.

Now, another special case of Binomial distribution is called Gaussian Distribution, which is considered as a very crucial one because almost all physical processes can be represented, or approximated by using Gaussian. Gaussian distribution is a special case of Binomial distribution where m becomes infinitely large[2]:

$$\lim_{m \to \inf} B(n;m,p) = G(x,\sigma,\mu) = \frac{1}{\sigma\sqrt{2\pi}} exp[-\frac{1}{2}(\frac{x-\mu}{\sigma})^2] \tag{4}$$

where $\sigma = \sqrt{mp(1-p)}$ is standard deviation and $\mu = mp$ is the mean of the distribution.

What is more, when we are to use it for calculating successive events, then Poisson distribution(eq.3) is going to be:

$$P_q(n+1,t) = \frac{(\alpha t)^n exp[-\alpha t)]}{n!} \tag{5}$$

This is for the probability of occurrence of n events in t time followed by another event in time dt, where $\alpha = \mu/t$

II. Method

We have used two different gamma-ray sources, Ba-133 and Cs-137(see fig. 2-2-3). After finding out the plateau region, we have recorded the number of gamma radiation emitted by those sources for 1-s and 10-s periods for each.

1) The operating voltage of the Geiger Counter is set to be 440V
2) Barium-133 is placed as gamma-ray source
3) The number of events is recorded for 1-s periods for 100 times
4) The number of events is recorded for 10-s periods for 100 times
5) Caesium-137 is placed as gamma-ray source
6) The number of events is recorded for 1-s periods for 100 times
7) The number of events is recorded for 10-s periods for 100 times
8) The number successive counts is recorded by chart recorder.
9) The distance between each count is measured.

III. The Experimental Setup

Here is our set-up:

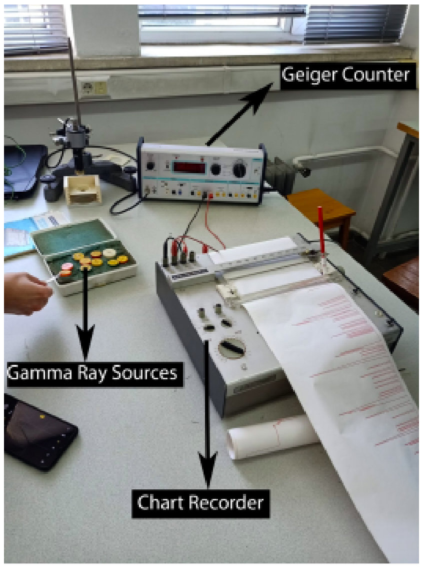

Fig. 1. Apparattus

Fig. 2. Barium-133

Fig. 3. Caesium-137

- Geiger Counter
- Gamma-Ray sources
- Chart recorder

IV. The Data

The values that we got by chart recorder(in millimeters): 24-4-11-8-19-7-8-22-5-3-20-15-7-9-10-9-1-11-10-4-21-11-10-18-14-5-20-2-11-8-9-2-16-9-23-19-4-11-7-6-4-3-10-4-9-5-34-14-7-4-6-9-15-25-4-6-17-2-13-14-15-4-12-10-7-10-9-14-9

TABLE I
The number of counts for each source with corresponding frequencies

Ba-133 1s	Ba-133 10s	Cs-137 1s	Cs-137 10s
7	20	14	124
1	21	8	102
3	32	19	109
0	21	14	113
3	17	10	95

TABLE II
The same table is continued

Ba-133 1s	Ba-133 10s	Cs-137 1s	Cs-137 10s
4	30	9	99
4	30	11	110
3	21	12	87
5	19	10	113
0	26	11	102
2	26	8	103
4	27	12	99
6	18	14	107
1	29	9	96
5	12	6	89
1	23	7	108
2	29	10	103
5	21	13	88
3	20	6	113
3	16	18	104
1	30	13	123
1	21	13	98
1	34	5	111
3	23	11	106
2	21	7	108
1	13	13	109
1	23	9	118
5	34	9	119
0	14	11	111
2	19	12	106
4	28	11	101
6	20	5	90
1	31	12	117
5	20	12	106
1	22	9	104
2	20	13	132
5	15	9	95
3	15	8	108
3	22	8	101
2	34	10	100
3	32	11	106
1	27	12	105
1	20	9	102
1	28	13	94
3	30	11	116
2	16	11	91
3	15	8	105
2	23	7	111
4	22	10	110
2	30	7	118
4	32	9	108
3	18	14	85
2	30	17	87
5	18	13	105
7	30	12	103
4	16	10	123
6	21	7	91
17	18	10	113
2	25	11	93
13	19	12	125
14	30	15	125
15	16	11	101
4	21	10	114
12	18	7	108
10	25	10	130
9	30	14	89
11	18	11	101
8	22	13	98

V. The Analysis

We have started measuring counts for 100-seconds intervals with 380 volts and then increased it 20 volts each time. When we took our fourth measurement, we found out that the results are almost the same, which means we are in the plateau region,

TABLE III
THE SAME TABLE IS TO BE CONTINUED

Ba-133 1s	Ba-133 10s	Cs-137 1s	Cs-137 10s
9	29	9	82
14	31	11	116
11	30	12	108
13	16	13	121
9	21	13	115
11	18	13	99
12	25	13	103
13	33	9	122
13	24	15	99
9	24	16	118
15	25	10	97
16	33	11	101
10	24	11	110
11	24	13	103
11	25	7	116
13	29	17	125
7	20	13	105
17	31	6	111
13	23	10	99
6	22	9	129
10	29	15	92
9	31	16	109
15	30	10	117

so we decided to continue the experiment with 440 volts for the Geiger Muller Tube.

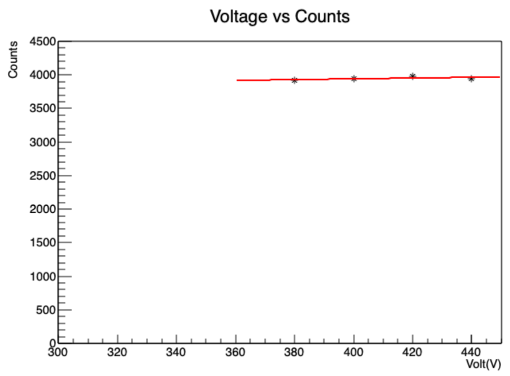

Fig. 4

Here is Gaussian and Poisson fits for four different datasets which includes 1 second and 10 seconds frequencies for Barium-133 and Caesium-137. Since the table cannot give both $\frac{\chi^2}{ndf}$ values for each fit in the same graph, I got the values from ROOT's built in function and then gave the final results below for each graph. We expect to see that Poisson fits converges to Gaussian fits as the mean is increased.

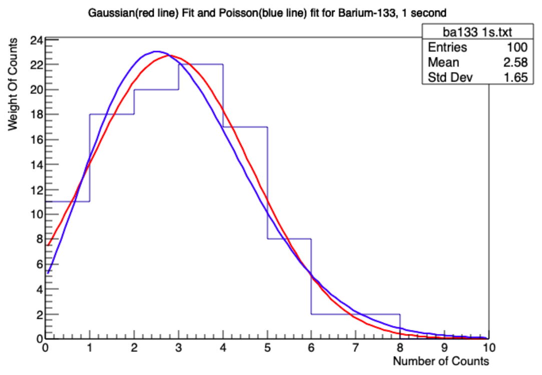

Fig. 5. for gaussian fit $\frac{\chi^2}{ndf}$ = 0.359341, and for poison fit $\frac{\chi^2}{ndf}$ = 0.494695

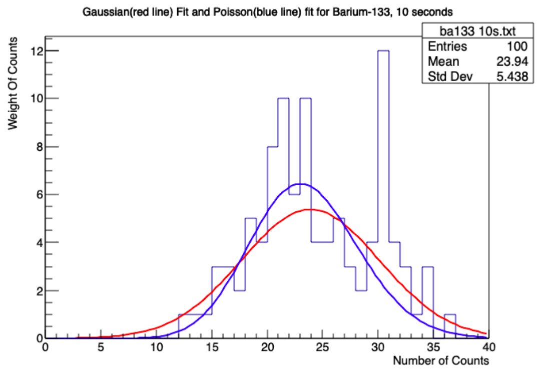

Fig. 6. for gaussian fit $\frac{\chi^2}{ndf}$ = 0.996615, and for poison fit $\frac{\chi^2}{ndf}$ = 1.042

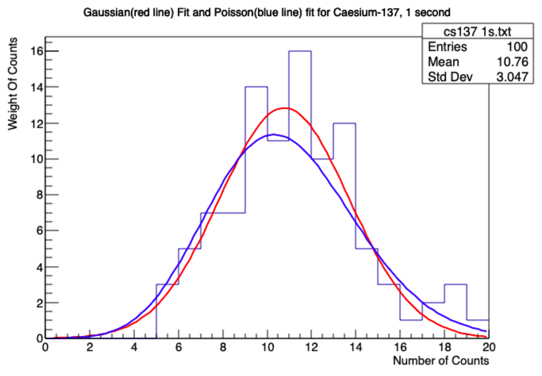

Fig. 7. for gaussian fit $\frac{\chi^2}{ndf}$ = 0.686221, and for poison fit $\frac{\chi^2}{ndf}$ = 0.750306

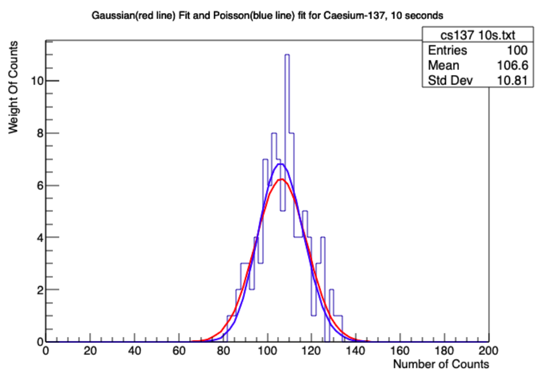

Fig. 8. for gaussian fit $\frac{\chi^2}{ndf}$ = 0.584526, and for poison fit $\frac{\chi^2}{ndf}$ = 0.589127

As expected, when the graph has lower means, $\frac{\chi^2}{ndf}$ values are slightly different. However, when the mean is increased due to the increased frequencies for measuring occurrences per 10 seconds, then $\frac{\chi^2}{ndf}$ values are really converged to each other, as we have expected from the theory that Poisson distribution converges to Gaussian distribution in the higher mean values.

I took the mean and standard deviation values for each graph, calculated by ROOT and graphed $\frac{\sqrt{\mu}}{\sigma}$ as a function of μ. The expectation is that line fit for the calculated values has 0 slope and it passes through the y-axis in 1.

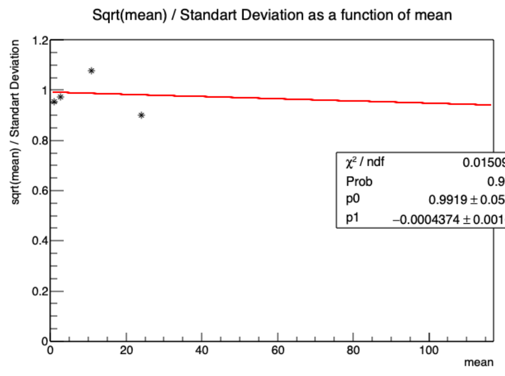

Fig. 9. The slope of the line is -0.0004 $\pm$ 0.001 and the y-intercept is 0.99 $\pm$ 0.05

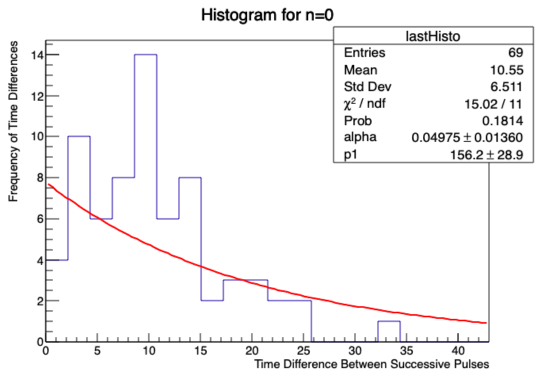

Fig. 10. The modified Poisson fit for n=0

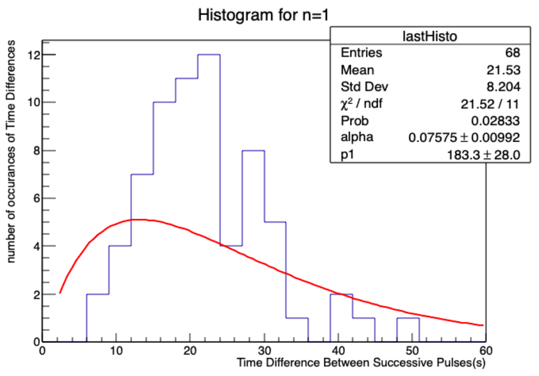

Fig. 11. The modified Poisson fit for n=1

I have calculated the theoretical value of α by dividing the total number of peaks to the total length which is: α = 0.094 1/s.

VI. The Result

We have shown that Poisson and Gaussian distributions converge to each other as the mean increases(see $\frac{\chi^2}{ndf}$ values for each graph from the table)

We also showed that $\frac{\sqrt{\mu}}{\sigma}$ as a function of μ has a slope of -0.0004 $\pm$ 0.001 while a y-intercept 0.99 $\pm$ 0.05. which means we got a horizontal line y $\sim$ 1.

Furthermore, we have calculated two experimental values for α by using a modified version of Poisson distribution for our fit, which is found to be 0.050$\pm$ 0.014 for n=0 and 0.076 $\pm$ 0.010 for n=1.

VII. The Conclusion

We have found out that Gaussian and Poisson fits to become closer as the mean of the distribution increases, as expected. $\frac{\chi^2}{ndf}$ values differs for our 1-s period measurements, while the values are almost the same when we recorded the events for the same gamma-ray sources for 10-s, which means higher mean values. Therefore, our Gaussian and Poisson fits agrees with the theory.

In order to test the claim that the mean of Poisson distribution is equal to the square of its standard deviation[2], we graphed $\frac{\sqrt{\mu}}{\sigma}$ as a function of μ and got an almost a horizontal line. The difference of the slope from 0, the expected value, seems to be acceptable in the limits of the precision of our experiment.

Finally, the experimental values obtained by applying the modified Poission distribution(eq. 5) for the data-sets of the chart recorder implies that we got a better result of reaching any α value with increasing n values. Our experimental α value was closer to the theoretical value for n=1 case so we concluded that Poisson distribution gives better results when we increase the number of events.

References

[1] *poissob wikipedi*. URL: https://en.wikipedia.org/wiki/Poisson_distribution (visited on 05/10/2024).

[2] E. Gülmez. *Advanced Physics Experiments*. 1st. Boğaziçi University Publications, 1999.

VIII. Appendix

I have used ROOT's built-in function to apply Gaussian, Poisson and Linear fits(see figures 4-11)

```
{

  std::vector<std::string> filenames = {
      "ba133 1s.txt",
      "ba133 10s.txt",
      "cs137 1s.txt",
      "cs137 10s.txt",

    };

  std::vector<std::string> titles = {
      "Barium-133, 1 second",
      "Barium-133, 10 seconds",
      "Caesium-137, 1 second",
      "Caesium-137, 10 seconds"

    };

  std::vector<std::vector<double>> histoRanges
     = {
   {0,10},{0,40},{0,20},{0,200}
  };
  std::vector<std::vector<double>> fitRanges =
     {
   {0.0,8},{0.0,30},{0.0,15},{0.0,120}
  };
  float histogramBins [4] = {10,40,20,100};

  std::vector<float> means;
  std::vector<float> sigmas;
  float kai2Gauss[4];
  float kai2Poisson[4];



  for (int i = 0; i < filenames.size(); ++i) {

    std::string gaussianTitle = "Gaussian(red
        line) Fit and Poisson(blue line) fit
        for " + titles[i];
    std::string poissonTitle = "Poisson Fit for
        " + titles[i];


    TH1F *histoGauss = new
        TH1F(filenames[i].c_str(),gaussianTitle.c_str(),histogramBins[i],
        histoRanges[i][0], histoRanges[i][1]);
    histoGauss->GetXaxis()->SetTitle("Number of
        Counts");
    histoGauss->GetYaxis()->SetTitle("Weight Of
        Counts");
    TH1F *histoPoisson = new
        TH1F(filenames[i].c_str(),poissonTitle.c_str(),histogramBins[i],
        histoRanges[i][0], histoRanges[i][1]);
    histoPoisson->GetXaxis()->SetTitle("Number
        of Counts");
    histoPoisson->GetYaxis()->SetTitle("Weight
        Of Counts");

    std::ifstream infile(filenames[i]);
    float value;

    while (infile >> value) {
        histoGauss->Fill(value);
        histoPoisson->Fill(value);
    }
    means.push_back(histoGauss->GetMean());
    sigmas.push_back(histoGauss->GetStdDev());


    //gStyle->SetOptFit(1111);

    TF1 *gauss = new TF1("gauss","gaus");
    TF1 *poisson = new
        TF1("poisson","[0]*TMath::Poisson(x,[1])",0,histo
    //gauss->SetParNames("Gauss_Constant","Gauss_Mean","G
    poisson->SetParNames("Poisson_Scale","Poisson_Mean");

    gauss->SetParameters(histoGauss->GetMaximum(),histoGa
        histoGauss->GetRMS() );
    histoGauss->Fit("gauss","S",0,fitRanges[i][1]);
    poisson->SetParameters(histoPoisson->GetMaximum(),his
    poisson->SetLineColor(kBlue);
    histoGauss->Fit("poisson","S+");

    float kaigauss = gauss->GetChisquare();
    float ndfgauss = gauss->GetNDF();
    float kaipoisson = poisson->GetChisquare();
    float ndfpoisson = poisson->GetNDF();
    kai2Poisson[i] = kaipoisson/ndfpoisson;
    kai2Gauss[i] = kaigauss/ndfgauss;

    cout<< "kai2/ndf value for gaussian fit
        for"<< titles[i].c_str()<<"is
        "<<kai2Gauss[i]<<endl;
    cout<< "kai2/ndf value for poisson fit
        for"<< titles[i].c_str()<<"is
        "<<kai2Poisson[i]<<endl;



    /*poisson->SetParameters(histoPoisson->GetMaximum(),h
    histoPoisson->Fit("poisson", "R");
 */
    TCanvas *c1 = new TCanvas();
    histoGauss->Draw();
    /*TCanvas *c2 = new TCanvas();
    histoPoisson->Draw();
   */
    }

    /*for(int j = 0; j < 4; j++) {
       cout<< "means "<<means[j]<<"sigmas
           "<<sigmas[j]<< endl;
    }
 */
    float x[4], y[4];
    for (int j = 0; j < 4; j++) {
       x[j] = means[j];
       float yValue = sqrt(means[j]) /
           sigmas[j];
       y[j]= yValue;
    }
    TCanvas *c3 = new TCanvas();
    TGraphErrors secondGraph(4, x, y);
    secondGraph.SetTitle("Sqrt(mean) / Standart
        Deviation as a function of mean");
    secondGraph.GetXaxis()->SetTitle("mean");
    secondGraph.GetYaxis()->SetTitle("sqrt(mean)
        / Standart Deviation");
    secondGraph.SetMinimum(0);
    secondGraph.SetMaximum(1.2);
TF1 *fitFunction = new TF1("fitFunction",
    "[0] + [1]*x",0,120);
secondGraph.Fit("fitFunction", "R");
secondGraph.Draw("A*");

fitFunction->Draw("same");

// PART 2

TF1 *n0 = new TF1("n0",
    "[1]*[0]*TMath::Exp(-[0]*x)");
TF1 *n1 = new TF1("n1",
    "[1]*[0]*[0]*x*TMath::Exp(-[0]*x)",2,60);
n0->SetParName(0, "alpha");
n1->SetParName(0, "alpha");

TCanvas *c4 = new TCanvas();



TH1F *lastHisto = new
    TH1F("lastHisto","n=0",20, 0.0, 43.0);
lastHisto->GetXaxis()->SetTitle("Time
    Difference Between Successive
    Pulses(s)");
lastHisto->GetYaxis()->SetTitle("number of
    occurances of Time Differences");
lastHisto->SetTitle("Histogram for n=0");



std::vector<std::string> filenames2 = {
   "occurance.txt",

 };
std::ifstream infile2(filenames2[0]);
float value2;
int number_of_events;
float total_time;
while (infile2 >> value2) {
     lastHisto->Fill(value2);
     number_of_events += 1;
     total_time += value2;
}

float theoreticalAlpha =
    number_of_events/total_time;

n0->SetParameters(lastHisto->GetMean(),lastHisto->GetMaximum());
n0->SetRange(0.1 ,40);
lastHisto->Fit("n0");
lastHisto->Draw();

TCanvas *c5 = new TCanvas();
TH1F *lastHisto2 = new
    TH1F("lastHisto","n=0",20, 0.0, 60.0);
lastHisto2->GetXaxis()->SetTitle("Time
    Difference Between Successive
    Pulses(s)");
lastHisto2->GetYaxis()->SetTitle("number of
    occurances of Time Differences");
lastHisto2->SetTitle("Histogram for n=1");


std::vector<std::string> filenames3 = {
   "occurances2.txt",

 };
    std::ifstream infile3(filenames3[0]);
    float value3;
    while (infile3 >> value3) {
         lastHisto2->Fill(value3);
    }

    n1->SetParameters(0.1,120);
    lastHisto2->Fit("n1","R");
    lastHisto2->Draw();
    gStyle->SetOptFit(1111);

    cout << "theoretical value of alpha
        calculated is "<<theoreticalAlpha<<endl;


}
```

This line of code is for the graph of voltages vs number of counts values with which we decided the operating voltage

```
   {
float operatingVoltages[4] = {380,400,420,440};
  float firstCounts[4] = {3922,3941,3982,3947};



  TGraphErrors *operating = new
     TGraphErrors(4,operatingVoltages,firstCounts,0,0);
  TCanvas *c1 = new TCanvas();

  operating->Draw("A*");
  operating->SetTitle("Voltage vs
     Counts;Volt(V);Counts");
  operating->GetXaxis()->SetLimits(300, 450);
  operating->SetMinimum(0);
  operating->SetMaximum(4500);

  TF1 *fnew = new
     TF1("fnew","[0]*x+[1]",360,500);
  fnew->SetParameters(0.01,1300);
  operating->Fit(fnew,"R");

}
```