AST325 Lab 1: Basic Photon Statistics

Juan Pablo Alfonzo
juanpablo.alfonzo@mail.utoronto.ca

October 2, 2019

Abstract

This report focuses on the statistical properties of photons that were obtained using a Charged Coupled Device (CCD) image sensor, which relies on the photoelectric effect. Various data sets were collected from the CCD results and basic statistics such as mean and standard deviation are calculated for each data set. Then some of the data sets were compared to Gaussian and Poisson distributions using Python code. Measurement error analysis was done on one of the collected data sets. There where two main conclusions from the experiment. Firstly, that taking weighted means and weighted standard deviations are a good way to account for measurement errors. Secondly, that photon count rates follow a Poisson distribution when the mean is small, but follow a Gaussian distribution when the mean is large.

1 Introduction

This lab uses the photoelectric effect to measure photons from distant astronomical sources. The photoelectric effect occurs when a photon strikes a metal surface releasing an electron. The first data set in this lab contains the measured distance to a nearby star that was obtained via observing photon count rates using a CCD.The first data set is analyzed to calculate its mean and standard deviation. A scatter plot and histogram of the data is also presented. Measurement error reduction was also done on this data set and will be discussed.

The other three data sets in this lab consist of the numbers of photons that struck the CCD sensor in a given time. The data sets differ by sample sizes and by integration times. The photon count rate data sets are analyzed through the calculation of mean and standard deviation, additionally scatter plots and histograms of the data sets are presented and compared to Poisson and Gaussian distributions.

Work for this experiment was evenly split amongst the author (Juan Pablo Alfonzo) and the three other group members: Parampreet Singh, Lucas Louwerse, and Nicholas Clark. The code used to analyze the data sets were discussed and worked on together as a group, but each member wrote their own unique code and had their own unique approach to handling the data.

2 Observations and Data

All four data sets in this experiment were collected by a CCD image sensor that converts incoming photons into electrons using the photoelectric effect. This electron is then sent through a chain of sensors that end with an analog to digital converter so that a computer can record the results. The specifics of this process can be found in the reference section of this paper.[1]

Since the main purpose of this lab was to do statistical analysis on photon count rates, 3 of the 4 collected data sets consist of number of photons recorded by the CCD in a given integration time. The 4 data sets have been put into tables bellow, shortened for brevity sakes as some data sets contained as many as 1000 elements. The original data sets can be found in the reference section of this paper.[1][2] No actual measurements were made by the author or group members as all the data sets were given on the course website.

Measurement Number	Distance (pc)	Measurement Error
1	38.91	1.41
2	37.14	0.36
...	...	...
29	33.76	3.74
30	36.51	0.99

Table 1: Data Set of Distance to a Nearby Star with Measurement Error [1]

Measurement Number	Photon Count Rate
1	13
2	17
...	...
29	12
30	10

Table 2: Data Set of Photon Count Rate with Integration Time of 1s [1]

Measurement Number	Photon Count Rate
1	2
2	7
...	...
999	2
1000	3

Table 3: Data Set of Photon Count Rate with *small* Integration Time [2]

Measurement Number	Photon Count Rate
1	23
2	28
...	...
999	33
1000	22

Table 4: Data Set of Photon Count Rate with *large* Integration Time [2]

3 Data Reduction and Methods

The data sets collected where then imported into Python as list so they could be statistically analyzed. The `pandas` module was used on Python in order to import data from an excel spreadsheet for the data sets shown in tables 3 and 4. The `numpy` module on Python was used to get the value of π and e. For the first data set dealing with the distance to a nearby star the mean (1) and standard deviation (2) of the data is computed using the following equations:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

$$\sigma = \sqrt{\frac{1}{N}\Big(\sum_{i=1}^{N} x_i - \bar{x}\Big)^2} \tag{2}$$

Where N is the number of trials and the x_i's are the elements in the data set

This data set also contains measurement errors that were accounted for by calculating a weighted mean (3) and weighted standard deviation (4) to get an uncertainty as follows:

$$\bar{x_w} = \frac{\sum_{i=1}^{N}(x_i/w_i)}{\sum_{i=1}^{N}(1/w_i)} \tag{3}$$

$$\sigma_w = \frac{1}{\sqrt{\sum_{i=1}^{N}(1/w_i^2)}} \tag{4}$$

Where w_i's are the respective measurement errors of the x_i's

For the other 3 data sets dealing with photon count rates the mean (1) and standard deviation (2) are taken and compared to Gaussian and Poison distributions. Gaussian distribution is given by:

$$G = \frac{1}{\sqrt{2\pi}\sigma} exp\Big[\frac{-1}{2}\Big(\frac{x-\mu}{\sigma}\Big)^2\Big] \tag{5}$$

Where G is the probability x events will be measured within an interval of μ events

Poison distribution is given by:

$$P = \frac{\mu^x}{x!} e^{-\mu} \tag{6}$$

Where P is the probability x events will be measured within an interval of μ events

4 Data Analysis and Models

Python code made by the author is used to calculated mean and standard deviation of all four data sets. The code used for this calculation can be found in appendix A. For the first data set whose weighted mean and weighted standard deviation the code for these computations can be found in appendix B. The data sets were also plotted into scatter plots and histograms using the `matplotlib` module on Python. Gaussian and Poisson distubrutions curvers were also plotted using `matplotlib`

4.1 Distance to Nearby Star Data Set

The data set from table 1 is plotted into the scatter plot and histogram bellow using the Python code found in Appendix C.

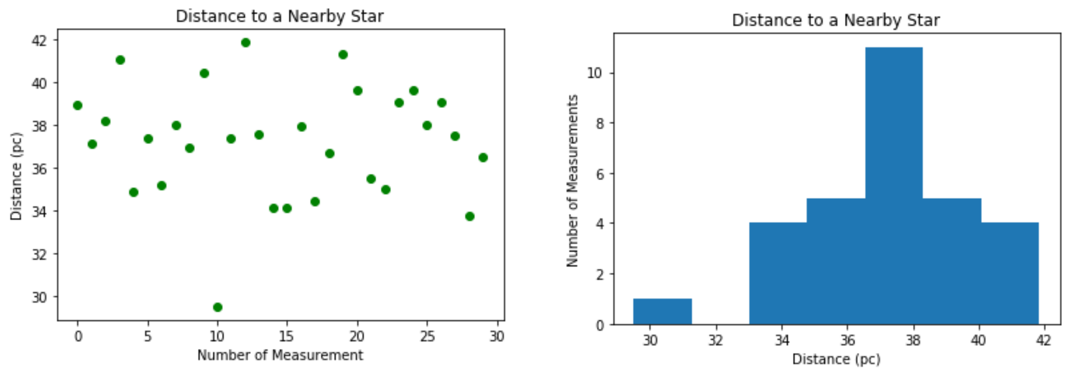

Figure 1: Scatter Plot and Histogram for Distance to a Nearby Star

4.2 Photon Count Rate with Integration Time of 1s

The data set from table 2 is plotted onto a histogram, using Python code identical to the one in appendix C with changes being made to the data set it uses and titles of the graph, and a Poisson distribution curve is plotted over top. The Python code for the Poisson distribution curve can be found in appendix D. Note that the Poisson distribution has a scale factor that can be found in the Python code. This was done to have a better comparison between the data and the Poisson distribution.

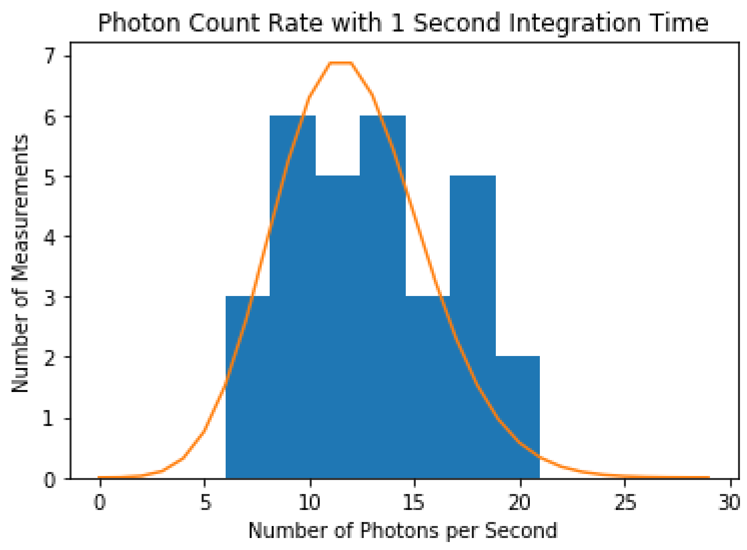

Figure 2: Photon Count Rate with 1s Integration Time compared to Poisson Distribution

4.3 Photon Count Rate with Small Integration Time

The data set from table 3 is plotted onto a scatter plot and histogram, using Python code identical to the one in appendix C with changes being made to the data set it uses and titles of the graph. The histogram has a Poisson distribution curve over top that was generated with identical code found in appendix D with the corresponding mean being used instead. Note that the Poisson distribution has a scale factor that can be found in the Python code. This was done to have a better comparison between the data and the Poisson distribution.

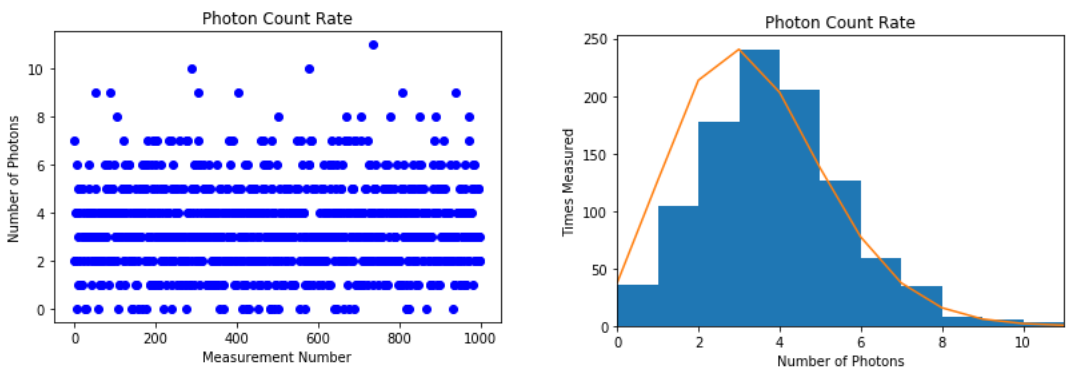

Figure 3: Scatter Plot and Histogram of Photon Count Rate using a Small Integration Time compared to a Poisson Distribution Curve

4.4 Photon Count Rate with Large Integration Time

The data set from table 4 is plotted onto a scatter plot and histogram, using Python code identical to the one in appendix C with changes being made to the data set it uses and titles of the graph. The histogram has a Poisson distribution curve over top that was generated with identical code found in appendix D with the corresponding mean being used instead. A Gaussian distribution is also plotted over top of the histogram using the Python code found in Appendix E. Note that the Gaussian distribution has a scale factor that can be found in the Python code. This was done to have a better comparison between the data and the Gaussian distribution.

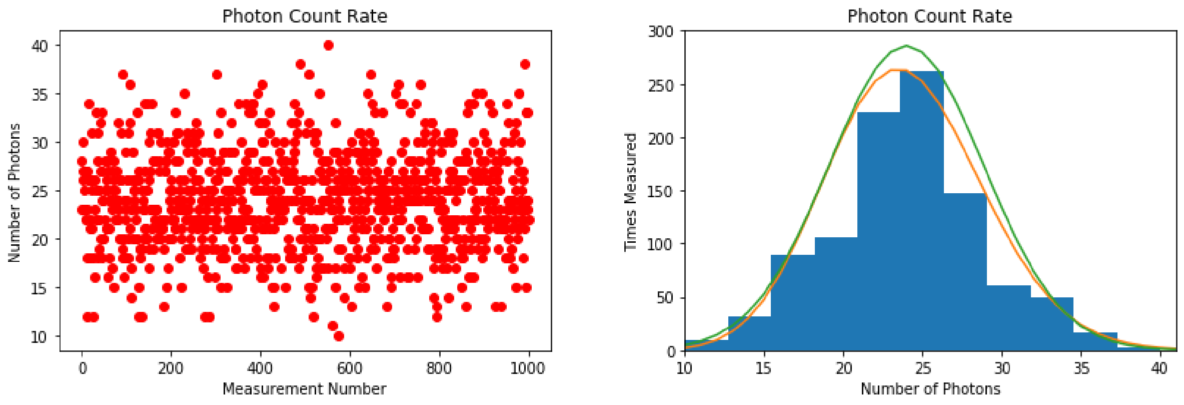

Figure 4: Scatter Plot and Histogram of Photon Count Rate using a Large Integration Time compared to a Poisson (Orange) and Gaussian (Green) Distribution Curve

5 Discussion

5.1 Distance to Nearby Star Data Set

The mean of this data set was calculated to be 37.20 and the standard deviation to be 2.65. The weighted mean was calculated to be 37.49 and the weighted standard deviation was calculated to be 0.02. The mean and weighted mean are quite close to one another, but the standard deviation and weighted standard deviation differ significantly. Looking at Figure 1, especially the histogram, it is clear that the weighted standard deviation is a more accurate representation of the data than the standard deviation.

5.2 Photon Count Rate with Integration Time of 1s

The mean of this data set was calculated to be 13.2 and the standard deviation was calculated to be 3.8. Figure 2 shows that this data set follows a Poisson distribution, although not very strongly as it does not closely follow the trend for Poisson distribution that states that the standard deviation squared is equal to the mean.

5.3 Photon Count Rate with Small Integration Time

The mean of this data set was calculated to be 3.38 and the standard deviation to be 1.78. The histogram of figure 3 shows that this data set closely follows a Poisson distributions since it follows the well known trend for Poisson distribution that the standard deviation squared is equal to the mean.

5.4 Photon Count Rate with Large Integration Time

The mean of this data set was calculated to be 23.98 and the standard deviation to be 4.88. The histogram of figure 4 shows that this data set more closely follows a Gaussian distribution rather than a Poisson one, and that furthermore given the mean of this data the Poisson distribution is quite similar to the Gaussian one.

6 Conclusions

6.1 Distance to Nearby Star Data Set

From our results what we can conclude is that using weighted means and weighted standard deviations is a good way to account for measurement errors. This is clear when we compare the weighted standard deviation to the standard deviation of the data set. The weighted one is more accurate when we compare it to the histogram of our data (figure 1) than the non-weighted one. The weighted standard deviation manages to take the measurement errors and account for the noise that is intrinsic to them. Hence, taking the weighted standard deviation allows us to get a more precise reading of our data which makes it a good method of dealing with measurement errors.

6.2 Photon Count Rate Data Sets

From the histogram of figure 3 we see that for a photon count rate with a small mean (3.38) the distribution of the data set is very similar to that of Poisson distribution. We see from figure 2 that for a larger mean (13.2) that the data set still follows a Poisson distribution but much more weakly than we see with a smaller mean. Finally we see from the histogram in figure 4 that for a big mean (23.98) the data set follows a Gaussian distribution more strongly than a Poisson one. This is exactly was is expected as the larger the mean the more a Poisson distribution begins to resemble a Gaussian one.

7 Appendix

7.1 Appendix A

Python code for mean and standard deviation calculation where measurement data sets were changed accordingly:

```
import numpy as np
import matplotlib as plt
# Take the sum of all the elements of inside the measurement result set (X) and divide
by number of elements, in this case 30
```

```
    M=sum(X)/30
    #Show the mean
    print(”The mean is:”, M)
    #Formula for standard deviation
    #Create an index i to be the number of the element in the set i=0
    #Create an empty list to fill with the outputs of the while loops
    S=[ ]
    #Create a while loop that will calculate standard deviations. Setting i < 30 as the set
contains 30 elements
    while i < 30:
        #Find the difference between the element in the the measurement set and the mean
        S.append((X[i]-M)**2)
        i = i + 1
    #Calculate the standard deviation by taking the sum of the output set of the while loop
and divide this by N-1, in this case 29, and then take the square root
    SD= (sum(S)/29)**(1/2)
    #show the standard deviation
    print (”The Standard Deviation is:”, SD)
```

7.2 Appendix B

Python code for weighted mean and weighted standard deviation calculation:

```
    import numpy as np
    import matplotlib as plt
    #create a list called Y that contains all measurement errors
    Y=[1.41, 0.36, 0.69, 3.53, 2.64, 0.17, 2.34, 0.46, 0.57, 2.91, 8.00, 0.17, 4.34, 0.03, 3.38,
3.39, 0.44, 3.07, 0.82, 3.81, 2.11, 2.02, 2.52, 1.55, 2.12, 0.46, 1.52, 0.03, 3.74, 0.99]
    #Calculate weighted mean using while loop
    #Use j as an index
    j=0
    #Empty set to put outputs of numerator of weighted mean equation
    N=[ ]
    #Empty set to put outputs of denominator of weighted mean
    D=[ ]
    while j<30:
        #Numerator of weighted mean equation and put into list N
        N.append(X[j]/(Y[j])**2)
        #Denominator of weighted mean equation and put into list D
        D.append(1/Y[j]**2)
       j=j+1
    #Take sum of N list and D list and divide to find weighted average
    WM=(sum(N)/sum(D))
    print(”This is the Weighted Mean:”, WM)
```

```
#Calculate weighted standard deviation using formula
WS=(1/sum(D))**(1/2)
print(”This is the Weighted Standard Deviation:”, WS)
```

7.3 Appendix C

Python code to plot the data into a scatter plot and histogram

```
import numpy as np
import matplotlib as plt
#Scatter plot of Distance(pc) vs Measurement
plt.subplot()
plt.plot(X, ’go’)
#X axis title
plt.xlabel(’Number of Measurement’)
#Y axis title
plt.ylabel(’Distance (pc)’) #Title of graph
plt.title(’Distance to a Nearby Star’)
#Print graph
plt.show()

#Histrogram of number of measurements vs Distance (pc)
plt.subplot()
plt.hist(X, bins=7)
#X axis title
plt.xlabel(’Distance (pc)’)
#Y axis title
plt.ylabel(’Number of Measurements’)
#Title of graph
plt.title(’Distance to a Nearby Star’)
#Print Histogram
plt.show()
```

7.4 Appendix D

Python code that creates and plots Poisson Distribution

```
import numpy as np
import matplotlib as plt
#Create empty list for output of Poisson function
R=[ ]
#create new index l
l=0
#create while loop to model the Poisson function
while l < 30:
    #resets base value of factorial
    fact = 1
```

```
        #define factorial function using for loop
        for j in range(1, l+1):
        fact = fact*j
    #Add calculated values to empty set R, scaled by 5 to better suit the data, set mean to
12 as in figure
    R.append(5*((12**(l+1)/fact)*np.exp(-(12))))
    l=l+1
    #create the Poisson distribution using list R
    plt.plot(R)
```

7.5 Appendix E

Python code that creates and plots Gaussian Distribution

```
    import numpy as np
    import matplotlib as plt
    #create an empty set G to put outputs of the Gaussian function
    G=[ ]
    g=0
    #create a while loop to create Gaussian function
    while g < 100:
        #Gaussian function output being put into empty list G
        G.append(3500*(1/((2*np.pi*(ld)**2)**0.5)*np.exp(-1*((g-lm)**2)/(2*(ld)**2))))
        g=g+1
    #Plotting Gaussian Function plt.plot(G)
    #print plot
    plt.show()
```

8 References

[1]Dae-Sik Moon,*Basic Photon Statistics.* http://www.astro.utoronto.ca/ astrolab/files/Lab1_AST325-326_2019_Fall.pdf

[2]AST325/326 Website, *Group G Small and Large Data.*
http://www.astro.utoronto.ca/ astrolab/