Geiger Counter

Jason Pruitt

November 2019

1 Analysis

For both high and low counts, the readings from the NIM crate were plotted in a histogram to display the frequency of each count as well as to test the randomness of the data. Since a fraction of the particles were missed due to the dead time, we applied a correction, given by

$$N_{true} = \frac{N_{obs}}{1 - R_{obs}}, \tag{1}$$

where N_{obs} were the readings directly from the NIM crate, and R_{obs} was the counting rate. $\overline{N}$ was simply the mean or average of the counts, but the uncertainty in $\overline{N}$ was given by

$$\delta\overline{N} = \frac{\sigma}{m}, \tag{2}$$

or the SDOM, where σ is the standard deviation of the counts, and m is the number of counts. For high counts shown in Figure 1, a Gaussian theory curve, given by

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{(-(x-\overline{x})^2)}{2\sigma^2}}, \tag{3}$$

where x was the number of counts, $\overline{x}$ was the mean of the counts, and σ was again the standard deviation of the counts, was superimposed over the histogram to compare with the data. For low counts shown in Figure 2, a Poissonian theory curve, given by

$$P(\mu) = e^{-\mu}\frac{\mu^{\nu}}{\nu!}, \tag{4}$$

where μ was the number of counts and ν was the mean of the counts. But merely overlaying the curves upon the respective histograms revealed nothing about the goodness of fit of the curves to the data. A Chi-Squared analysis,

$$\chi^2 = \Sigma \frac{(y_k - y_{th})^2}{\delta y_k^2}. \tag{5}$$

was employed to compute how well the high data was approximated by the Gaussian, and how well the low data was by the Poissonian. Both χ^2 values, shown in Table 2, were less than the number of bins, which indicated that the theory curves accurately modeled the distributions. Along with this method, writing a comparison script in python allowed us to find the fraction of values within 1σ and 2σ, shown in Table 2. For the high counts, 69 percent of the values fell within 1σ of the mean of the high counts, well in agreement with the accepted value of 68 percent for a Gaussian distribution. 94 percent of the high counts fell within 2σ of the mean, in moderate agreement with the accepted value of 98 percent. For the low counts, 75 percent of the values fell within 1σ of the low counts' mean, while 97 percent fell within 2σ, quite close to the accepted value of 98. Lastly, we utilized a third method to verify that the theory distributions modeled the data taken. The reliability factor,

$$R = \frac{\sigma_{exp}}{\sigma_{theory}}, \tag{6}$$

where σ_{exp} was the standard deviation of the counts and σ_{theory} was the square root of the counts' mean provided a ratio to compare our two different σ. The reliability factor values, shown in Table 2, both fairly close to 1, provide further evidence that our theoretical predictions accurately model the data.

	$\overline{N}$	σ_{theory}	σ_{exp}
High:	3126± 5	55.91	47.11
Low:	3.60± 0.18	1.89	1.84

Table 1: Measured Values and related statistics

	χ^2	Fraction within 1σ	Fraction within 2σ	Reliability Factor
High:	11.78	0.69	0.94	0.84
Low:	6.99	0.75	0.97	0.97

Table 2: Values relating to the validity of the theory curves

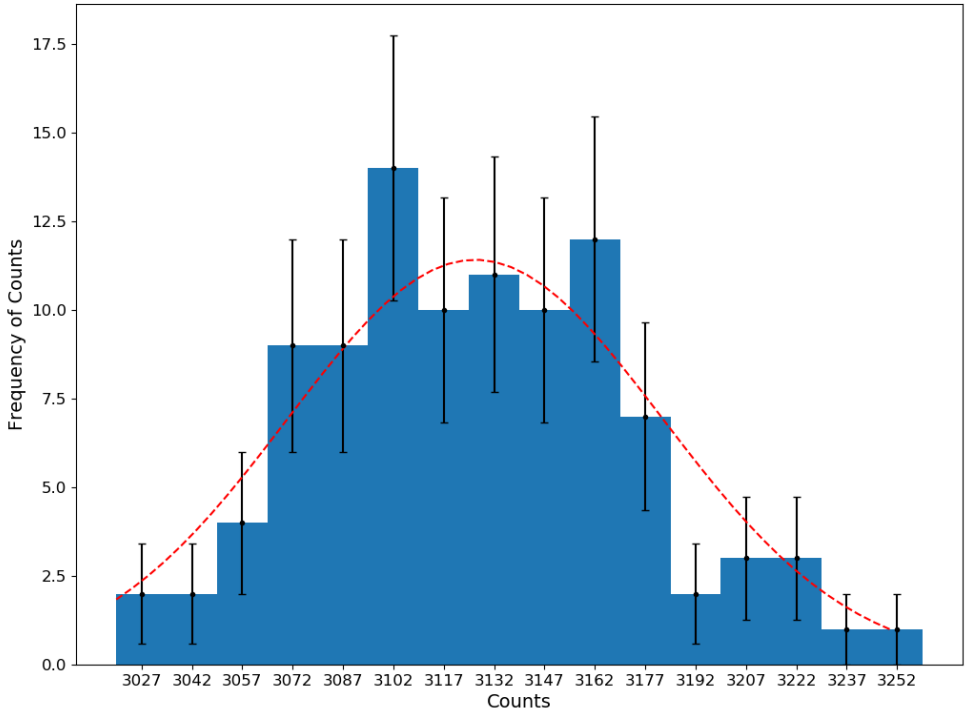

Figure 1: Histogram of high counts, with Gaussian theory curve superimposed

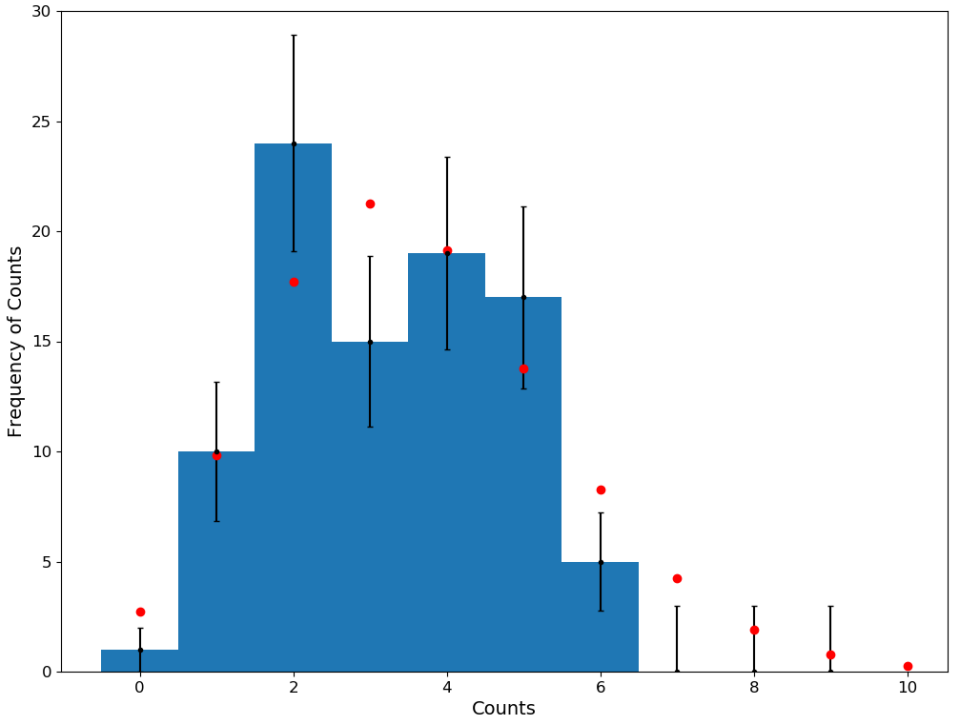

Figure 2: Histogram of low counts, with Poissonian theory values superimposed

2 Appendix

Appendix: Code

```
import math
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.stats import chisquare
from scipy.optimize import curve_fit
from scipy.special import factorial
from matplotlib.ticker import FormatStrFormatter

deadTime = 606e-6
pulseWidth = 32.8e-6
recoveryTime = 1.41e-3
hiRate = 10
loRate = .1
bgRate = 100

## Correction
def nTrue(nObs,rObs):
        nObs = np.array(nObs)
        C = 1/(1-rObs*deadTime)
        return C*nObs

## Read data from file
with open('GeigerHighLow.csv','r') as gc:
        header = gc.readline()
        line = gc.readline()
        high = []
        low = []
        while line:
                line = line.split(',')
                high.append(float(line[0]))
                low.append(float(line[1]))
                line = gc.readline()

high_corr = nTrue(high,hiRate)
low_corr = nTrue(low,loRate)
high_sorted = np.sort(high_corr)
low_sorted = np.sort(low)

def gauss(x,xbar):
        sig = np.sqrt(xbar)
        g = (1/(np.sqrt(2*np.pi)*sig))*np.exp((-(x-xbar)**2)/(2*sig**2)
```

```
            )
        return(g)

gaussian = lambda x,xbar : ((1/(np.sqrt(2*np.pi)*np.sqrt(xbar)))
                                                    * np.exp((-(x-
                                                        xbar)**2)
                                                        /(2*np.sqrt(
                                                        xbar)**2)))

def poisson(x,xbar):
        A = np.exp(-xbar)
        B = []
        for xVal in x:
                xVal = int(xVal)
                B.append((xbar**xVal)/math.factorial(xVal))
        B = np.array(B)
        return(A*B)

# Start fig
fig1,ax1 = plt.subplots(figsize = (12,9))

# Histogram and errorbar
n_bin1 = np.arange(high_sorted[0],high_sorted[-1]+15,15)
binspace1 = n_bin1[2] - n_bin1[1]
mid1 = n_bin1 + (binspace1/2)
mid1 = mid1[:-1]
hi_counts,hi_entries,hi_patches = ax1.hist(high_corr,bins = n_bin1,
    align = 'mid')
ax1.errorbar(mid1,hi_counts,yerr=np.sqrt(hi_counts),capsize = 3,fmt = '
    k.')

# Gaussian Fit
xtheory1 = np.linspace(high_sorted[0],high_sorted[-1])
gaussCorr = len(mid1)*len(high)*gauss(xtheory1,np.mean(high_corr))
ax1.plot(xtheory1,gaussCorr,'r--')

# Chi Squared
chiG = 0
for index, count in enumerate(mid1):
        g = (len(mid1)*len(high)*gaussian(count,np.mean(high_corr)))
        chiG = chiG + (((hi_counts[index] - g)**2)/hi_counts[index])

# Plot formatting
ax1.set_xlabel('Counts',fontsize = 14)
ax1.set_ylabel('Frequency of Counts',fontsize = 14)
ax1.set_xticks(mid1)
```

```
ax1.xaxis.set_major_formatter(FormatStrFormatter('%1.0f'))
ax1.tick_params(labelsize = 12)

fig1.savefig('GaussHist.png')
# plt.show()
plt.close()

# Start fig
fig2,ax2 = plt.subplots(figsize = (12,9))

# Histogram
n_bin2 = np.arange(low_sorted[0],low_sorted[-1])
binspace2 = n_bin2[1] - n_bin2[0]
mid2 = n_bin2 + binspace2/2

lo_counts,lo_entries,lo_patches = ax2.hist(low,bins = np.arange
    (-0.5,7.5),range = (-.05,7.5),align = 'mid')
ax2.errorbar(mid2[:-1]-.5,lo_counts,yerr = np.sqrt(lo_counts),capsize =
     2,fmt ='k.')
ax2.errorbar([7,8,9],[0,0,0],yerr = 3,capsize = 2,fmt = 'k.')

# Poisson Fit
xtheory2 = np.linspace(0,10,11)
poissonCorr = 100*1*poisson(xtheory2,np.mean(low))
ax2.plot(xtheory2,poissonCorr,'ro')

# Chi Squared
chiP = 0
for index,count in enumerate(mid2[:-1]):
        p = (100*1*poisson([count],np.mean(low)))
        if lo_counts[index] >= 5:
                chiP = chiP + (((lo_counts[index] - p)**2)/lo_counts[
                    index])
chiP = float(chiP[0])


# ax2.set_xticks(mid2)
ax2.set_xlabel('Counts',fontsize = 14)
ax2.set_ylabel('Frequency of Counts',fontsize = 14)
ax2.xaxis.set_major_formatter(FormatStrFormatter('%1.0f'))
ax2.tick_params(labelsize = 12)
ax2.set_ylim(0,30)
plt.savefig('PoissonHist.png')
# plt.show()
plt.close()
```

```
hiBar = np.mean(high_corr)
hiSig = np.std(high_corr)
hi_sigTheory = np.sqrt(hiBar)
hiBar_unc = hiSig/np.sqrt(len(high_corr))
hi_rel = hiSig/hi_sigTheory

loBar = np.mean(low_corr)
loSig = np.std(low_corr)
lo_sigTheory = np.sqrt(loBar)
loBar_unc = loSig/np.sqrt(len(low_corr))
lo_rel = loSig/lo_sigTheory


## Comparison
hi_oneSig = len([x for x in high_corr
if hiBar - hiSig <= x <= hiBar + hiSig])/len(high_corr)
hi_twoSig = len([x for x in high_corr
if hiBar - 2*hiSig <= x <= hiBar + 2*hiSig])/len(high_corr)

lo_oneSig = len([x for x in low_corr
if loBar - loSig <= x <= loBar + loSig])/len(low_corr)
lo_twoSig = len([x for x in low_corr
if loBar - 2*loSig <= x <= loBar + 2*loSig])/len(low_corr)

print('\nGaussian chi-Squared:',chiG)
print(f'N Bar High: \t {hiBar}')
print(f'{hi_oneSig} of the high values are within one sigma')
print(f'{hi_twoSig} of the high values are within two sigma')
print(f'High Sigma Exp: \t {hiSig}')
print(f'High Sigma Theory: \t {hi_sigTheory}')
print(f'High reliability factor: {hi_rel}')


print('\nPoissonian Chi-Squared:',chiP)
print(f'N Bar Low: \t {loBar}')
print(f'{lo_oneSig} of the low values are within one sigma')
print(f'{lo_twoSig} of the low values are within two sigma')
print(f'Low Sigma Exp: \t\t {loSig}')
print(f'Low Sigma Theory: \t {lo_sigTheory}')
print(f'Low reliability factor: {lo_rel}\n')

df = pd.DataFrame({
        'NBar' : [hiBar,loBar],
        'NBarUnc' : [hiBar_unc,loBar_unc],
        'sigTheory' : [hi_sigTheory,lo_sigTheory],
        'sigExp' : [hiSig,loSig],
```

```
        'ChiSq': [chiG,chiP],
        '1sig': [hi_oneSig,lo_oneSig],
        '2sig': [hi_twoSig,lo_twoSig],
        'Reliability' : [hi_rel,lo_rel],
        })

print(df.to_latex())
```