Data Analysis - Mini-project

June 5, 2021

0.1 Am241 - ADC histogram and Gaussian fitting project

0.1.1 by Tran Thi Hien Mai and Nguyen Tuan Ngoc

This project we work on isotope Am241 and its ADC spectra. The measurements are done with the 55V input voltage for SiPM.

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from scipy import optimize
import os
from os import listdir

Path = 'C:\\Users/DELL/Python.Practice/Data Analysis/Part2/Project/
 ↪Measure_Am241_55V/'
```

0.2 1. Extracting data

There are 52,666 ADC files in folder Measure_Am241_55V. We only select the files having data, and skip the ones having no data (the files having '._ADC' in their filename). Also, there are High Gain files and Low Gain files needed to separate.

[2]:
```
FileList = os.listdir(Path)
HG_List = []
LG_List = []
Error = []
ADC_List = []

count = 0
for i in range(len(FileList)):
    if '._ADC' in FileList[i]:
        Error.append(FileList[i])
        count +=1
    else:
        ADC_List.append(FileList[i])
print(count)
print("There are totally", len(FileList), "files in this directory.")
```

```
#print(FileList[26328:26330])
print("There are", len(Error), "files containing no data.")
print("There are",len(ADC_List), "ADC files.")
for i in range(len(ADC_List)):
    if 'ADC_ReadOut_HG' in ADC_List[i]:
        HG_List.append(ADC_List[i])
    else:
        LG_List.append(ADC_List[i])
HG_List.sort()
LG_List.sort()

print("There are", len(HG_List),"High Gain files.")
print("There are", len(LG_List),"Low Gain files.")
```

```
26329
There are totally 52666 files in this directory.
There are 26329 files containing no data.
There are 26337 ADC files.
There are 13161 High Gain files.
There are 13176 Low Gain files.
```

- Next, we read the selected .lvm files using np.loadtext, skipping 1st row (NaN values)
- Then,we sum the ADC values from channels 5,6,9,10 for each High Gain (HG) file or each Low Gain (LG) file
- Result should be summing ADC values of 4 channels corresponding to the number of files read.

[3]:
```
SUM_HG = np.zeros(len(HG_List))
SUM_LG = np.zeros(len(LG_List))

for i in range(len(HG_List)):
    #print ('HG_List[', i, '] = ', HG_List[i])
    HG_Data = np.loadtxt(Path+HG_List[i], skiprows=1)
#    print(HG_Data[5][2],HG_Data[6][2],HG_Data[9][2], HG_Data[10][2])
    SUM_HG[i] = HG_Data[5][2]+HG_Data[6][2]+HG_Data[9][2]+HG_Data[10][2]

for i in range(len(LG_List)):
    #print ('LG_List[', i, '] = ', LG_List[i])
    LG_Data = np.loadtxt(Path+LG_List[i], skiprows=1)
#    print(LG_Data[5][2], LG_Data[6][2], LG_Data[9][2], LG_Data[10][2])
    SUM_LG[i] = LG_Data[5][2]+ LG_Data[6][2]+ LG_Data[9][2]+ LG_Data[10][2]

print(SUM_HG.shape)
print(SUM_HG)
print(SUM_LG.shape)
print(SUM_LG)
```

```
(13161,)
```

```
[4480. 4687. 4712. ... 4671. 4704. 4505.]
(13176,)
[3716. 3736. 3748. ... 3732. 3735. 3718.]
```

0.3 2. Plotting histogram

0.3.1 2.1. High Gain histogram

- Focus on the ADC values between 2000 and 8000.
- Setting bins number = 4000

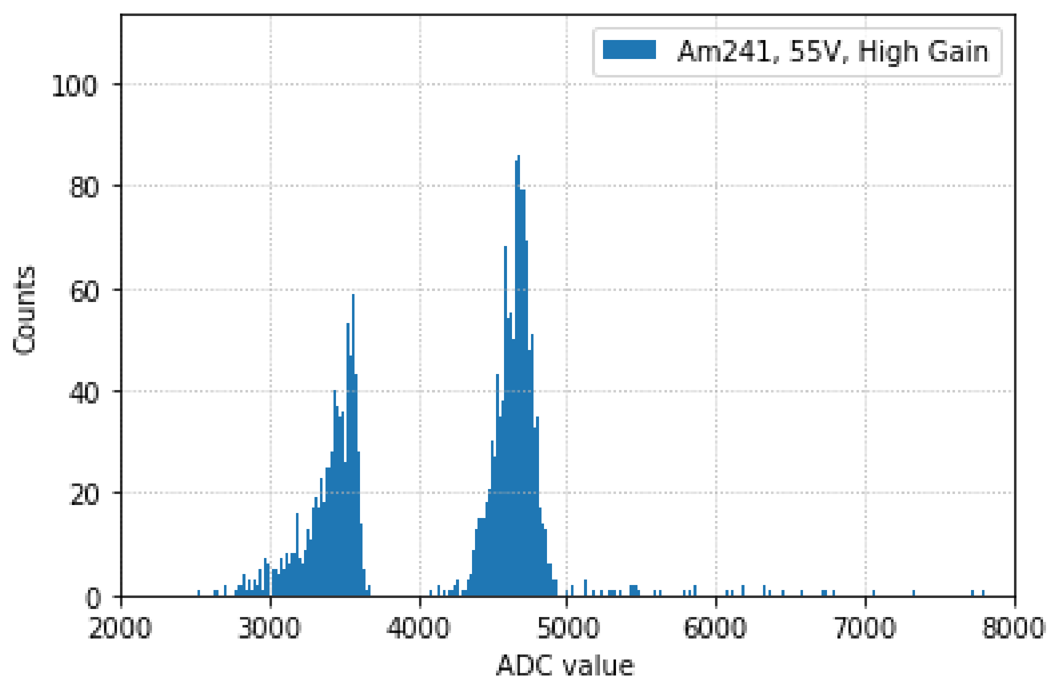
[31]:
```
# High Gain Histogram
x_min_HG = 2000
x_max_HG = 8000
fit_range_HG = [[4200], [5500]]

# Low Gain Histogram
x_min_LG = 3500
x_max_LG = 4000
fit_range_LG = [[3690],[3770]]
```

[6]:
```
print(max(SUM_HG))
print(min(SUM_HG))
bins1 = 4000

fig1, ax1 = plt.subplots()
#ax1.set_axisbelow(True)
ax1.set_xlim(2000,8000)
(counts1,edges1, patches1) = plt.hist(SUM_HG, bins1, label="Am241, 55V, High␣
 ↪Gain") # bins1 = 4000
plt.xlabel("ADC value")
plt.ylabel("Counts")
plt.legend(loc = 'upper right')
plt.grid(linestyle=':')
#fig1.tight_layout()
plt.savefig("Am241_HG.png",facecolor='w', dpi=600)
plt.show()
```

```
13680.0
2334.0
```



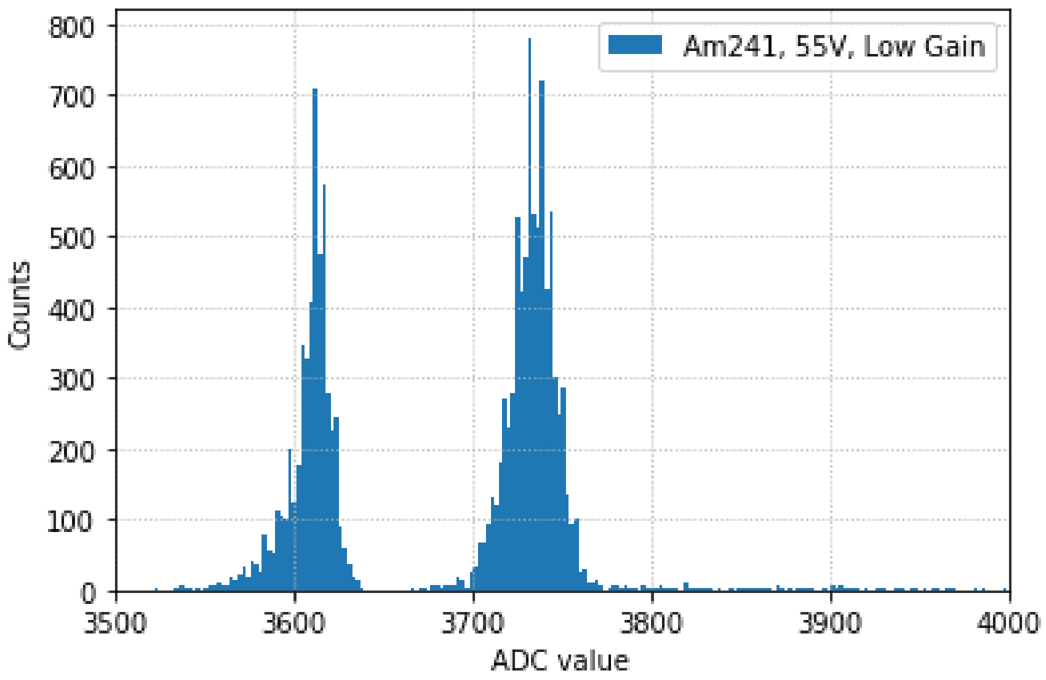
0.3.2 2.2. Low Gain histogram

- Focus on the ADC values between 3500 and 4000.
- Setting bins number = 4000

[27]:
```
print(max(SUM_LG))
print(min(SUM_LG))
bins2 = 4000

fig2,ax2 = plt.subplots()
#ax2.set_axisbelow(True)
ax2.set_xlim(3500,4000)
(counts2,edges2, patches2) = plt.hist(SUM_LG, bins2, label="Am241, 55V, Low␣
 ↪Gain")  # bins2 = 6000
plt.xlabel("ADC value")
plt.ylabel("Counts")
plt.legend(loc = 'upper right')
plt.grid(linestyle=':')
#fig2.tight_layout()
plt.savefig("Am241_LG.png",facecolor='w', dpi= 600)
plt.show()
```

```
12899.0
3491.0
```



0.4 3. Gaussian fit

- Each peak of the histogram can be considered as a Gaussian distribution. Gaussian lines are often above a Compton continuum.
- We model a model to make the histogram fit both a Gaussian distribution and a linear function.
- Fitting function: ## $f(x) = a\exp(\frac{-(x-x_0)}{2(\sigma)^2}) + bx + c$

[15]:
```
def GaussianFit(x, a, x0, sigma, b, c):
    return a*np.exp(-(x-x0)**2/(2*sigma**2))+b*x+c
```

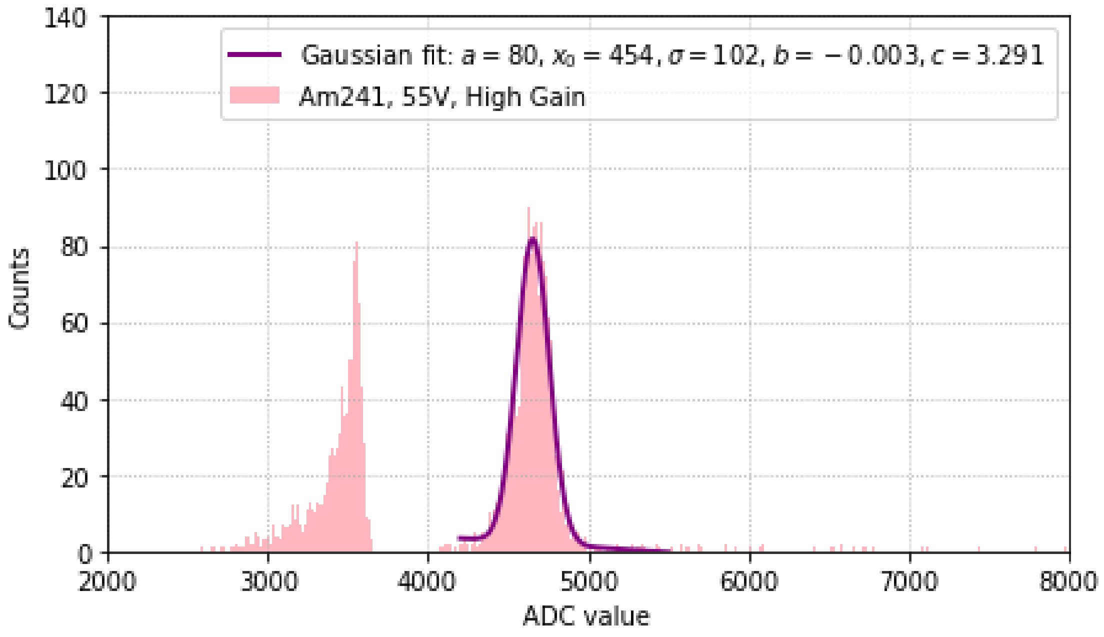
- We also calculate binscenters to take the center value of two edges to get a more nicely fitting result.
- Using curve_fit from scipy: - popt: Optimal values for the parameters so that the sum of the squared residuals of f(xdata, *popt) - ydata is minimized. - pcov: the covariance of popt.

[30]:
```
binscenters = np.array([0.5 * (edges1[i] + edges1[i+1])  for i in␣
 ↪range(len(edges1)-1)])
print(binscenters)

x1 = []
y1 = []
for i in range(len(edges1)):
    if (edges1[i] > 4200) and (edges1[i] < 5500):
        y1.append(counts1[i])
        x1.append(binscenters[i])
y1_data = np.array(y1)
x1_data = np.array(x1)
print(len(x1_data))
print("Highest counts (Peak):",max(y1_data))

popt, pcov = curve_fit(GaussianFit, x1_data-x1_data[0],y1_data)
fig1, ax1 = plt.subplots(1,1,figsize=(7,4))
ax1.set_xlim(2000,8000)
ax1.set_ylim(0,140)
(counts1,edges1, patches1) = plt.hist(SUM_HG, bins1,color='lightpink',␣
 ↪label="Am241, 55V, High Gain")
plt.plot(x1_data, GaussianFit(x1_data-x1_data[0], *popt), color='purple',␣
 ↪linewidth=2, label= r'Gaussian fit: $a=%5.0f, x_0=%5.0f, \sigma=%5.0f, b = %5.
 ↪3f, c = %5.3f$' % tuple(popt))
plt.xlabel("ADC value")
plt.ylabel("Counts")
plt.legend(fontsize=10, loc='upper right')
plt.grid(linestyle=':')
plt.savefig("ADC_fit_HG.png", facecolor= 'w', dpi= 600)
plt.show()
```

```
[ 2335.41825  2338.25475  2341.09125 ... 13672.90875 13675.74525
 13678.58175]
459
Highest counts (Peak): 108.0
```

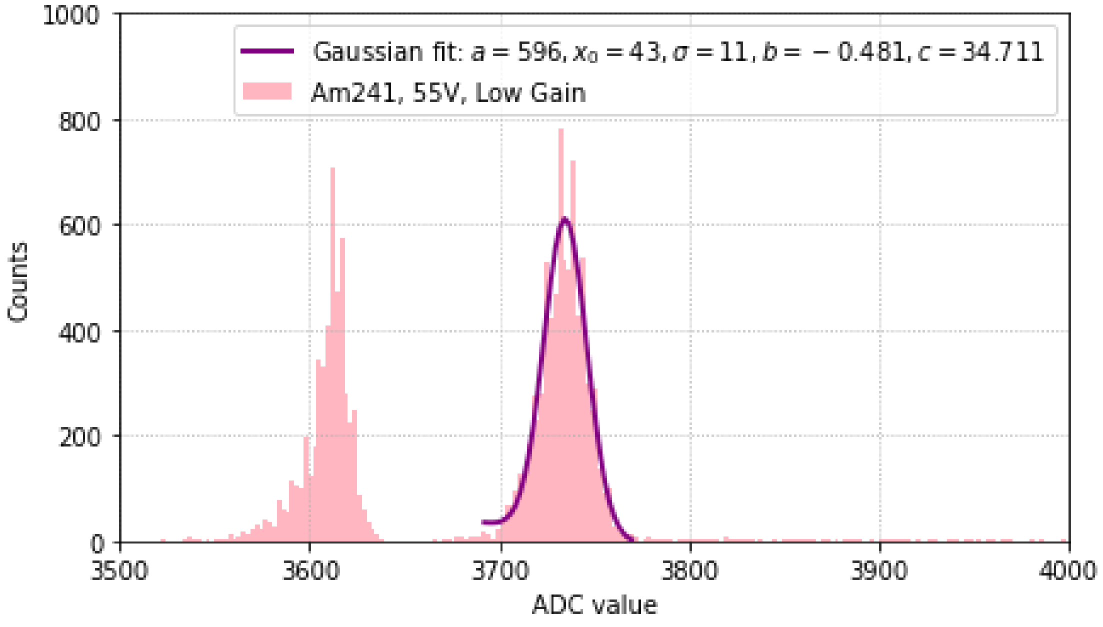


```
[29]: binscenters = np.array([0.5 * (edges2[i] + edges2[i+1])  for i in␣
       ↪range(len(edges2)-1)])
      print(binscenters)

      x2 = []
      y2 = []
      for i in range(len(edges2)):
          if (edges2[i] > 3690) and (edges2[i] < 3770): #3690,3770
              y2.append(counts2[i])
              x2.append(binscenters[i])
      y2_data = np.array(y2)
      x2_data = np.array(x2)
      print(len(x2_data))
      print("Highest counts (Peak):",max(y2_data))

      popt, pcov = curve_fit(GaussianFit, x2_data-x2_data[0], y2_data)
      fig2,ax2 = plt.subplots(1,1, figsize=(7,4))
      ax2.set_xlim(3500,4000)
      ax2.set_ylim(0,1000)
      (counts2,edges2, patches2) = plt.hist(SUM_LG, bins2,color='lightpink',␣
       ↪label="Am241, 55V, Low Gain")
      plt.plot(x2_data, GaussianFit(x2_data-x2_data[0], *popt), color='purple',␣
       ↪linewidth=2,label= r'Gaussian fit: $a=%5.0f, x_0=%5.0f, \sigma=%5.0f, b = %5.
       ↪3f, c = %5.3f$' % tuple(popt))
      plt.xlabel("ADC value")
      plt.ylabel("Counts")
      plt.legend(fontsize=10, loc = 'upper right')
      plt.grid(linestyle=':')
      plt.savefig("ADC_fit_LG.png", facecolor='w', dpi=600)
      plt.show()
```

```
[ 3492.176   3494.528   3496.88   ... 12893.12   12895.472 12897.824]
34
Highest counts (Peak): 782.0
```


Gaussian fit: $a = 596, x_0 = 43, \sigma = 11, b = -0.481, c = 34.711$
Am241, 55V, Low Gain
Counts
ADC value
1000
800
600
400
200
0
3500
3600
3700
3800
3900
4000