Nuclear Physics Statistics Project

Moises Barbera Ramos
The University of Liverpool, Physics Department, Course Phys392
17/12/2019

Abstract

In this study two ranges from an Eu152 isotope dataset have been investigated aiming to find the Minimum Detectable Activity of the sample by a spectrometer. A study of 3 peaks in the 4200 to 8000 channels range and a study of a single peak in the 1 to 800 channels range and above 77 keV. This report shows the individual analysis of each peak identifying the peak count and background to finally find their respective MDAs, which in average equals 13.37 Bq.

1. Introduction

In the field of nuclear physics, there is special interest on the Minimum Detectable Activity (MDA), a statistical representing the smallest quantity of a radioisotope which can be detected with a certain confidence in a system.

In this report, a study of data produced from a gamma ray detector which is placed at a fixed distance from a source of Eu152 is presented, and a 95% confidence is considered for the statistics.

In order to perform this study, two ranges of channel data have been selected from the Eu152 dataset. This dataset contains the number of counts, the energy and channel as recorded from a digital spectrometer. The ranges studied here are from channel 1 to 800 and from 4200 to 8000, and the most relevant peaks from each are analyzed to find their MDA.

It is important to note, that only peaks above a 77 keV energy threshold are considered since we are looking to analyze peaks from gamma rays and considering peaks below this energy would mean X-rays are being studied instead.

In order to perform this MDA analysis, the peak area measurement, count estimation, the Critical Limit (L_C), and the Detection Limit (L_D), must be calculated first

- Peak area measurement

Here, a summation of the number of counts in each of the studied channels inside the peak is performed including a subtraction of background present in the peak.

- Count estimation

The channel numbers coinciding with the upper and lower edges of the peak region must be first estimated.

Using a gaussian and background fit on the peak, the number of counts can be calculated, it is derived as follows:

$$N = C - B \qquad \text{(Eq. 1)}$$

Where C is the total number of counts in peak region, and B is the number of counts from the background in the same region.

- Critical Limit (L_C)

States the statistical significance of a measured peak area.

$$L_C = K_\alpha \cdot \sigma_0 \qquad \text{(Eq. 2)}$$

Where K_α for a 95% confidence limit equals 95% and the σ_0 equals to sigma of variance 2B:

$$L_C = 1.645 \cdot \sqrt{2B} \qquad \text{(Eq. 3)}$$

Then the confidence limits, while N > L_C equals:

$$N \pm K\sigma \qquad \text{(Eq. 4)}$$

Where K is again 1.645 and σ is $\sqrt{N + 2B}$

- Detection Limit (L_D)

Following the condition that detection limit must be above critical limit, L_D can be described as:

$$L_D = K_\alpha^2 + 2 \cdot K_\alpha \cdot \sigma_\alpha \qquad \text{(Eq. 5)}$$

Finally, understood the maths behind this process, we can combine our findings to calculate the Minimum Detectable Activity, Eq. 6, by calculating the count rate calculated as the L_D/t where t is the live time of Eu152 (299.8 seconds) and this would be divided by the branching ratio of the gamma transition, f, which is set to 1 in this study and by the efficiency of the detector which varies with energy as shown in table 1.

Energy/keV	Efficiency	Energy/keV	Efficiency
59.5	0.042293	661.7	0.012586
88.0	0.047077	898.0	0.008659
122.1	0.051100	1173.2	0.006979
165.9	0.042985	1332.0	0.006235
391.7	0.019621	1836.1	0.005067

Table 1: Measured detector efficiency

$$MDA = \frac{L_D}{t\varepsilon f} \qquad \text{(Eq. 6)}$$

2. Methods

Having the data divided in channel, count and energy, as seen on table 1, a visualization of the data could be plotted for each of the two ranges due to study, showing the log of the number of counts in the y axis and the channel on the x axis aiming to identify the 3 strongest peaks in the range 4200 to 8000 and the single strongest peak in the 1 to 800 channel range.

By doing the logarithm of the number of counts, by implementing the semiology function in python, we respond to the skewness towards large values so that the peaks can be better interpreted. Manly the highly skewed distributions become less skewed, figure 1.

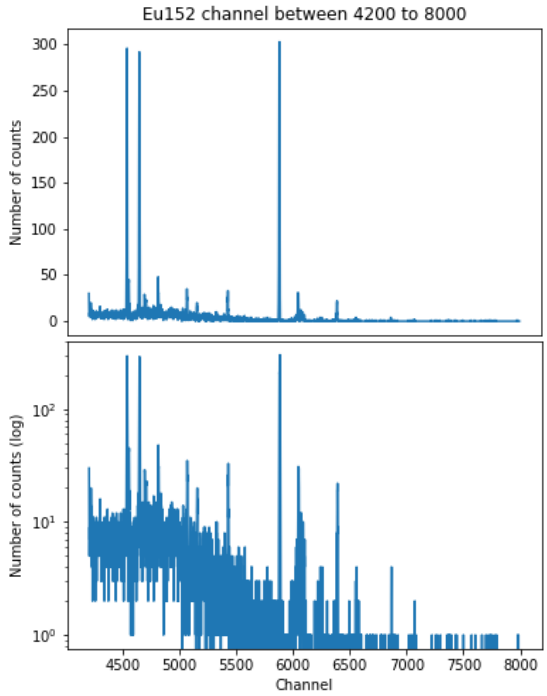

Figure 1: In the top plot there is the data from Eu152 dataset plotted for ranges between 4200 to 8000 channels with Counts versus Channel while the bottom plot shows the same information but with the log of counts plotted instead.

Once the relevant peaks have been selected a gaussian fit is fitted on top of it so that we can better understand their distribution. The gaussian function created for this purpose is shown below, code 1, and the results are shown on the following section 3.

```
# Defined a new model containing the Gaussian peak observed in the data
def gaussian(x, A, mu, sigma, a, b):
    return A*np.exp(-((x-mu)**2) / (2*(sigma**2))) + np.mean(range_bg)
```

Code 1: Code showing the fit method used for each peak. x is the data plotted in the x axis, A is the amplitude of the peak, mu is the mean, sigma is the standard deviation from the mean and a and b are parametres in the distribution. Then the model is corrected adding the mean of the background.

The analysis starts by calculating the peak area measurement and count estimation as described above.

Then, in order to stablish the statistical significance, it is necessary to account for the uncertainty in the background as the peaks studied only become significant after exciding a certain value above the background.

	Channel	Count	Energy
4200	4201.0	30.0	1005.639982
4201	4202.0	19.0	1005.879421
4202	4203.0	5.0	1006.118861
4203	4204.0	5.0	1006.358301
4204	4205.0	7.0	1006.597740

Table 2: Table showing the first five elements of the range set between 4200 and 8000 channels starting right at the first channel passed 4200. It presents the information regarding the Channel, the Counts and the Energy.

The critical limit states the net number of counts above a given number of standard deviations that should be satisfied in order to confirm that the net count is valid and not the result of simple background recordings.

For instance, if the net count is above $K_\alpha \cdot \sigma_0$ (where K_α is a constant that varies according to the percentage of confidence studied and σ_0 is the standard deviation for this peak) then the

count would be statistically significant while it would not be significant if the net count was below or equal this threshold.

As seen on equation 2, $L_C = K_\alpha \cdot \sigma_{0,}$ so if the net count is above L_C then the activity has been detected and a confidence limit, equation 4, can be introduced for this data. On the other hand, should the count lay below L_C the sample would be considered inactive.

```
# The critical limit for a confidence og 95% is given by:
Lc = 1.645*(2*B)**0.5
print('Critical Limit (Lc) = {:.3f}'.format(Lc))
print('')
print('As N > Lc - because {:.3f} > {:.3f}'.format(N, Lc))
print('')

#Confidence Limit
Sigma_N = (N + 2*B)**0.5
Conf_lim_err = 1.645*Sigma_N
print('Then the confidence limit = {:.3f} +/- {:.3f}'
      .format(N, Conf_lim_err))
```

Code 2: This algorithm shows the method used to find the critical limit, considering a 95% confidence so that the value of K is 1.645, and then the confidence limit is also calculated. The values of N and B correspond to the count estimation from each peak.

If we want to have a 95% probability of detection, there should be a minimum number of counts from the sample to be certain about this condition. In the case that the sample activity was right at the critical limit (L_C) then, only a 50% of the time a significant result would be detected, and this wouldn't provide useful statistical information. Introducing the detection limit (L_D) we can overcome this situation as it lies above the critical limit.

```
# for Kα = Kβ = 1.645

Ld = 2.71 + 4.65*(B)**0.5

print('Detection limit = {:.3f}'.format(Ld))
```

Code 3: Calculations used to find the detection limit where K(alpha) and k(beta) are both 1.645 for a 95% confidence. The algorithm has already undergone the transformations from equation 5.

Then, it can be understood that if the expected count was below the critical limit no activity would be measured although, if this count was above the critical limit and below detection limit there is a chance that activity would be detected. Also, if counts were above detection limit, there is a high probability that activity would be detected.

Finally, the minimum detectable activity, introduced previously, for each peak, is calculated following the next algorithm, code 4. It represents the smallest quantity of a radioisotope which can be detected with 95% confidence.

```
# Knowing the energy range we know the efficiency of accelerator
eff_1 = 0.006979

# Live time for Eu152 (s)
t_Eu = 299.8

# Branching ratio assume f = 1
f = 1

# Minimum Detectable Activity
MDA = (Ld) / (t_Eu * eff_1 * f)

print('Minimum Detectable Activity = {:.3f} Bq'.format(MDA))
```

Code 4: The present algorithm calculates the MDA using the count rate at the peak, which is the detection limit over the time taken to measure the data, the branching ratio, which is considered to be 1 and the efficiency which changes for any given energy. The live time for the used Eu152 sample used in this report is 299.8 seconds. In the case of the energy range of the peak being between energies 898.0 and 1173.2 keV then an efficiency of 0.006979 is used.

3. Results

This study has analyzed 4 peaks, 3 from range 4200 to 8000 channels, figure 2, figure 3 and figure 4. While the 4th peak studied is from the range 1 to 800 channels, figure 6.

- Peak 1

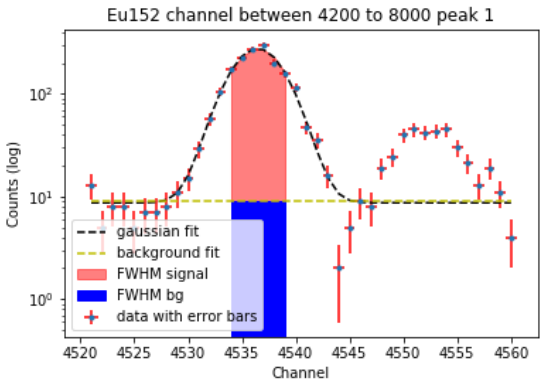

Figure 2: Peak 1 from figure 1, all components are described in the legend.

Critical Limit = 16.808

And as N = 1214.8

Then N > L_C and the confidence limit is 1214.8 ± 59.748.

From code 3, the detection limit was found to be 36.306 and from code 4, the MDA = 17.352 Bq where the efficiency is 0.006979.

- Peak 2

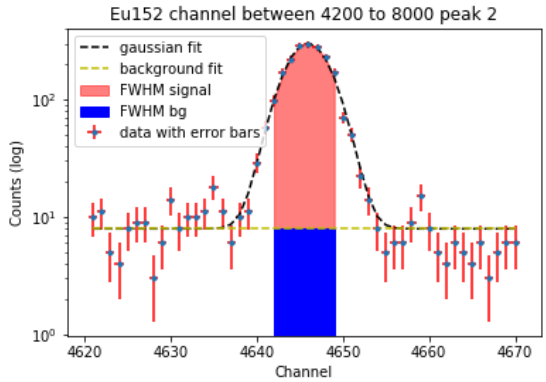

Figure 3: Peak 2 from figure 1, all components are described in the legend.

Critical Limit = 18.494

N = 1570.8

As N > L_C

Confidence Limit = 1570.8 ± 67.769

Detection Limit = 39.677

With efficiency = 0.006979, the MDA = 18.963 Bq.

- Peak 3

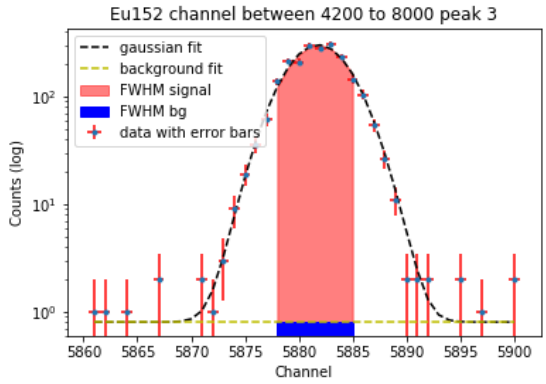

Figure 4: Peak 3 from figure 1, all components are described in the legend.

Critical Limit = 5.885

N = 1727.6

As N > L_C

Confidence Limit = 1727.6 ± 68.626

Detection Limit = 14.474

With efficiency = 0.005067, the MDA = 9.528 Bq.

- Peak 4

This is the strongest peak present in the 1 to 800 channel range seen in figure 5:

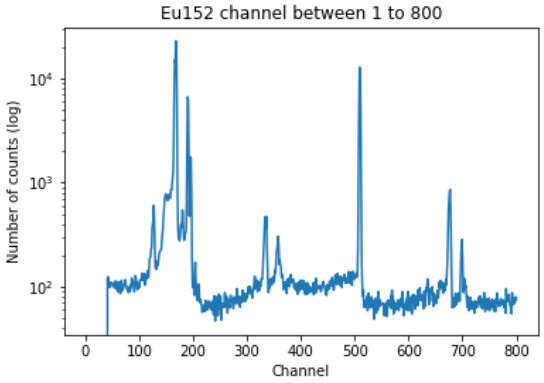

Figure 5: Eu152 sample data distributed from channels 1 to 800. Plots the log of the number of counts versus the channel range.

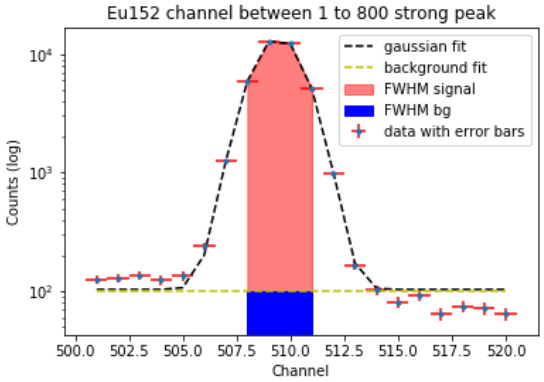

Figure 6: Peak 4, strongest peak from figure 5 after 77 keV energy range as below this threshold we record X-rays rather than the gamma rays we are interested in. all components are described in the legend and the background plotted, as it is not constant, is the local background around the peak itself.

Critical Limit = 47.283

N = 31846.9

As N > L_C

Confidence Limit = 31846.9 ± 297.345

Detection Limit = 97.221

With efficiency = 0.047077, the MDA = 7.668 Bq.

4. Conclusion

This study showcases the analysis of data from a radioactive source recorded by a spectrometer. For two different ranges of channels, from 4200 to 8000 and from 1 to 800, the most meaningful peaks of data have been identified and a gaussian and background model has been fitted into it to help identify the peak area measurement and count estimation. Also, to identify the amount of data detected as background.

Because of the relevance of the Minimum Detectable Activity in the field of nuclear physics, this one has been calculated for each of the studied ranges. Each of the results from the minimum detectable activity show the resulting activity concentration for the Eu152 isotope.

	MDA (Bq)
PEAK 1	17.352
PEAK 2	18.963
PEAK 3	9.528
PEAK 4	7.668
Average	13.37775

Table 3: Shows the different MDA values for each of the studied peaks, there is a noticeable difference between the first two calculated values and the last two which may conclude there has been an error either on the data taking or on the analysis. The average MDA for the Eu152 isotope has also been calculated.

And the average Minimum Detectable Activity for the Eu152 isotope is approximately 13.37 as calculated on this study.

If the statistical variation from each peak was greater than the actual amount of radioactivity in this isotope, it wouldn't be statistically significant, and it wouldn't be detected.

Appendix:

CODE:

```
import numpy as np
import math
%matplotlib inline
import matplotlib.pyplot as plt
from scipy.io import  loadmat
import pandas as pd
from scipy.stats import norm, chi2
from scipy.optimize import curve_fit
import warnings
warnings.filterwarnings("ignore")


# Study the mat file components
mat = loadmat('Eu152.mat', mat_dtype = True, squeeze_me = True)
mat


W = loadmat('Eu152.mat' , mat_dtype = True , squeeze_me = True)
locals().update({k : W[k] for k in ['channel2', 'count2', 'energy2']})


# create dataframe to inspect better given data
data = channel2, count2, energy2
data
df = pd.DataFrame(list(data))
df_transposed = df.T # Transposes dataframe leading to 3 columns only

headers = ['Channel', 'Count', 'Energy']
df_transposed.columns = headers      # Change the header name of each colun

# After some calculations I found that around 21-29% of the counts in column count is 0
df_transposed.head()



# Data Range E between 4200 to 8000

df_range = df_transposed.iloc[4200:8001]
df_range.head()


# Detector efficiency and length of time the data were taken over
# Variables for Energy (keV) = efficiency

Energy_59_5 = 0.042293
Energy_88_0 = 0.047077
Energy_1221_1 = 0.051100
Energy_165_9 = 0.042985
Energy_391_7 = 0.019621
Energy_661_7 = 0.012586
Energy_898_0 = 0.008659
```

```
Energy_1173_2 = 0.006979
Energy_1332_0 = 0.006235
Energy_1836_1 = 0.005067



# Channel only Range E between 4200 to 8000
Channel_range = channel2[4200:8001]

# Count only Range E between 4200 to 8000
Count_range = count2[4200:8001]

# Energy only Range E between 4200 to 8000
Energy_range = energy2[4200:8001]



plt.plot(Channel_range, Count_range)        # Plot data with log in data y
plt.tick_params(
   axis='x',         # changes apply to the x-axis
   which='both',     # both major and minor ticks are affected
   bottom=False,     # ticks along the bottom edge are off
   top=False,        # ticks along the top edge are off
   labelbottom=False) # labels along the bottom edge are off
plt.title('Eu152 channel between 4200 to 8000')
#plt.xlabel('Channel')
plt.ylabel('Number of counts')
plt.savefig("Eu152_4200_8000(2).png")


plt.semilogy(Energy_range, Count_range)
plt.xlabel('Energy (keV)')
plt.ylabel('Number of counts (log)')


# Defined a new model containing the Gaussian peak observed in the data + the background
def gaussian(x, A, mu, sigma, a, b):
   return A*np.exp(-((x-mu)**2) / (2*(sigma**2))) + np.mean(range_bg)
```

PEAK 1

```
# Peak 1
Channel_range_peak1 = channel2[4520:4560]
Count_range_peak1 = count2[4520:4560]
Energy_range_peak1 = energy2[4520:4560]


# Guessed values for [A, mu, sigma, a, b]
A = 300
mu = 4538                  # guessed peak
sigma = 2.5
a = 4520
b = 9
range_bg = count2[4520:4530]    #guessed background
```

```
solu, cov = curve_fit(gaussian, Channel_range_peak1, Count_range_peak1, p0 = [A, mu, sigma, a, b])

# Fitted values for [A, mu, sigma, a, b]
A_fit = solu[0]
mu_fit = solu[1]
sigma_fit = solu [2]
a_fit = solu[3]
b_fit = solu[4]

# Print fitted values
print('Fit value for A is: {:.3f}, for mu is: {:.3f}, for sigma is: {:.3f}, for a is: {:.3f} and for b is: {:.3f} '
    .format(A_fit, mu_fit, sigma_fit, a_fit, b_fit))

# Print the gaussian fit and the data
#plt.plot(Channel_range_peak1,  gaussian(Channel_range_peak1, A_fit, mu_fit, sigma_fit, a_fit, b_fit), 'k--
', label = 'Fitted peak model')

# Each chaannel separated by a 0.5 keV window so we can assume 0.5 error on channel
# Channel number = Energy/0.2393 + 0.2462
error_channel = 0.5
error_counts_1 = Count_range_peak1**0.5
#plt.errorbar(Channel_range_peak1,  Count_range_peak1,   error_counts_1, error_channel, fmt = '.',
ecolor = 'r', label = 'data with error bars')


FWHM = 2*((2*np.log(2))**0.5)*sigma_fit
print('FWHM value = {:.1f}'.format(FWHM))
print('')

# Total number of counts
# Peak region
# FWHM approx. region
peak_reg                    =                    list(count2)[list(channel2).index(int(mu_fit)-int(FWHM/2)-
1):list(channel2).index(int(mu_fit)+int(FWHM/2)+1)]
C = sum(peak_reg)
print('C = ', C)
print('')
# Counts bg
avg_bg = sum(range_bg)/len(range_bg)
B = avg_bg*len(peak_reg)
print('B = ', round(B,3))
print('')
# Net number of counts
N = C - B
print('Net number of counts N = {:.1f}'.format(N))

fill_reg = list(channel2)[int(mu_fit)-int(FWHM/2)-1:(int(mu_fit)+int(FWHM/2)+1)]
y2_fill = b_fit


plt.semilogy(Channel_range_peak1,gaussian(Channel_range_peak1, A_fit, mu_fit, sigma_fit, a_fit, b_fit),
'k--', label = 'gaussian fit')
plt.semilogy(Channel_range_peak1,[y2_fill]*len(Channel_range_peak1), 'y--', label = 'background fit')
plt.fill_between(fill_reg,gaussian(fill_reg,  A_fit,  mu_fit,  sigma_fit,  a_fit,  b_fit),  y2_fill,  color='r',  label  =
'FWHM signal', alpha = 0.5)
plt.fill_between(fill_reg, y2_fill, color='b', label = 'FWHM bg')
```

```
plt.errorbar(Channel_range_peak1, Count_range_peak1,  error_counts_1, error_channel, fmt = '.',  ecolor = 'r', label = 'data with error bars')

plt.title('Eu152 channel between 4200 to 8000 peak 1')
plt.xlabel('Channel')
plt.ylabel('Counts (log)')
plt.legend()
plt.savefig("Eu152_4200_8000_peak1.png")
plt.show()
```

CRITICAL LIMIT

```
# The critical limit for a confidence og 95% is given by:
Lc = 1.645*(2*B)**0.5
print('Critical Limit (Lc) = {:.3f}'.format(Lc))
print('')
print('As N > Lc - because {:.3f} > {:.3f}'.format(N, Lc))
print('')

#Confidence Limit
Sigma_N = (N + 2*B)**0.5
Conf_lim_err = 1.645*Sigma_N
print('Then the confidence limit = {:.3f} +/- {:.3f}' .format(N, Conf_lim_err))
```

DETECTION LIMIT

```
# for Kα = Kβ = 1.645

Ld = 2.71 + 4.65*(B)**0.5

print('Detection limit = {:.3f}'.format(Ld))
```

MDA

```
# To know efficiency we need to know the energy range of our peak
print('Energy range peak 1 between ' + str(list(Energy_range_peak1)[0]) + ' and ' + str(list(Energy_range_peak1)[-1]))


# Knowing the energy range we know the efficiency of accelerator is between Energy_898_0 and Energy_1173_2
eff_1 = 0.006979

# Live time for Eu152 (s)
t_Eu = 299.8

# Branching ratio assume f = 1
f = 1

# Minimum Detectable Activity
```

```
MDA = (Ld) / (t_Eu * eff_1 * f)

print('Minimum Detectable Activity = {:.3f} Bq'.format(MDA))
```

PEAK 2

```
# Peak 2
Channel_range_peak2 = channel2[4620:4670]
Count_range_peak2 = count2[4620:4670]
Energy_range_peak2 = energy2[4620:4670]


# Guessed values for [A, mu, sigma, a, b]
A = 300
mu = 4645   # guessed peak
sigma = 2.5
a = 4620
b = 8
range_bg = count2[4620:4630]

solu, cov = curve_fit(gaussian, Channel_range_peak2, Count_range_peak2, p0 = [A, mu, sigma, a, b])

# Fitted values for [A, mu, sigma, a, b]
A_fit = solu[0]
mu_fit = solu[1]
sigma_fit = solu [2]
a_fit = solu[3]
b_fit = solu[4]

# Print fitted values
print('Fit value for A is: {:.3f}, for mu is: {:.3f}, for sigma is: {:.3f}, for a is: {:.3f} and for b is: {:.3f} '
    .format(A_fit, mu_fit, sigma_fit, a_fit, b_fit))
print('')
FWHM = 2*((2*np.log(2))**0.5)*sigma_fit
print('FWHM value = {:.3f}'.format(FWHM))
print('')

# Print the gaussian fit and the data
#plt.semilogy(Channel_range_peak2,    gaussian(Channel_range_peak2,  A_fit,  mu_fit,  sigma_fit,  a_fit,
b_fit), 'k--', label = 'Fitted peak model')

# Each chaannel separated by a 0.5 keV window so we can assume 0.5 error on channel
# Channel number = Energy/0.2393 + 0.2462
error_channel = 0.5
error_counts_2 = Count_range_peak2**0.5
#plt.errorbar(Channel_range_peak2,  Count_range_peak2,    error_counts_2,  error_channel,  fmt  =  '.',
ecolor = 'r', label = 'error bars')

# Total number of counts
# Peak region
# FWHM approx. region
peak_reg                =                list(count2)[list(channel2).index(int(mu_fit)-int(FWHM/2)-
1):list(channel2).index(int(mu_fit)+int(FWHM/2)+1)]
C = sum(peak_reg)
```

```
print('C = ', C)
print('')

# Counts bg
avg_bg = sum(range_bg)/len(range_bg)
B = avg_bg*len(peak_reg)
print('B = ', round(B, 3))
print('')

# Net number of counts
N = C - B
print('Net number of counts = {:.1f}'.format(N))


fill_reg = list(channel2)[int(mu_fit)-int(FWHM/2)-1:(int(mu_fit)+int(FWHM/2)+1)]
y2_fill = b_fit


plt.semilogy(Channel_range_peak2,gaussian(Channel_range_peak2, A_fit, mu_fit, sigma_fit, a_fit, b_fit), 'k--', label = 'gaussian fit')
plt.semilogy(Channel_range_peak2,[y2_fill]*len(Channel_range_peak2), 'y--', label = 'background fit')
plt.fill_between(fill_reg,gaussian(fill_reg, A_fit, mu_fit, sigma_fit, a_fit, b_fit), y2_fill, color='r', label = 'FWHM signal', alpha = 0.5)
plt.fill_between(fill_reg, y2_fill, color='b', label = 'FWHM bg')
plt.errorbar(Channel_range_peak2, Count_range_peak2, error_counts_2, error_channel, fmt = '.', ecolor = 'r', label = 'data with error bars')

plt.title('Eu152 channel between 4200 to 8000 peak 2')
plt.xlabel('Channel')
plt.ylabel('Counts (log)')
plt.legend()
plt.savefig("Eu152_4200_8000_peak2.png")
plt.show()
```

CRITICAL LIMIT

```
# The critical limit for a confidence og 95% is given by:
Lc = 1.645*(2*B)**0.5
print('Critical Limit (Lc) = {:.3f}'.format(Lc))
print('')
print('As N > Lc - because {:.3f} > {:.3f}'.format(N, Lc))
print('')

#Confidence Limit
Sigma_N = (N + 2*B)**0.5
Conf_lim_err = 1.645*Sigma_N
print('Then the confidence limit = {:.3f} +/- {:.3f}'.format(N, Conf_lim_err))
```

DETECTION LIMIT

```
# for Kα = Kβ = 1.645
```

```
Ld = 2.71 + 4.65*(B)**0.5

print('Detection limit = {:.3f}'.format(Ld))
```

MDA

```
# To know efficiency we need to know the energy range of our peak
print('Energy range peak 2 between ' + str(list(Energy_range_peak2)[0]) + ' and ' + str(list(Energy_range_peak2)[-1]))

# Knowing the energy range we know the efficiency of accelerator is between Energy_898_0 and Energy_1173_2
eff_2 = 0.006979

# Live time for Eu152 (s)
t_Eu = 299.8

# Branching ratio assume f = 1
f = 1

# Minimum Detectable Activity
MDA = (Ld) / (t_Eu * eff_2 * f)

print('Minimum Detectable Activity = {:.3f} Bq'.format(MDA))
```

PEAK 3

```
# Peak 3
Channel_range_peak3 = channel2[5860:5900]
Count_range_peak3 = count2[5860:5900]
Energy_range_peak3 = energy2[5860:5900]


# Guessed values for [A, mu, sigma, a, b]
A = 300
mu = 5883   # guessed peak
sigma = 2.5
a = 4620
b = 0.8
range_bg = count2[5800:5850]

solu, cov = curve_fit(gaussian, Channel_range_peak3, Count_range_peak3, p0 = [A, mu, sigma, a, b])

# Fitted values for [A, mu, sigma, a, b]
A_fit = solu[0]
mu_fit = solu[1]
sigma_fit = solu [2]
a_fit = solu[3]
b_fit = solu[4]

# Print fitted values
```

```
print('Fit value for A is: {:.3f}, for mu is: {:.3f}, for sigma is: {:.3f}, for a is: {:.3f} and for b is: {:.3f} '
    .format(A_fit, mu_fit, sigma_fit, a_fit, b_fit))
print('')
FWHM = 2*((2*np.log(2))**0.5)*sigma_fit
print('FWHM value = {:.3f}'.format(FWHM))
print('')

# Print the gaussian fit and the data
#plt.plot(Channel_range_peak3,  gaussian(Channel_range_peak3, A_fit, mu_fit, sigma_fit, a_fit, b_fit), 'k--
', label = 'Fitted peak model')

# Each chaannel separated by a 0.5 keV window so we can assume 0.5 error on channel
# Channel number = Energy/0.2393 + 0.2462
error_channel = 0.5
error_counts_3 = Count_range_peak3**0.5
#plt.errorbar(Channel_range_peak3,  Count_range_peak3,   error_counts_3,  error_channel,  fmt  =  '.',
ecolor = 'r', label = 'error bars')

# Total number of counts
# Peak region
# FWHM approx. region
peak_reg                =                list(count2)[list(channel2).index(int(mu_fit)-int(FWHM/2)-
1):list(channel2).index(int(mu_fit)+int(FWHM/2)+1)]
C = sum(peak_reg)
print('C = ', C)
print('')

# Counts bg
avg_bg = sum(range_bg)/len(range_bg)
B = avg_bg*len(peak_reg)
print('B = ', B)
print('')

# Net number of counts
N = C - B
print('Net number of counts = {:.1f}'.format(N))

fill_reg = list(channel2)[int(mu_fit)-int(FWHM/2)-1:(int(mu_fit)+int(FWHM/2)+1)]
y2_fill = b_fit

plt.semilogy(Channel_range_peak3,gaussian(Channel_range_peak3, A_fit, mu_fit, sigma_fit, a_fit, b_fit),
'k--', label = 'gaussian fit')
plt.semilogy(Channel_range_peak3,[y2_fill]*len(Channel_range_peak3), 'y--', label = 'background fit')
plt.fill_between(fill_reg,gaussian(fill_reg,  A_fit,  mu_fit,  sigma_fit,  a_fit,  b_fit),  y2_fill,  color='r',  label  =
'FWHM signal', alpha = 0.5)
plt.fill_between(fill_reg, y2_fill, color='b', label = 'FWHM bg')
plt.errorbar(Channel_range_peak3, Count_range_peak3,  error_counts_3, error_channel, fmt = '.',  ecolor
= 'r', label = 'data with error bars')

plt.title('Eu152 channel between 4200 to 8000 peak 3')
plt.xlabel('Channel')
plt.ylabel('Counts (log)')
plt.legend()
plt.savefig("Eu152_4200_8000_peak3.png")
plt.show()
```

CRITICAL LIMIT

```
# The critical limit for a confidence og 95% is given by:
Lc = 1.645*(2*B)**0.5
print('Critical Limit (Lc) = {:.3f}'.format(Lc))
print('')
print('As N > Lc - because {:.3f} > {:.3f}'.format(N, Lc))
print('')

#Confidence Limit
Sigma_N = (N + 2*B)**0.5
Conf_lim_err = 1.645*Sigma_N
print('Then the confidence limit = {:.3f} +/- {:.3f}'.format(N, Conf_lim_err))
```

DETECTION LIMIT

```
# for Kα = Kβ = 1.645

Ld = 2.71 + 4.65*(B)**0.5

print('Detection limit = {:.3f}'.format(Ld))
```

MDA

```
# To know efficiency we need to know the energy range of our peak
print('Energy range peak 3 between ' + str(list(Energy_range_peak3)[0]) + ' and ' +
str(list(Energy_range_peak3)[-1]))


# Knowing the energy range we know the efficiency of accelerator is between Energy_1332_0 and
Energy_1836_1
eff_3 = 0.005067

# Live time for Eu152 (s)
t_Eu = 299.8

# Branching ratio assume f = 1
f = 1

# Minimum Detectable Activity
MDA = (Ld) / (t_Eu * eff_3 * f)

print('Minimum Detectable Activity = {:.3f} Bq'.format(MDA))
```

FROM 1 TO 800 RANGE

```
# Data Range between 1 to 800

df_range = df_transposed.iloc[:800]
```

```
df_range.tail()

# Channel only Range between 1 to 800
Channel_range = channel2[:800]

# Count only Range between 1 to 800
Count_range = count2[:800]

# Energy only Range between 1 to 800
Energy_range = energy2[:800]


plt.semilogy(Channel_range, Count_range)
plt.title('Eu152 channel between 1 to 800')
plt.xlabel('Channel')
plt.ylabel('Number of counts (log)')
plt.show()
plt.savefig("Eu152_1_800.png")
```

SINGLE STRONGEST PEAK

```
# Peak 1
Channel_range_peak = channel2[500:520]
Count_range_peak = count2[500:520]
Energy_range_peak = energy2[500:520]


# Guessed values for [A, mu, sigma, a, b]
A = 30000
mu = 500   # guessed peak
sigma = 150
a = 0.5
b = 100
range_bg = count2[400:480]    #guessed background

solu, cov = curve_fit(gaussian, Channel_range_peak, Count_range_peak, p0 = [A, mu, sigma, a, b])

# Fitted values for [A, mu, sigma, a, b]
A_fit = solu[0]
mu_fit = solu[1]
sigma_fit = solu [2]
a_fit = solu[3]
b_fit = solu[4]

# Print fitted values
print('Fit value for A is: {:.3f}, for mu is: {:.3f}, for sigma is: {:.3f}, for a is: {:.3f} and for b is: {:.3f} '
    .format(A_fit, mu_fit, sigma_fit, a_fit, b_fit))

# Print the gaussian fit and the data
#plt.plot(Channel_range_peak1,  gaussian(Channel_range_peak1, A_fit, mu_fit, sigma_fit, a_fit, b_fit), 'k--
', label = 'Fitted peak model')

# Each chaannel separated by a 0.5 keV window so we can assume 0.5 error on channel
# Channel number = Energy/0.2393 + 0.2462
error_channel = 0.5
```

```
error_counts = Count_range_peak**0.5
#plt.errorbar(Channel_range_peak1, Count_range_peak1, error_counts_1, error_channel, fmt = '.', ecolor = 'r', label = 'data with error bars')


FWHM = 2*((2*np.log(2))**0.5)*sigma_fit
print('FWHM value = {:.1f}'.format(FWHM))
print('')

# Total number of counts
# Peak region
# FWHM approx. region
peak_reg = list(count2)[list(channel2).index(int(mu_fit)-int(abs(FWHM)/2)-1):list(channel2).index(int(mu_fit)+int(abs(FWHM)/2)+1)]
C = sum(peak_reg)
print('C = ', C)
print('')
# Counts bg
avg_bg = sum(range_bg)/len(range_bg)
B = avg_bg*len(peak_reg)
print('B = ', round(B,3))
print('')
# Net number of counts
N = C - B
print('Net number of counts N = {:.1f}'.format(N))

fill_reg = list(channel2)[int(mu_fit)-int(FWHM/2)-1:(int(mu_fit)+int(FWHM/2)+1)]
y2_fill = b_fit


plt.semilogy(Channel_range_peak,gaussian(Channel_range_peak, A_fit, mu_fit, sigma_fit, a_fit, b_fit), 'k--', label = 'gaussian fit')
plt.semilogy(Channel_range_peak,[y2_fill]*len(Channel_range_peak), 'y--', label = 'background fit')
plt.fill_between(fill_reg,gaussian(fill_reg, A_fit, mu_fit, sigma_fit, a_fit, b_fit), y2_fill, color='r', label = 'FWHM signal', alpha = 0.5)
plt.fill_between(fill_reg, y2_fill, color='b', label = 'FWHM bg')
plt.errorbar(Channel_range_peak, Count_range_peak, error_counts, error_channel, fmt = '.', ecolor = 'r', label = 'data with error bars')

plt.title('Eu152 channel between 1 to 800 strong peak')
plt.xlabel('Channel')
plt.ylabel('Counts (log)')
plt.legend()
plt.savefig("Eu152_1_800_peak(1).png")
plt.show()
```

CRITICAL LIMIT

```
# The critical limit for a confidence og 95% is given by:
Lc = 1.645*(2*B)**0.5
print('Critical Limit (Lc) = {:.3f}'.format(Lc))
print('')
print('As N > Lc - because {:.3f} > {:.3f}'.format(N, Lc))
print('')
```

```
#Confidence Limit
Sigma_N = (N + 2*B)**0.5
Conf_lim_err = 1.645*Sigma_N
print('Then the confidence limit = {:.3f} +/- {:.3f}'.format(N, Conf_lim_err))
```

DETECTION LIMIT

```
# for Kα = Kβ = 1.645

Ld = 2.71 + 4.65*(B)**0.5

print('Detection limit = {:.3f}'.format(Ld))
```

MDA

```
# To know efficiency we need to know the energy range of our peak
print('Energy range strong peak between ' + str(list(Energy_range_peak)[0]) + ' and ' + str(list(Energy_range_peak)[-1]))


# Knowing the energy range we know the efficiency of accelerator
eff = 0.047077

# Live time for Eu152 (s)
t_Eu = 299.8

# Branching ratio assume f = 1
f = 1

# Minimum Detectable Activity
MDA = (Ld) / (t_Eu * eff * f)

print('Minimum Detectable Activity = {:.3f} Bq'.format(MDA))
```