Production and spectroscopic investigation of new neutron-rich isotopes near the neutron N=126 shell closure using the multinucleon transfer reactions

I. Kulikov
CERN, 1211 Geneva, Switzerland
and
V. Vedeneev
JINR, Joliot Curie 6, 141980, Dubna, Russia

November 15, 2020

Abstract

The MASHA setup stands for Mass Analyzer of Super Heavy Atoms. The apparatus has a resolving power of about 1700 and uses the so-called ISOL (Isotope Separation On-Line) method to separate the reaction products. The project has been denoted to the analysis of the data collected from multi-nucleon transfer reactions of $^{40}Ar+^{144}Sm \rightarrow ^{186-xn}Hg+xn$ and $^{48}Ca+^{242}Pu \rightarrow ^{220-xn}Rn+xn$ and to the performance of the position-sensitive detector based in the focal plane of the MASHA setup. For the data analysis, two programs have been written in Python language. The results of the program are energies of the α-active Hg and Rn isotopes. These energies were compared to the tabulated values, and no deviations have been found.

Keywords: mass separation, super heavy elements, chart of nuclides, α-decay, python

1 Introduction

The island of stability is a place on the nuclear chart with a set of isotopes of superheavy elements which considered to be long-lived compare to the known isotopes of these elements. These superheavy elements have been carefully investigated at many nuclear facilities. To list some of them - Joint Institute for Nuclear research(1), GSI Helmholzzentrum fur Schwerionenforschung and Lawrence Berkeley National Laboratory. Since these super heavy elements' production rate is extremely low, the identification and characterization of them require knowledge in both chemistry and nuclear physic fields(2),(3).

The project's main goal was to analyze the heavy neutron-rich nuclei near the N=126 neutron shell closure(4). The $^{180-185}Hg$(5), $^{201-205}Rn$ and $^{212,218,219}Rn$ nuclei has been produced in multi-nucleon transfer reactions of $^{40}Ar+^{144}Sm \rightarrow {}^{186-xn}Hg+xn$ and $^{48}Ca+^{242}Pu \rightarrow {}^{220-xn}Rn+xn$. The products of the reactions were separated by use of Isotope Separator Online(ISOL) technique, and they have been detected by multi-channel Si detector based in the focal of the MASHA apparatus.(6)

The report consists of three parts. The first part describes MASHA spectrometer. The second part denotes the data analysis procedure. The main results of the project are summarized in the conclusion section. The developed code for the analysis is listed and explained in the appendix section.

2 MASHA spectrometer

The main parts of the MASHA mass spectrometer are shown in fig.1. The ion beam of hundreds of MeV energy passes the rotating target and then, the reaction products and the uninterrupted beam stops inside the hot catcher. The hot catcher by itself is a very complex system. The Faraday cup and the pick-up detector has been installed to measure the beam intensity. The pick-up sensor represents an electrically isolated disc with the hole at the center, which allow the products pass it and gather secondary electrons emitted from target. It gives the uninterruptible method of beam intensity measurement. The energy of the beam accumulates on the target inside of the hot catcher. Next, neutral atoms diffuse out from the polygraphene structure of the hot-catcher and get ionized by

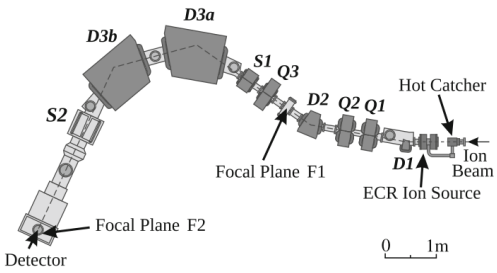


Figure 1: The basic layout of the MASHA mass spectrometer. For detailed description see the text

the use of electron cyclotron resonance (ECR) ion source. The incoming products of nuclear reaction get ionized to $Q = +1$ by applying the ultra-high frequency of 2.45GHz to the energy of 38 keV. The actual mass separation happens in the system of a magneto-optical mass-to-charge ration analyzer, which consists of four dipole magnets (D1, D2, D3a, D3b), three quadrupole lenses (Q1-Q3), and two sextupole lenses (S1, S2). The resolving power of the system easily reaches $\frac{M}{\Delta M}$ =1700. The ions of interest are focused on the focal plane and get detected by a multi-channel Si detector. The front panel of the silicon detector system has 192 strips covered by copper and has an energy resolution of about 30 keV. In order to increase the geometrical efficiency of α-decay products, four additional detector panel has been installed on top-bottom and right-left sides of the front detector panel. This allows detecting 90% of the α-particles. The signals from the detector have been recorded by the unique signal-diagnostic system, described in details here (7).

3 Data analysis

The following section describes the data analysis procedure of super heavy Hg and Rn isotopes and the main results. For the data analysis gathering, two programs have been written in Python language. The first program identifies the peaks on the strip detector and fits the data with three different fit functions of choice. The second program builds the heat map.

For the peak analyzer, the first step is to create the database for the analysis. The files

have been downloaded by the use of the special python package named Pandas. The data consists of two columns (the energy and the counts). The typical spectrum is shown in fig.2. To find the peaks, another spatial python package has been used, namely peakutils. This function asks two parameters, the threshold of the peak and the minimum distance between peaks. After the identification, peaks can be fitted by free different functions. The most straightforward fit function is Gaussian. However, the shape of the peak has an asymmetry. The peak looks more like a combination of Gaussian (right side of the peak) and an exponent (left side of the peak). The literature on the peak shape of α-decays suggests to build the exponential Gaussian hybrid fit function(8),(9). This fit function has fitted most of the peaks. Another case is double structured peaks. This shape can be explained by the fact that some isotopes can endure the α-decay from a different energy level. Therefore, one isotope can have multiple α-decays. The smaller peaks sit on the tail of the most probable α-decay channel. Therefore, to fit such a peak, a particular function was built. It is the sum of a Gaussian and a simple a*x+b line. The peakutils based on the least square minimization method and provides only the systematic uncertainty. The rest of the algorithm is used to build the graphs.

The second step in the data analysis procedure was to build a heat map. For this purpose, the seaborn package has been used. The program makes identification of peaks in the M*N matrix in x and y-direction. X corresponds to the strip number, Y corresponds to the energy channel. After, the program asks the user for a strip number to perform the calibration of the energy. The calibration is done in such a way: the strip has to have at least two peaks. After peaks are found, and the strip is chosen, the program calculates the scale division's value (y2-y1)/(number of channels between two peaks). By this, the program calibrates the y-axis. There was an idea to do calibration using linear extrapolation E=a*x+b, where x is a strip number, and E is energy on y-axes. However, due to the asymmetry of the M*N matrix, it could not be performed, and a more straightforward solution has been used.

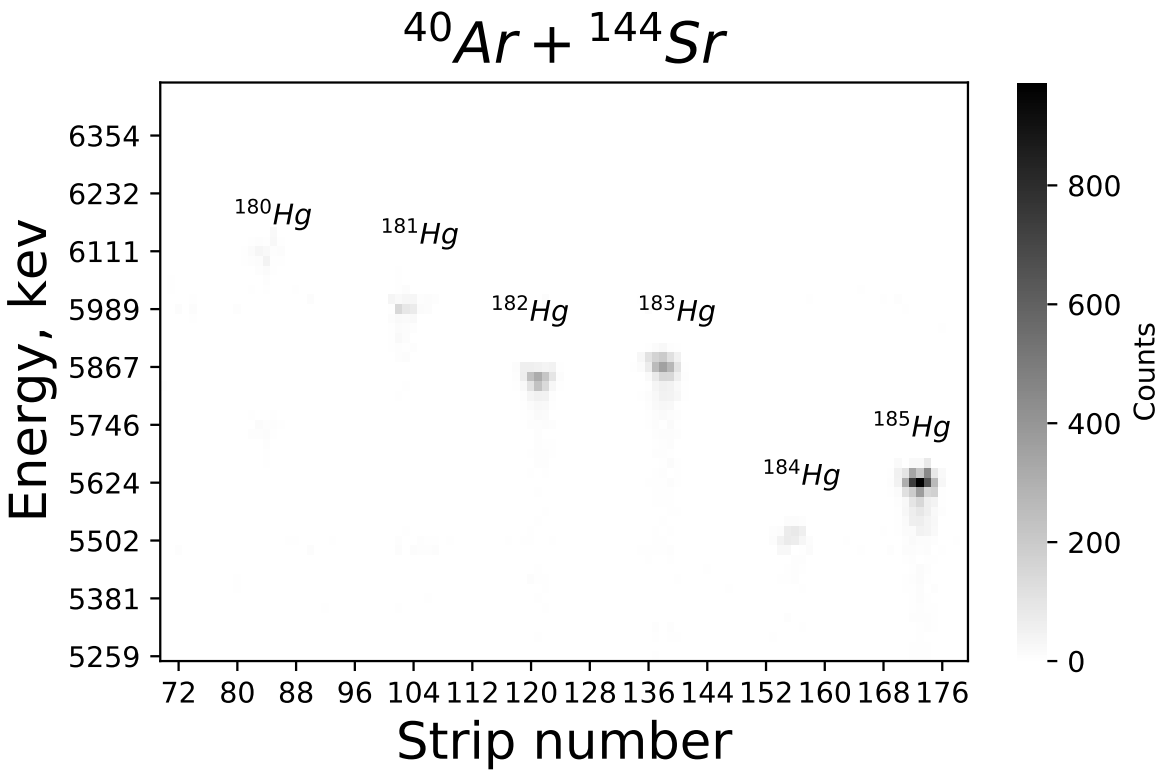


Figure 2: The heat map of ^{180}Hg ions

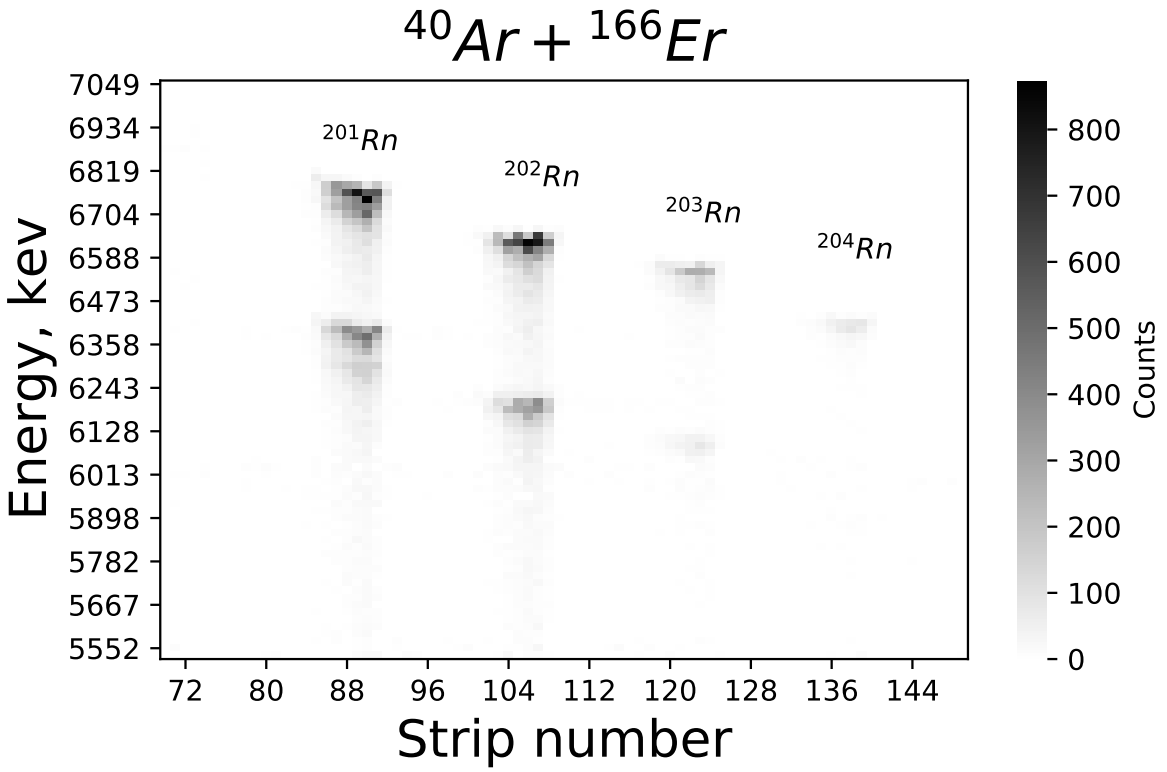


Figure 3: The heat map of $^{201-205}Rn$ ions

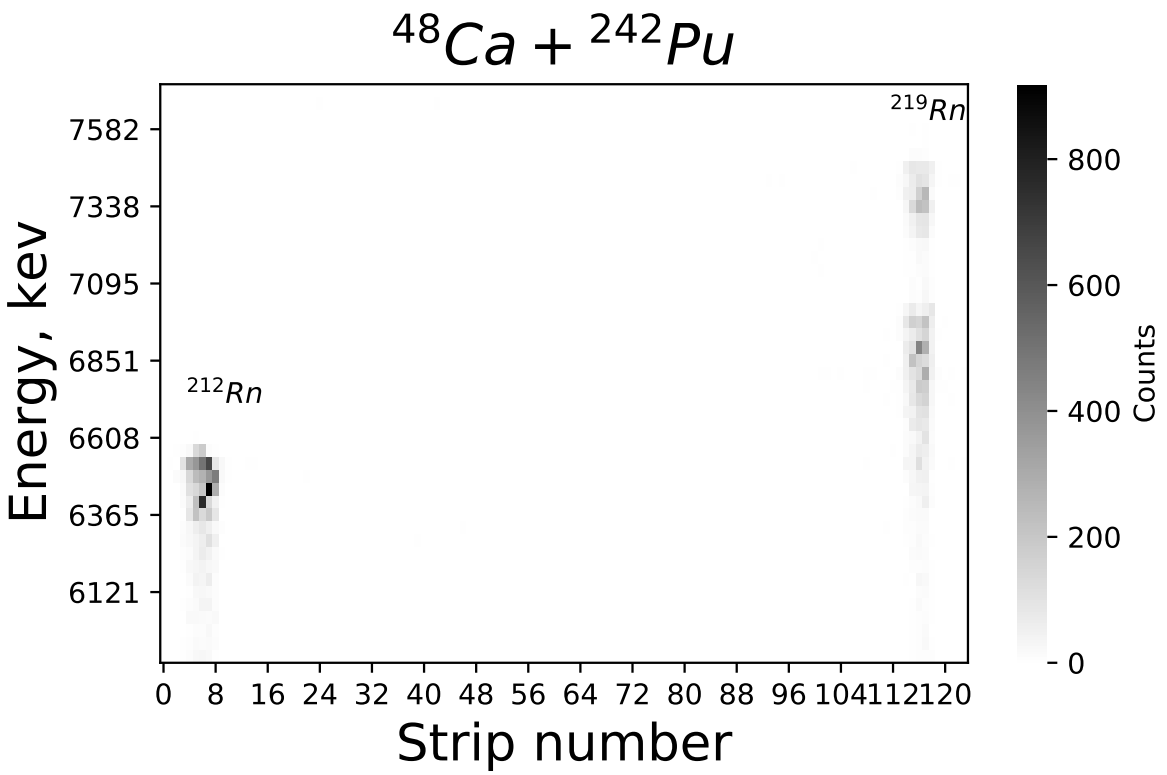


Figure 4: The heat map of ^{212,218,219}Rn ions

The results of the data analysis of $^{180-185}Hg$, $^{201-205}Rn$ and $^{212,218,219}Rn$ nuclei are listed in table 1-3. The literature values are taken from the nuclear chart. The graphs of the identification procedure have been summarized in the appendix section, as well as the results of the fitting. Fig. 3-5 shows the results as a heat map.

Table 1: Summary of the data investigation on Hg isotopes

		Energy, keV	
Nuclide	Counts	Experiment	Literature
^{180}Hg	57	6111.1(14)	6119
^{176}Pt	27	5746(2)	5753
^{181}Hg	217	5997.7(9)	6006
^{177}Pt	54	5497.7(11)	5517
^{182}Hg	553	5871.9(11)	5867
^{182}Hg	18	5697.5(36)	5700
^{178}Pt	19	5452.0(17)	5446
^{183}Hg	601	5902.4(9)	5904
^{183}Au	10	5328.6(80)	5346
^{184}Hg	222	5531.8(10)	5535
^{184}Tl	21	6029.7(13)	5988
^{185}Hg	1628	5657(1.7)	5653
^{185}Au	50	5094.8(34)	5069

Table 2: Summary of the data investigation on *Rn* isotopes

Nuclide	Counts	Energy, keV	
		Experiment	Literature
^{201}Rn	1674	6780.8(8)	6773
^{197}Po	1083	6397(16)	6383
^{197}Po	402	6278(30)	6281
^{202}Rn	1946	6642.6(7)	6640
^{198}Po	860	6191.4(13)	6182
^{203}Rn	231	6498.3(16)	6499
^{203}Rn	608	6561.4(12)	6549
^{199}Po	31	5961.9(27)	5952
^{199}Po	146	6066.7(13)	6059
^{204}Rn	193	6412.7(8)	6419
^{200}Po	15	5848.4(34)	5862
^{205}Rn	40	6261(11)	6268
^{201}Po	6	5771.2(75)	5786
^{205}At	18	5904.6(41)	5902

Table 3: Summary of the data investigation on *Rn* isotopes

Nuclide	Counts	Energy, keV	
		Experiment	Literature
^{212}Rn	1139	6258.9(12)	6264
^{218}Rn	7	7110.8(36)	7129
^{214}Po	6	7658(3.9)	7687
^{222}Ra	8	6586.8(39)	6559
^{210}Po	7	5268.4(139)	5280
^{246}Es	8	7356.3(14)	7360
^{219}Rn	454	6811.4(13)	6819
^{215}Po	450	7379.4(8)	7386
^{211}Bi	338	6630.7(37)	6623
^{227}Pa	84	6447.6(87)	6465
^{231}Np	84	6292.9(65)	6258

4 Conclusion

The literature overview of new neutron-rich isotopes near N=126 shell closure and the literature overview about the MASHA mass spectrometer has been performed. The data analysis of $^{180-185}Hg$, $^{201-205}Rn$ and $^{212,218,219}Rn$ was made. Two programs have been written in python language. The first is used for finding and fitting the peaks. The second one is used to build the heat map. Three different fit functions were written to perform the fitting. The inspiration for fit functions was found in the additional literature sources. The results of the data analysis are the α-decay. They have been compared to the values provided by the Nuclear Chart, and no deviations were found.

5 Acknowledgments

I. K. thanks Viacheslav Vedeneev for nice and helpful supervision. Also, I. K. thanks JINR student training center for such a nice opportunity to learn.

References

[1] G. H.W., Mendeleev's principle against einstein's relativity: news from the chemistry of superheavy elements, Russian Chemical Reviews 78 (2009) 1139–1144.

[2] R. Eichler, N. Aksenov, A. Belozerov, G. Bozhikov, V. Chepigin, S. Dmitriev, R. Dressler, H. Gaggeler, V. Gorshkov, F. Haenssler, M. Itkis, A. Laube, V. Lebedev, O. Malyshev, O. Yu.Ts, O. Petrushkin, D. Piguet, P. Rasmussen, S. Shishkin, A. Shutov, A. Svirikhin, E. Tereshatov, G. Vostokin, M. Wegrzecki, A. Yeremin, Chemical characterization of element 112, Nature 447 (2007) 72–75.

[3] S. Dmitriev, Y. Oganessyan, V. Utyonkov, S. Shishkin, A. Eremin, Y. Lobanov, Y. Tsyganov, V. Chepygin, E. Sokol, G. Vostokin, N. Aksenov, M. Hussonnois, M. Itkis, H. Gaggeler, D. Schumann, B. H., R. Eichler, D. Shaughnessy, P. Wilk, J. Kenneally, M. Stoyer, J. Wild, Chemical identification of dubnium as a decay product of element 115 produced in the reaction $^{48}ca +^{243} am$.

[4] A. Rodin, A. Belozerov, E. Chernysheva, S. Dmitriev, A. Gylyaev, A. Gylyaeva, M. Itkis, J. Kliman, N. Kondratiev, L. Krupa, A. Novoselov, Y. Oganessian, A. Podshibyakin, V. Salamatin, I. Sivacek, S. Stepantsov, D. Vanin, V. Vedeneev, S. Yukhimchuk, C. Granja, S. Pospisil, Separation efficiency of the masha facility for short-lived mercury isotopes, Hyperfine Interact 227 (2014) 209–221.

[5] I. Kulikov, A. Algora, D. Atanasov, P. Ascher, K. Blaum, R. Cakirli, A. Herlert, W. Huang, J. Karthein, Y. Litvinov, D. Lunney, V. Manea, M. Mougeot, L. Schweikhard, A. Welker, F. Wienholtz, Masses of short-lived ^{49}sc, ^{50}sc, ^{70}as, ^{73}br and stable ^{196}hg nuclides., Nuclear Physics A 1002 (2020).

[6] A. Rodin, A. Belozerov, E. Chernysheva, S. Dmitriev, A. Gylyaev, A. Gylyaeva, M. Itkis, J. Kliman, N. Kondratiev, L. Krupa, A. Novoselov, Y. Oganessian, A. Podshibyakin, V. Salamatin, I. Sivacek, S. Stepantsov, D. Vanin, V. Vedeneev, S. Yukhimchuk, C. Granja, S. Pospisil, Masha separator on the heavy ion beam for determining masses and nuclear physical properties of isotopes of heavy and superheavy elements, Hyperfine Interact 57 (2014) 386–393.

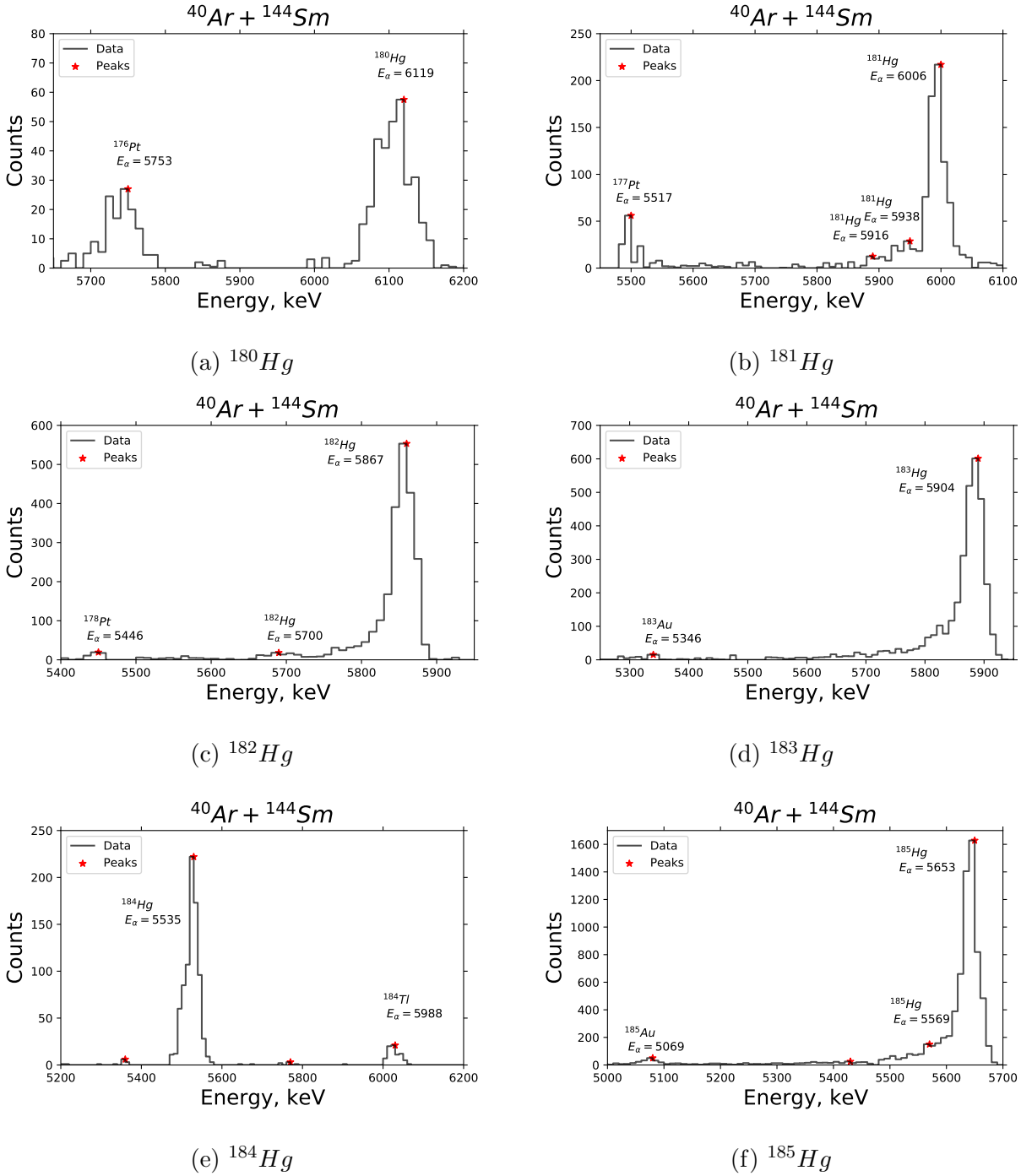
[7] V. Vedeneev, A. Rodin, L. Krupa, A. Belozerov, E. Chernysheva, S. Dmitriev, A. Gulyaev, A. Gylyaeva, D. Kamas, J. Kliman, A. Komarov, S. Motycak, A. Novoselov, V. Salamatin, S. Stepantsov, A. Podshibyakin, S. Yukhimchuk, C. Granja, S. Pospisil, Mendeleev's principle against einstein's relativity: news from the chemistry of superheavy elements, Hyperfine Interact 19 (2017) 1–14.

[8] G. A. Marzo, A comparison of different peak shapes for deconvolution of alpha-particle spectra, Nuclear Instruments and Methods in Physics Research A 832 (2016) 191–201.

[9] S. Pomme, B. Marroyo Caro, Improved peak shape fitting in alpha spectra, Applied Radiation and Isotopes 96 (2015) 148–153.

A Appendix

A.1 The peak identification and results of the fitting

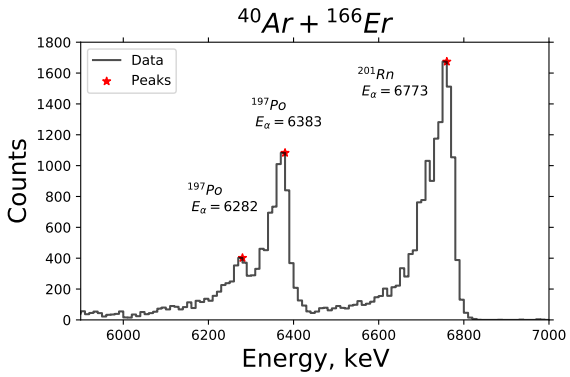


(a) ^{180}Hg (b) ^{181}Hg

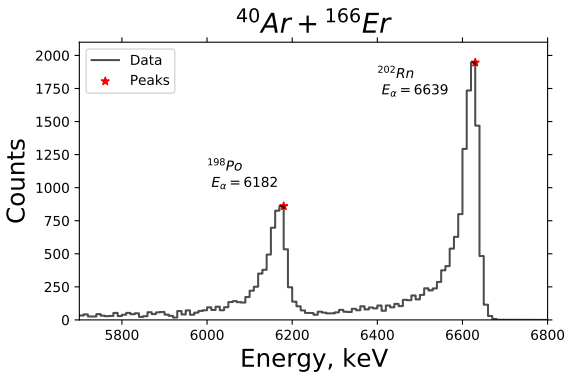
(c) ^{182}Hg (d) ^{183}Hg

(e) ^{184}Hg (f) ^{185}Hg

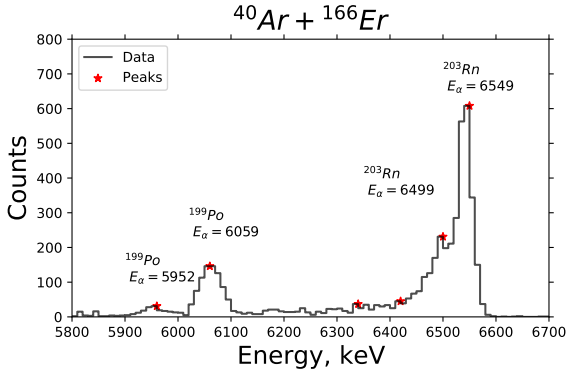
Figure 5: The energy spectrum of $^{180-185}Hg$ ions.



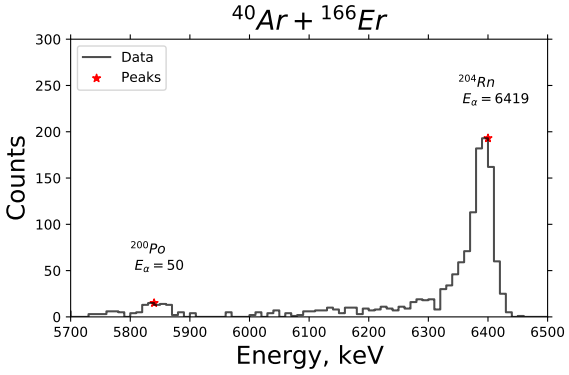
(a) ^{201}Rn



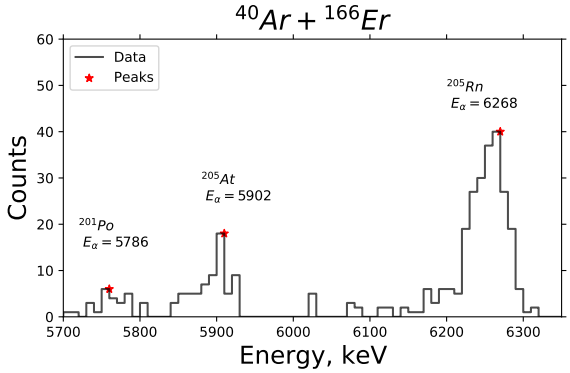
(b) ^{202}Rn



(c) ^{203}Rn

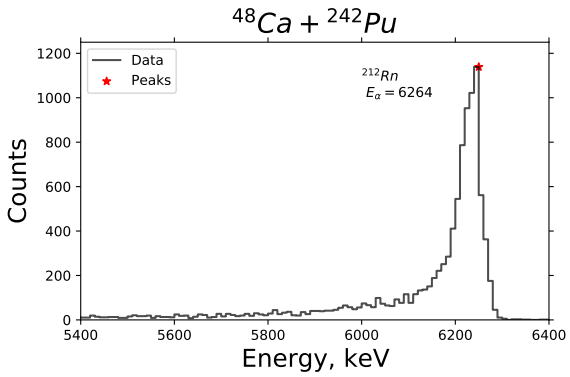


(d) ^{204}Rn

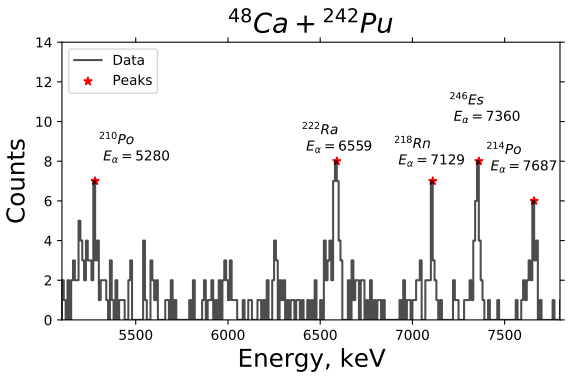


(e) ^{205}Rn

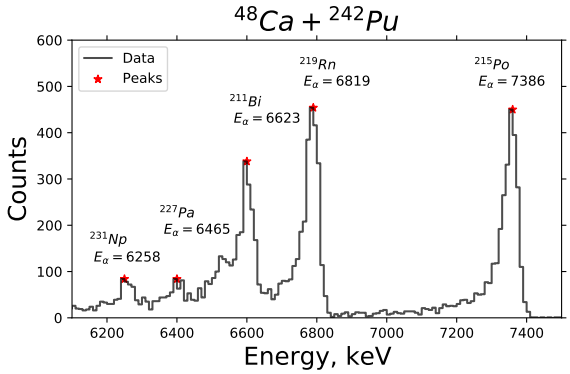
Figure 6: The energy spectrum of $^{201-205}Rn$ ions.



(a) ^{212}Rn

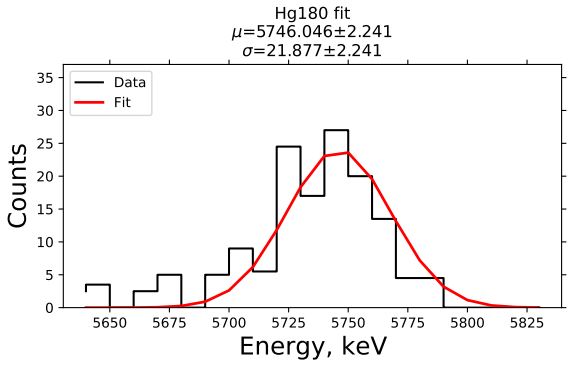


(b) ^{218}Rn

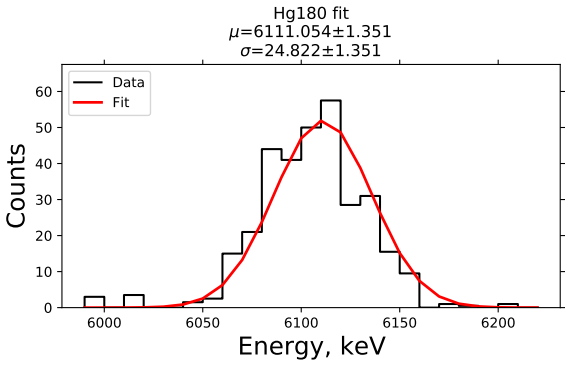


(c) ^{219}Rn

Figure 7: The energy spectrum of $^{212,218,219}Rn$ ions.

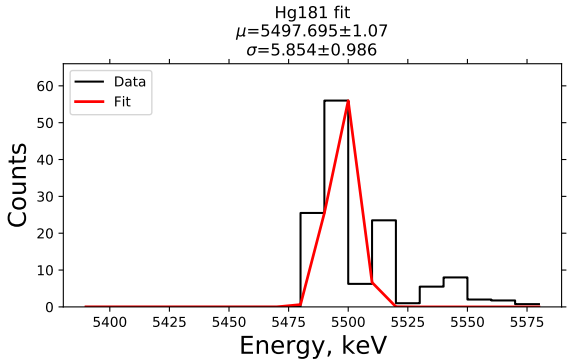


(a) ^{176}Pt

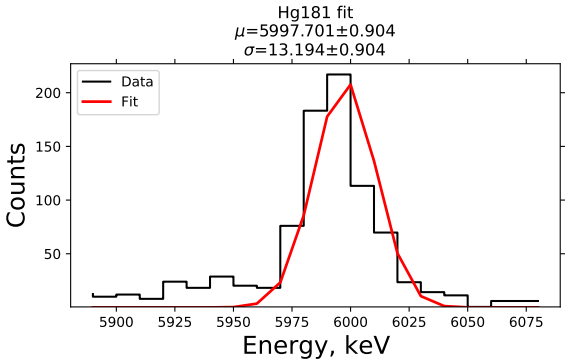


(b) ^{180}Hg

Figure 8: The fit of ^{176}Pt and ^{180}Hg ions.

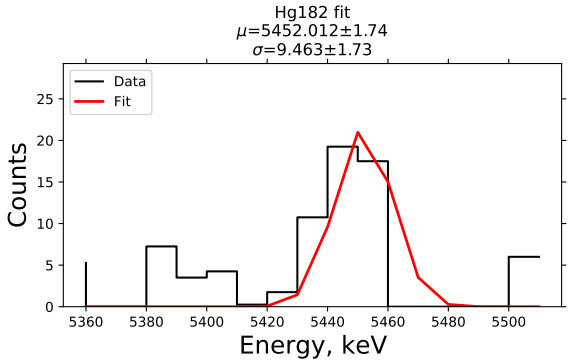


(a) ^{177}Pt

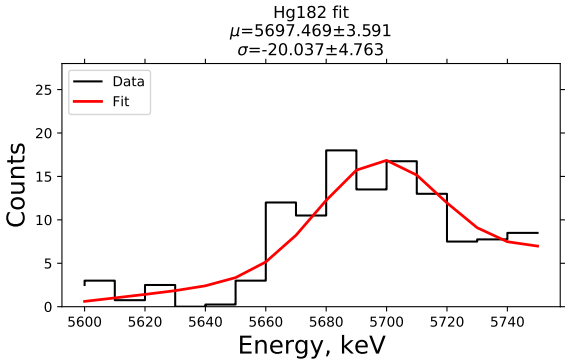


(b) ^{181}Hg

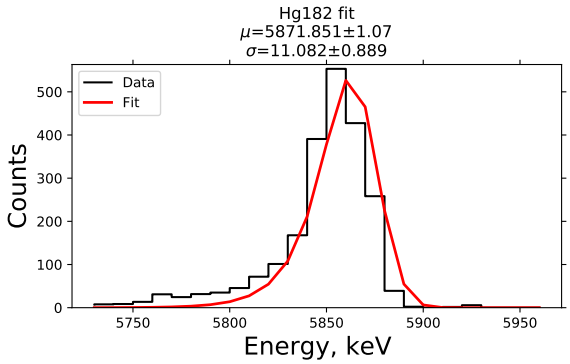
Figure 9: The fit of ^{177}Pt and ^{181}Hg ions.



(a) ^{178}Pt

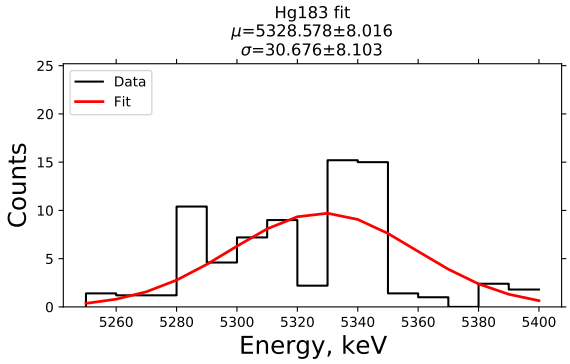


(b) ^{182}Hg

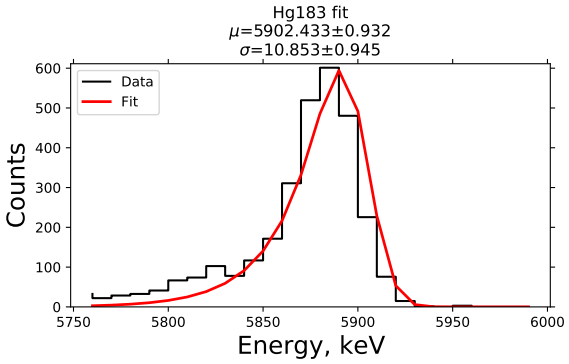


(c) ^{182}Hg

Figure 10: The fit of ^{178}Pt and ^{182}Hg ions.

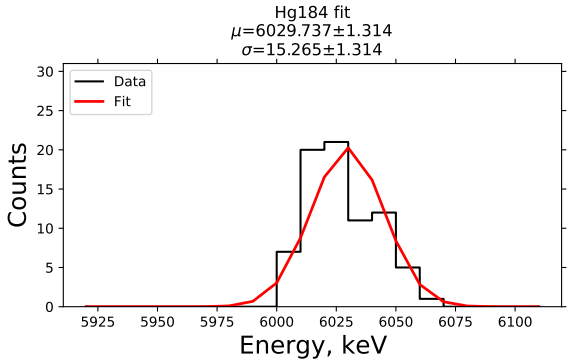


(a) ^{183}Au

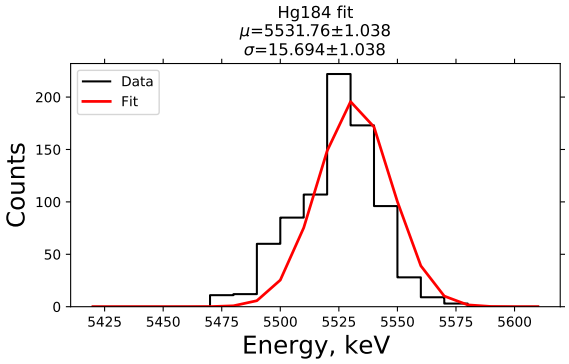


(b) ^{183}Hg

Figure 11: The fit of ^{183}Au and ^{183}Hg ions.

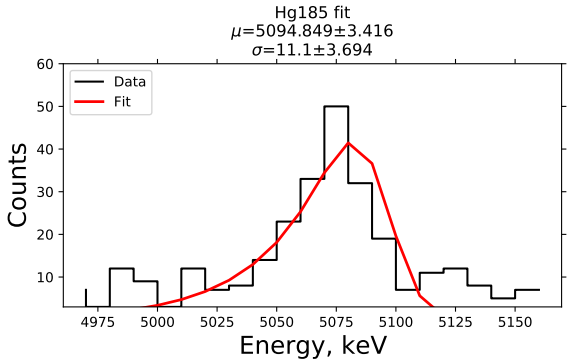


(a) ^{184}Tl

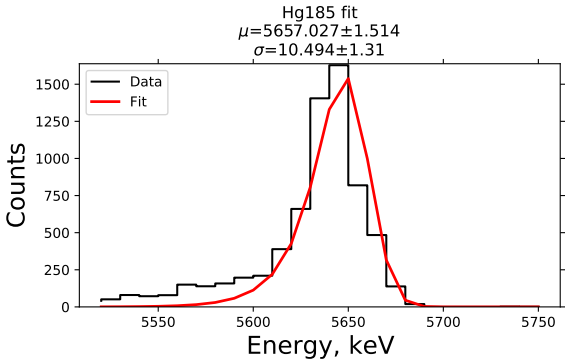


(b) ^{184}Hg

Figure 12: The fit of ^{184}Tl and ^{184}Hg ions.

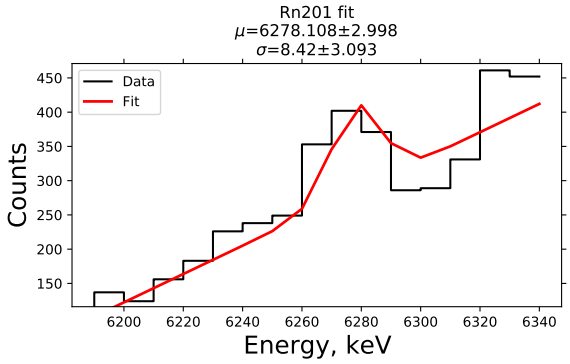


(a) ^{185}Au

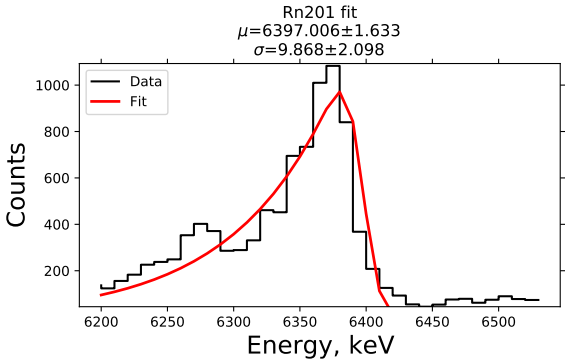


(b) ^{185}Hg

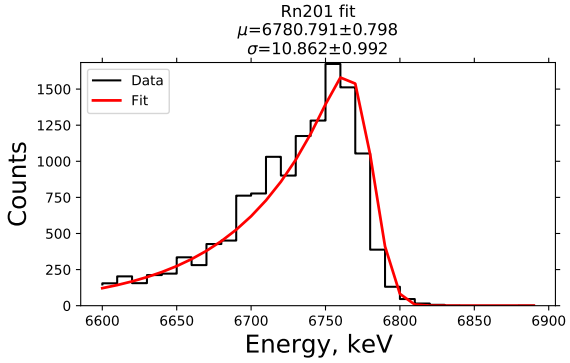
Figure 13: The fit of ^{185}Au and ^{185}Hg ions.



(a) ^{197}Po

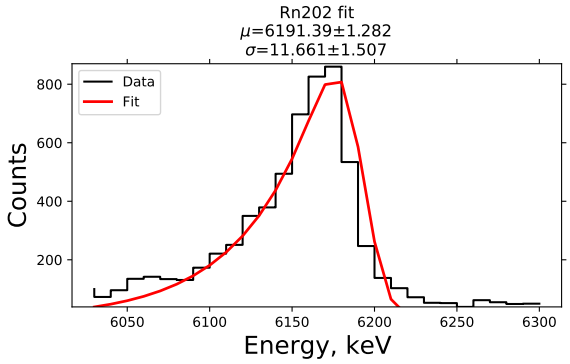


(b) ^{197}Po

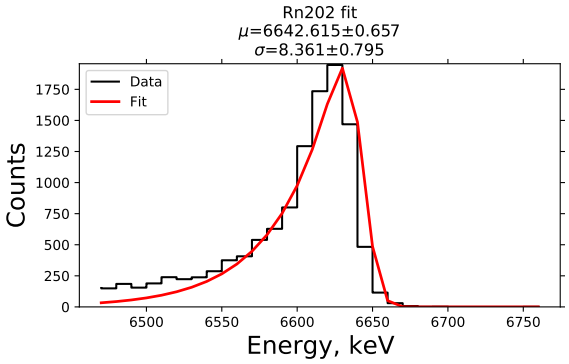


(c) ^{201}Rn

Figure 14: The fit of ^{197}Po and ^{201}Rn ions.

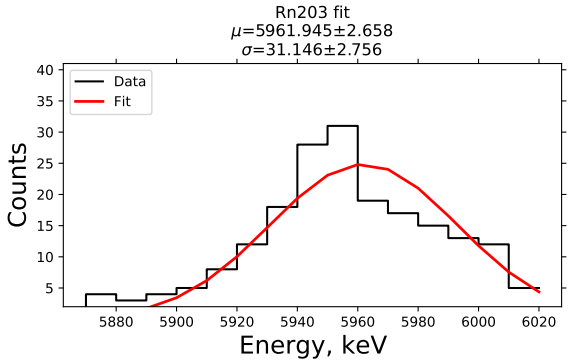


(a) ^{198}Po

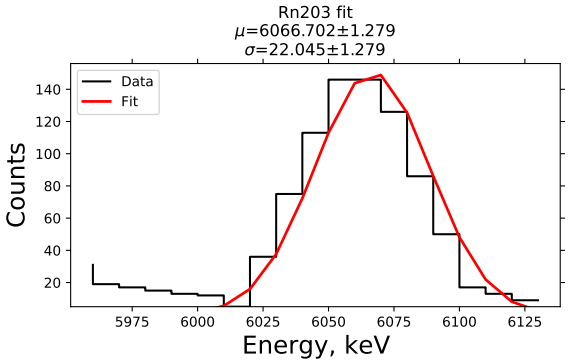


(b) ^{202}Rn

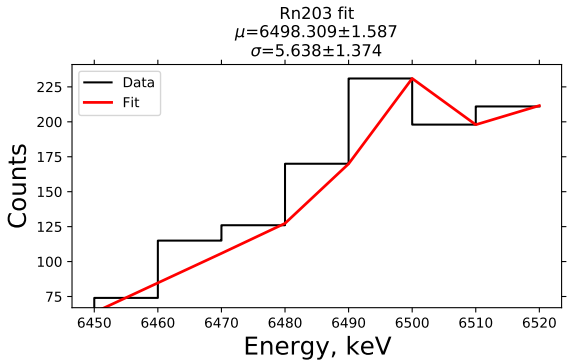
Figure 15: The fit of ^{198}Po and ^{202}Rn ions.



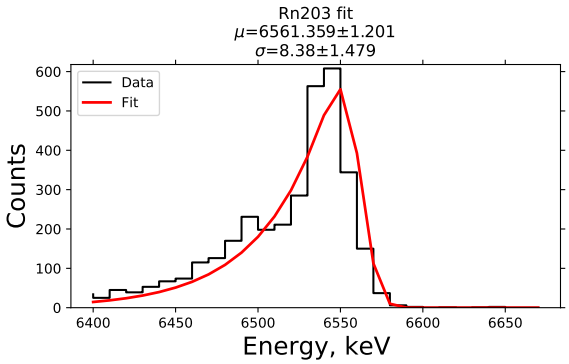
(a) ^{199}Po



(b) ^{199}Po

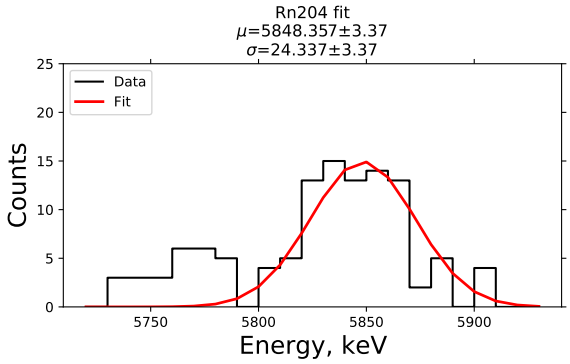


(c) ^{203}Rn

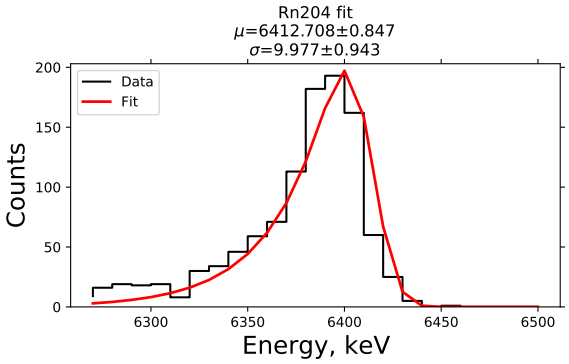


(d) ^{203}Rn

Figure 16: The fit of ^{199}Po and ^{203}Rn ions.

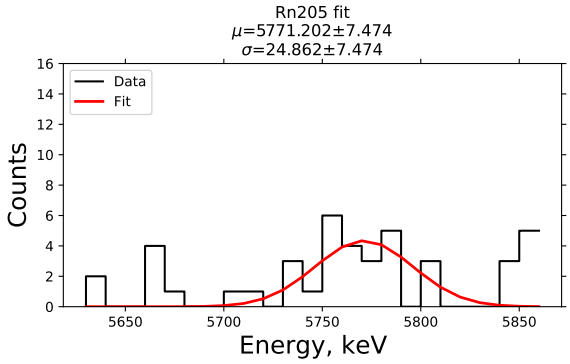


(a) ^{200}Po

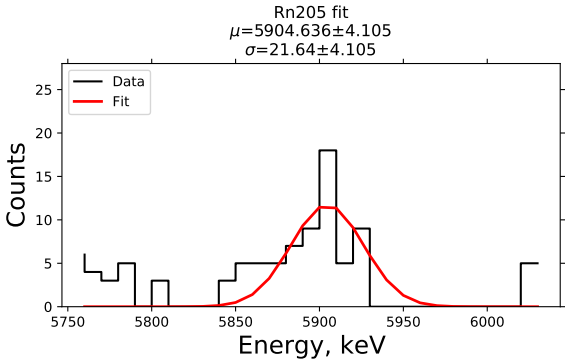


(b) ^{204}Rn

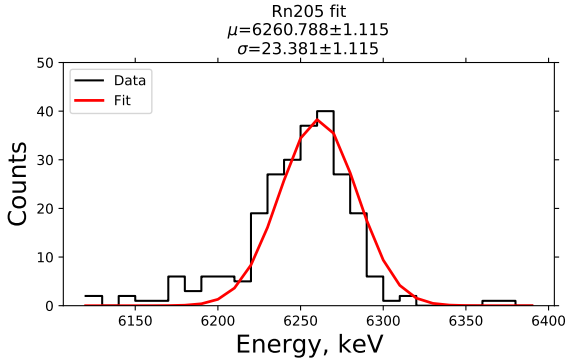
Figure 17: The fit of ^{200}Po and ^{204}Rn ions.



(a) ^{201}Po



(b) ^{205}At



(c) ^{205}Rn

Figure 18: The fit of ^{201}Po, ^{205}At, ^{205}Rn ions.

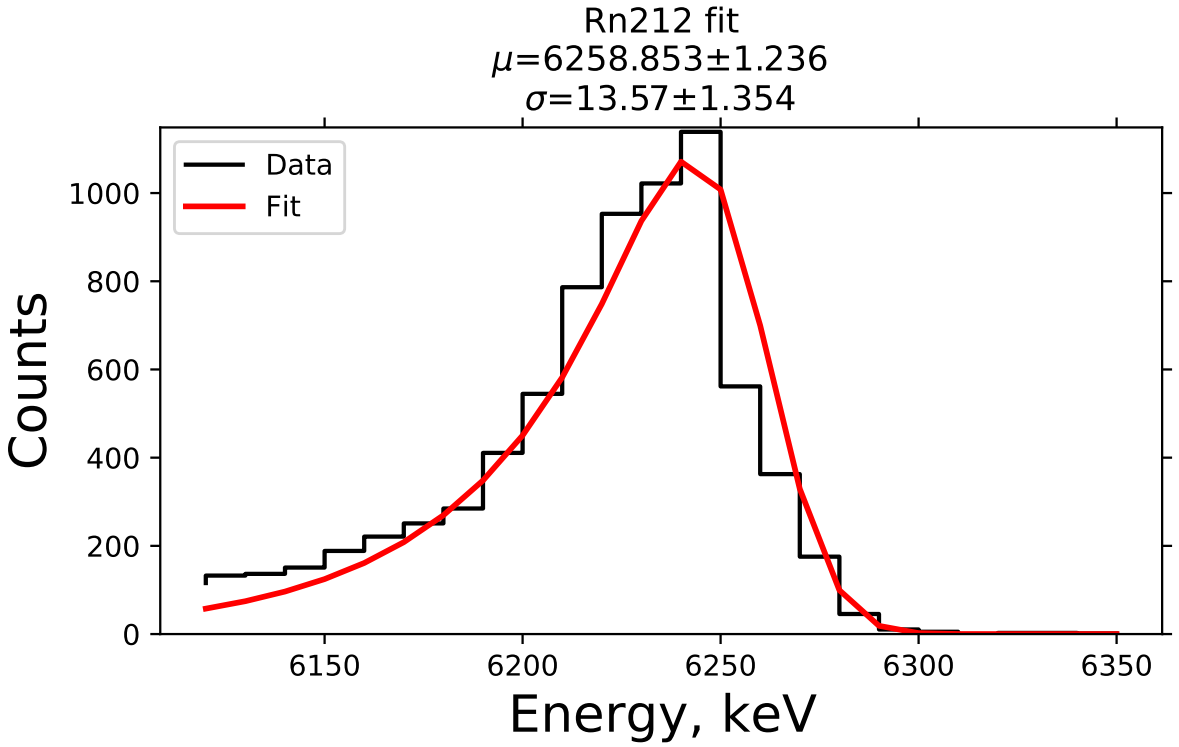
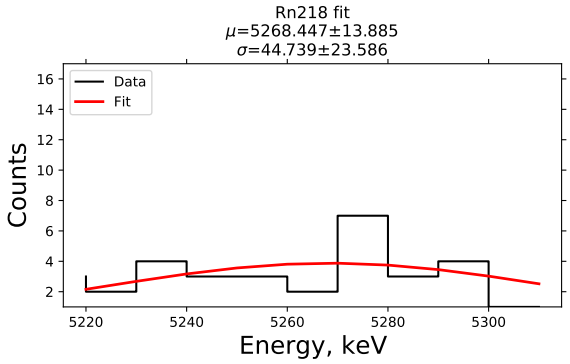
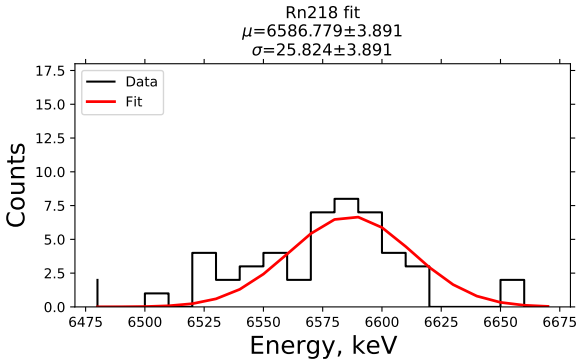


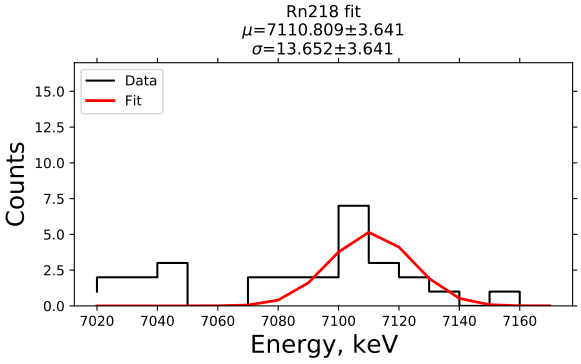
Figure 19: The fit of ^{212}Rn ions.



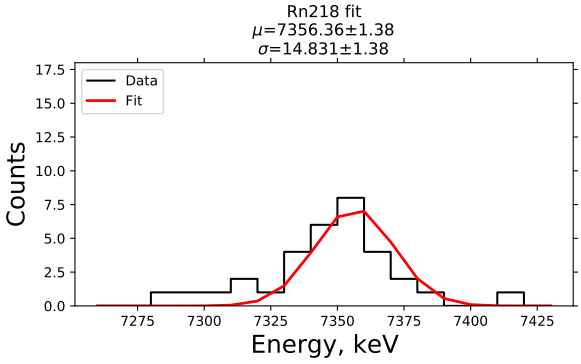
(a) ^{210}Po



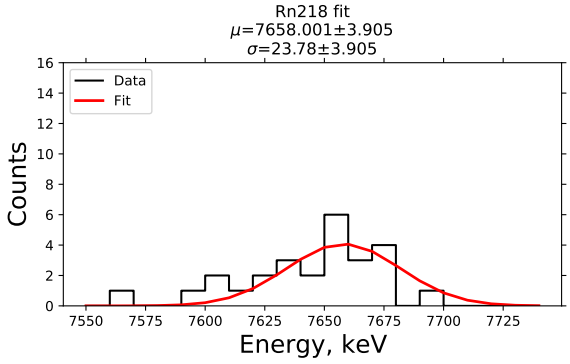
(b) ^{222}Ra



(c) ^{218}Rn

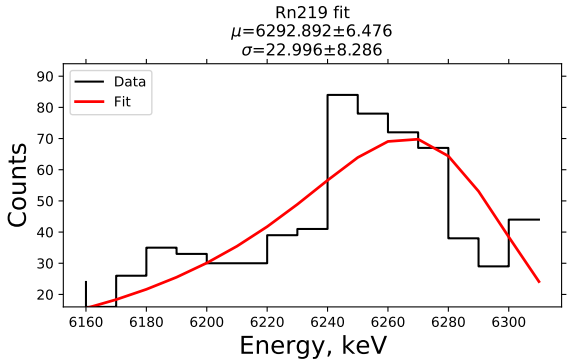


(d) ^{246}Es

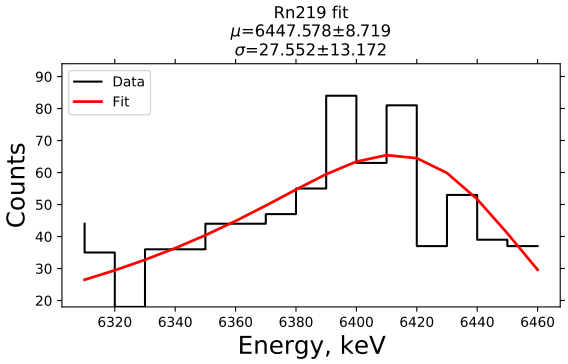


(e) ^{214}Po

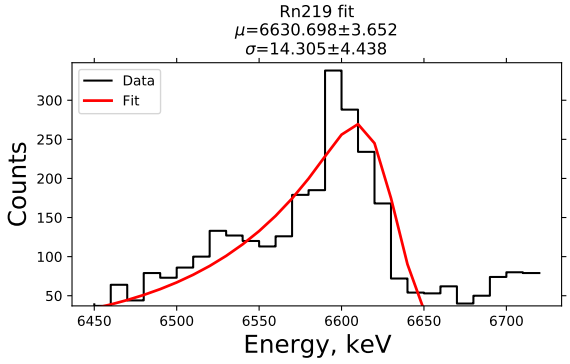
Figure 20: The fit of ^{210}Po, ^{222}Ra, ^{218}Rn, ^{246}Es, ^{214}Po ions.



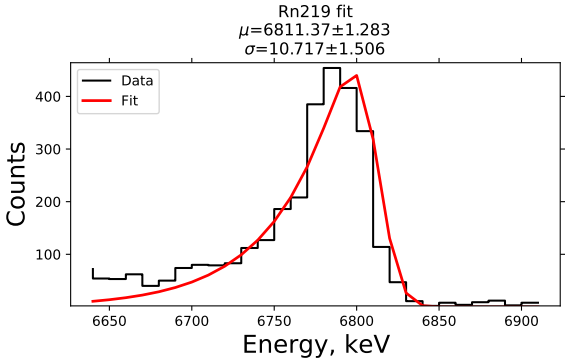
(a) ^{231}Np



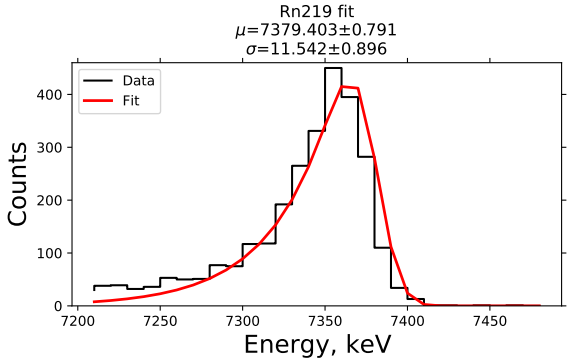
(b) ^{227}Pa



(c) ^{211}Bi



(d) ^{219}Rn



(e) ^{215}Po

Figure 21: The fit of ^{231}Np, ^{227}Pa, ^{211}Bi, ^{219}Rn, ^{215}Po ions.

A.2 Peak finder-analyzer

```
# -*- coding: utf-8 -*-
```

```
"""
Created on Tue Oct 13 21:38:39 2020

@author: Ivan
"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import peakutils
from scipy.optimize import curve_fit
from scipy import special
from matplotlib import rc
rc('font',**{'family':'sans-serif','sans-serif':['Helvetica']})

#read the data and take only the meaningfull part
file_name=input('Please enter the file name of the dat format >')
data = pd.read_csv(file_name + '.dat',sep='     ', decimal=',', float_precisi
data.columns=["energy", "counts"]
data.energy=data.energy*10
data[data<0]=0

# parameters of the peak estimator and loop of estimation

while True:
    threshold = float(input('The threshold for the peak estimation \n number
    minimum_distanse=float(input('Minimum distance between peaks  > '))

#find the peaks
    cb = np.array(data.counts)
```

```
    peaks = peakutils.indexes(cb, thres=threshold, min_dist=minimum_distanse)
    peaks= pd.DataFrame(data, index=peaks)

#plot the data
    fig, ax = plt.subplots()
    print('Energy range to plot the graph')
    x_min=float(input('x_min -> '))
    x_max=float(input('x_max -> '))
    print('Counts range to plot the graph')
    y_min=float(input('y_min -> '))
    y_max=float(input('y_max -> '))

#plt.text(5050, 150, '$^{185}Au_{5/2-}$ \n' '$E_{\u03B1}$=5069 keV')

    ax.tick_params(axis='both',direction='out', top=True,right=True)
    ax.set_title('$^{48}Ca+^{242}Pu$', fontsize=20)
    plt.step(data.energy, data.counts,c='black',alpha=0.7)
    plt.scatter(peaks.energy,peaks.counts,c='red',marker='*',alpha=1)
    ax.set_ylabel('Counts', fontsize=18)
    ax.set_xlabel('Energy, keV', fontsize=18)
    plt.ylim(y_min, y_max)
    plt.xlim(x_min,x_max )
    plt.legend(('Data', 'Peaks'), loc='upper left')
    while True:
        isotopes = input("add the name of the isope on plot? y/n -> ")
        if isotopes =="y":
            name=input("name of the isotope -> ")
            x_pos= float(input("x position on the plot -> "))
            y_pos=float(input("y position on the plot -> "))
            ener=input("alpha energy -> ")
```

```
            plt.text(x_pos, y_pos, name + '\n $E_{\u03B1}=$' + ener )
        if isotopes == "n":
            break
    saving=input('Save the plot? y/n > ')
    if saving =="y":
        plt.tight_layout()
        plt.savefig(file_name + ".pdf")
        plt.show()
    else:
        plt.show()
    cont = input("Repeat the peak finding and ploting? y/n > ")
    if cont == "n":
            break

print('Identified peaks \n')
peaks=peaks.reset_index()
peaks=peaks.drop(['index'], axis=1)
print(peaks)

#peak fitting

def line_gaus(x, amplitude, mean, stddev,slope,incr):
    return amplitude*1/(stddev*np.sqrt(2*np.pi)) * np.exp(-0.5*((x - mean)/st

def gaus(x, amplitude, mean, stddev):
        return amplitude*1/(stddev*np.sqrt(2*np.pi)) * np.exp(-0.5*((x - mean

def alpha_gaus(x, amplitude, mean, stddev,tau):
        return amplitude/(2*tau) * np.exp((x - mean)/tau+0.5*(stddev/tau)**2)
```

```
while True:
    x=int(input('The serial number of the peak to be fitted > '))
    a=float(input('How much of the data to fit? > '))
    #mu=float(input('The aproximate Mean value of the peak > '))
    #sig=float(input('The aproximate Sigma of the peak > '))
    left_gauss_bound = peaks.energy[x]-a-30
    right_gauss_bound = peaks.energy[x]+a
    #data = pd.read_csv(file_name + '.dat',sep='          ', decimal=',', float
    #data.columns=["energy", "counts"]
    #data[data<0]=0
    data=data.loc[(data['energy'] >= left_gauss_bound) & (data['energy'] <= r
    x_values_1 = np.asarray(data['energy'])
    y_values_1 = np.asarray(data['counts'])

    fit_function = input("Which function whould you like to use \n Gaussian (

    if fit_function=='g':
        p0 = [max(y_values_1),np.mean(x_values_1),np.std(x_values_1)]
        coeff, var_matrix = curve_fit(gaus, x_values_1, y_values_1, p0=p0)
        hist_gaus_fit = gaus(x_values_1, *coeff)

    if fit_function=='ag':
        tau=float(input('Any idea about tau of the peak? (roughtly equals to
        p0 = [sum(y_values_1),np.mean(x_values_1),np.std(x_values_1),tau]
        coeff, var_matrix = curve_fit(alpha_gaus, x_values_1, y_values_1, p0=
        hist_gaus_fit = alpha_gaus(x_values_1, *coeff)

    if fit_function=='lg':
        slope=(y_values_1[0]-y_values_1[len(y_values_1)-1])/(x_values_1[0]-x
        incr=y_values_1[0]-slope*x_values_1[0]
```

```
    p0 = [sum(y_values_1),np.mean(x_values_1),np.std(x_values_1),slope,in
    coeff, var_matrix = curve_fit(line_gaus, x_values_1, y_values_1, p0=p
    hist_gaus_fit = line_gaus(x_values_1, *coeff)

fig2, ax2 = plt.subplots()
plt.step(data.energy, data.counts,c='black')
plt.plot(x_values_1,hist_gaus_fit, 'red', linewidth=2)
ax2.tick_params(axis='both',direction='out', top=True,right=True)
ax2.set_title('$^{48}Ca+^{242}Pu$', fontsize=20)
ax2.set_ylabel('Counts', fontsize=18)
ax2.set_xlabel('Energy, keV', fontsize=18)
stderr_sigma_gaus = np.sqrt(var_matrix[2,2])
stderr_mu_gaus = np.sqrt(var_matrix[1,1])
a=coeff[1]
b=stderr_mu_gaus
c=coeff[2]
d=stderr_sigma_gaus
energy_and_counts=np.array(peaks.iloc[x])
peak_parameters=str([energy_and_counts,a, b])
plt.legend(('Data', 'Fit'), loc='upper left')
title=file_name+' fit ' +'\n$ \mu$='+str(round(a,3))+ '$\pm$'+str(round(b,3
ax2.set_title(title)
plt.ylim(min(data.counts), max(data.counts)+10)
saving=input('Save the Fit? y/n > ')
if saving =="y":
    plt.tight_layout()
    plt.savefig(file_name+"_Peak_"+str(x)+".pdf")
    file1 = open(file_name+"_Peak_"+str(x)+ ".txt","w")#write mode
    file1.write(peak_parameters)
    file1.close()
```

```
        plt.show()
    else:
        plt.show()
    cont = input("Repeat or fit another peak? y/n> ")
    if cont == "n":
        break
```

A.3 Heat map code

```
# -*- coding: utf-8 -*-
"""
Created on Fri Oct 16 12:50:47 2020

@author: Ivan
"""
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import peakutils
import re
import seaborn as sns
from matplotlib import rc
rc('font',**{'family':'sans-serif','sans-serif':['Helvetica']})
#read the data and take only the meaningfull part

file_name=input('Please enter the isotope name to download the matrix>')
data = pd.read_csv('Matrix_'+file_name + '.dat',sep='   ', decimal=',', float
data[data<0]=0

right_indexes=np.arange(len(data.index)-1,-1,-1)
right_indexes=right_indexes.astype(int)
```

```
data=data.set_index(right_indexes)
data=data.sort_index(axis=0)
x=[]

file_name_1=input('Please choose isotope_0 for colibration \n for example  R

with open(file_name_1 + '.txt', 'r') as f:
    first_line = f.readline()
    x=re.findall("(?<=[AZaz])?(?!\d*=)[0-9.+-]+",first_line)
peak_0=pd.DataFrame(x)

file_name_2=input('Please choose isotope_1 for colibration >')
with open(file_name_2 + '.txt', 'r') as f:
    first_line = f.readline()
    y=re.findall("(?<=[AZaz])?(?!\d*=)[0-9.+-]+",first_line)
peak_1=pd.DataFrame(y)

'''
while True:
    print('Please create the square M*M matrix for plotting, peak finding and
    x_min=float(input('x_min > '))
    x_max=float(input('x_max > '))
    y_min=float(input('y_min > '))
    y_max=float(input('y_max > '))
    x=np.arange(x_min,x_max,1)
    y=np.arange(y_min,y_max,1)
    data=data.take(x,axis=1)
    data=data.take(y,axis=0)
    cont = input("Repeat the plotting? y/n > ")
    fig1, ax1 = plt.subplots()
```

```
    ax1 = sns.heatmap(data, cmap="gist_gray_r", cbar_kws={'label': 'Counts'},
    ax1.set_yticklabels(ax1.get_yticklabels(), rotation=0)
    ax1.set_xticklabels(ax1.get_xticklabels(), rotation=0)
    for _, spine in ax1.spines.items():
        spine.set_visible(True)
    plt.show()
    if cont == "n":
            break

'''



a_list=[]
for i in range(0,len(data.index)):
    x_peaks=np.array(data.loc[i])
    threshold= 0.5
    minimum_distanse=10
    x_peaks= peakutils.indexes(x_peaks, thres=threshold, min_dist=minimum_di
    #x_peaks= pd.DataFrame(data, index=x_peaks)
    a_list.append(x_peaks)

b_list=[]
for j in range(0,len(data.columns)):
    y_peaks=np.array(data.iloc[:,j])
    threshold= 0.5
    minimum_distanse=10
    y_peaks = peakutils.indexes(y_peaks, thres=threshold, min_dist=minimum_d
    b_list.append(y_peaks)
```

```
df_x = pd.DataFrame(a_list)
df_y = pd.DataFrame(b_list)
df_x=df_x.reset_index()
df_y=df_y.reset_index()
df_x=df_x.melt(id_vars=["index"]).dropna(axis=0).sort_values(by=['index'])
df_y=df_y.melt(id_vars=["index"]).dropna(axis=0).sort_values(by=['index'])
df_x = df_x.rename(columns={'index': 'row', 'value': 'col'})
df_y = df_y.rename(columns={'index': 'col', 'value': 'row'})
df_x=df_x.reset_index()
df_y=df_y.reset_index()
df_x=df_x.drop(['index'], axis=1)
df_y=df_y.drop(['index'], axis=1)
df_x=df_x.drop(['variable'], axis=1)
df_y=df_y.drop(['variable'], axis=1)

columnsTitles=["row","col"]
df_y=df_y.reindex(columns=columnsTitles)

final_list=[]
for j in range(0,len(df_x)):
    a=np.array(df_x.loc[j])
    for i in range(0,len(df_y)):
        b=np.array(df_y.loc[i])
        equal_arrays=(a==b).all()
        if equal_arrays == True:
            final_list.append(b)

Final_peaks = pd.DataFrame(final_list)
Final_peaks.columns=["row", "col"]
row=np.array(Final_peaks.row)
```

```
col=np.array(Final_peaks.col)
counts=data.iloc[row, col]
#counts.columns = counts.columns.map(str)

counts=counts.groupby(counts.columns, axis=1).sum()
counts=counts.sort_index(axis=1)
counts = counts.groupby(level=0).last()

'''
while True:
    strip_1=int(input('Please inter the aproximate strip number of peak_0 re
    strip_2=int(input('Please inter the aproximate strip number of peak_1 re
    b=np.array(counts.columns)
    #equal_arrays_1=(strip_1==b).all()
    #equal_arrays_2=(strip_2==b).all()
    #print(strip_1)
    #print(strip_2)
    if strip_1 in b and strip_2 in b:
        y1=float(peak_0.loc[2])#counts.loc[:,strip_1].idxmax(axis=1)
        y2=float(peak_1.loc[2])#counts.loc[:,strip_2].idxmax(axis=1)
        x1=strip_1
        x2=strip_2
        a1=(y2-y1)/(x2-x1)
        b1=y1-a1*x1
        print(x1,y1)
        print(x2,y2)
        print(a1,b1)
        print("colibration finished successfully")
    else:
        if not strip_1 in b:
```

```
            print("No such string found for peak_0")
        if not strip_2 in b:
            print("No such string found for peak_1")
    cont = input("Repeat the colibration? y/n > ")
    if cont == "n":
        break
'''
y1=float(peak_0.loc[3])#counts.loc[:,strip_1].idxmax(axis=1)
y2=float(peak_1.loc[3])#counts.loc[:,strip_2].idxmax(axis=1)
a1=(y2-y1)/14
#first_element=y1-a1*98
element_0=y1-a1*53
element_1=element_0+a1*len(data.index)
colibration=np.arange(element_0,element_1,a1)
#colibration=a1*colibration
colibration=colibration.astype(int)
data=data.set_index(colibration)



while True:
    fig, ax = plt.subplots()
#data = data.pivot("Energy,Mev", "Strip number", "counts")
    ax = sns.heatmap(data, cmap="gist_gray_r", cbar_kws={'label': 'Counts'})
    ax.set_yticklabels(ax.get_yticklabels(), rotation=0)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=0)
    for _, spine in ax.spines.items():
        spine.set_visible(True)

    print('Strip range to plot the graph')
    x_min=float(input('x_min > '))
```

```
    x_max=float(input('x_max -> '))

    print('Energy range to plot the graph (from 0 until lenght of the matrix)
    y_min=float(input('y_min -> '))
    y_max=float(input('y_max -> '))
    plt.ylim(y_max, y_min)

    plt.xlim(x_min, x_max)

    plt.xlabel("Strip number", fontsize=18)
    plt.ylabel("Energy, kev",fontsize=18)
    ax.set_title("$^{48}Ca+^{242}Pu$", fontsize=20)
    while True:
        isotopes = input("add the name of the isotope on plot? y/n -> ")
        if isotopes =="y":
            name=input("name of the isotope -> ")
            x_pos= float(input("x position on the plot -> "))
            y_pos=float(input("y position on the plot -> "))
            plt.text(x_pos, y_pos, name)
        if isotopes == "n":
            break
    plt.tight_layout()
    plt.savefig("Heat_map_"+ file_name + ".pdf")
    plt.show()
    cont = input("Repeat the plotting? y/n -> ")
    if cont == "n":
            break
#del x,y,file_name,file_name_1,file_name_2,first_line,threshold,minimum_dist
```