10 March 2023

Plots of Xray – Compton experiment

1 IN-LAB PLOT LIST.

You should do the calibration again if needed.

- You need to plot every spectrum you record according to the instructions.
- And you need to prepare a channel – energy calibration plot. And do a linear regression.
- Then you need to present a spectrum you record earlier as (Energy, Counts).

Then

- You need to present the $(\cos(\theta)\ ,\ \Delta\lambda)$. What would be the slope? Look at the Compton formula.
- Plot the amplitude of the shifted line as function of the angle. Compare it with Klein-Nishina formula.

See next page ↓

2 YOU CAN USE THE FOLLOWING SAMPLE.

To use this sample, you need to understand every line of it.

2.1 THE FUNCTIONS YOU WILL USE.

```
import pandas as pd # read the data from files
import matplotlib.pyplot as plt # plot the data
import numpy as np # just for fun :)
from scipy.signal import find_peaks # find the first estimate to a
line.
from scipy.stats import linregress # for calibration and compton fit
from scipy.optimize import curve_fit # for gausian fit
import scipy.constants as const # physical constants.

#Define the Gaussian function
def Gauss(x, H, A, x0, sigma):
     return H + A * np.exp(-(x - x0) ** 2 / (2 * sigma ** 2))

# Mean over near chanels.
def smooth(y, box_pts):
    box = np.ones(box_pts)/box_pts
    y_smooth = np.convolve(y, box, mode='same')
    return y_smooth

# just linear function
def lin(x,a,b):
    return a*x+b

# convertion of energies in eV to wavelength in m
def e2lam(e):
    hc=1.24*10**-6; # eV*m
    return hc/e

# compton shift in whavelength
def comp(theo_angles):
    h=6.626*10**-34 # eV*s
    c=2.9*10**8; # m/s
    me=9.1*10**-31; # kg
    return h/(me*c)*(1-np.cos(theo_angles))+e2lam(17479)

# The Klein-Nishina formula to energy 17479 eV
def klein_nish(A,theo_angles):
    E0=17479 # eV . The line energy without the Plexiglas
    lam0=e2lam(E0);
    return
```

```
1/2*A**2*(lam0/comp(theo_angles))**2*(lam0/comp(theo_angles)+comp(theo
_angles)/lam0-np.sin(theo_angles)**2)
```

2.2 THE LOOP

```
# Define the line energy and amplitudes
comp_amp=[]
comp_eng=[]

# Define 14 colors
colors = ['red', 'blue', 'green', 'yellow', 'orange', 'purple',
'pink', 'brown', 'black', 'gray', 'cyan', 'magenta', 'lime', 'navy']

# The loop run on different angles
plt.figure(dpi=300)
for i in range(1,15):
    # import the data
    data=pd.read_csv('run{}'.format(i), sep='\t', header=1)
    chanals=np.array(data['Channel/#'])
    Impulses=np.array(data['Impulses/#'])
    Imp_smooth=smooth(Impulses,50)

    # cut relevant interval
    x=conv(chanals);
    y=Imp_smooth;
    indss = (x>16000) & (x<18500)
    x=x[indss];
    y=y[indss];

    # plot the relevant interval
    plt.plot(x,y,':',color=colors[i-1])

    # first estimate the line energy
    peaks, properties = find_peaks(y, prominence=5,
width=20,distance=1000)
    plt.plot(x[peaks], y[peaks], "rx") # plot the estimation

    # Fit the line to gaussian. p0 is the initial guess
    parameters, covariance = curve_fit(Gauss, x,
y,p0=[10,y[peaks].item(),x[peaks].item(), 500]);

plt.plot(x,Gauss(x,parameters[0],parameters[1],parameters[2],parameter
s[3]),'--',color=colors[i-1])

    #acumulate the line energies and amplitudes
comp_amp.append(Gauss(parameters[2],parameters[0],parameters[1],parame
ters[2],parameters[3]))
    comp_eng.append(parameters[2]) # eV
```

```
plt.xlabel(r'Energy [eV]')
plt.ylabel(r'Intensity [au]')
```