Python Exam Spring 2023 Guide

In this exam, we analyze data from Compton Scattering experiment in order to determine the electron mass. The detailed experimental guide can be found here.

A. Introduction to the physics background:
If you wish to know more of the experiment, you can visit https://wanda.fiu.edu/boeglinw/courses/Modern_lab_manual3/scint_compton.html . But the summary below is all you need to proceed to the exam.

In Compton scattering, photons (with energy E_0) can scatter off an electron at any angle θ between 0 to 180 degrees. Depending on the scattering angle θ, the energy of the photons E_f is given by

Eqn 1: $\frac{E_0}{E_f} = 1 + \frac{E_0}{M}(1 - cos\theta)$

Here we assume that speed of light c=1 in natural unit system. In this experiment, the photon is from the $^{137}_{55}C_s$ (Cesium) gamma decay that has a known energy $E_0 = 0.6617MeV$. The electrons that the photons scatter off is from an Aluminum target.

Once the photons scatter off the electrons, their energies can be detected using scintillators and PMT's because it can give its energy via photoelectric effect to the electrons in the scintillators. As a result, these electrons start to move in the scintillator, they produce light which can be turned into measurable electric signals using the PMT. However, the PMT doesn't measure the energies directly. It provides an electric signal that gets proportionally turned into a digital signal. All signals are then record edas a histogram. Therefore, the first step we need to perform is to know the proportionality of the digital signal to the photon energies that was given to the electrons in the scintillator. This process is called calibration. This is done by placing three separate known gamma decay sources (0.6617MeV for $^{137}_{55}C_s$, 1.1732MeV and 1.3325MeV for the $^{60}_{27}C_s$) directly on top of the scintillators, the resulting digital spectrum will be recorded as data files ending with .Spe extensions. It can be converted to histograms (x-axis being the digital channel numbers). This histogram will then have three different peaks, each of them corresponding to one of the known photon energies. You can use **Gaussian fits to determine the peak positions**, and then **use straight line fit to determine how to convert the Channel numbers to energies**.

Once this calibration process is done, we can proceed to measure the scattered photon energies at different scattering angels. For each angle, two spectra will be

recorded. One spectrum with the target (it will be self-obvious in the file names), let's call it ht, another spectrum(we can call it hempty) will be recorded without the target (the file names will contain the word "empty"). The difference between the two histograms (ht-hempty) will give you the spectrum that resulted from the Compton scattering. It should contain a peak (plus some background) which you need to perform gaussian fits on to determine the photon energy. Once all these energies are determined, you can determine the electron mass by performing a linear fit on an appropriate graph.

Exam (total 282 points + 60 possible extra points)

Before starting the exam, download all files that ends with .Spe extension. You can also download the zip file that has all the files already in them and then unzip them in an appropriate directory for this exam on your computer.

0. import numpy and LT.box (2pt)
i. Calibration
1. Create an array (with three elements), with the name "Es", that contains the three known gamma decay energies of Cs, and Co; **(5pt)**

2. Create an empty array (also size 3), with the name "peaks" **(5pt)**

Note 1. All files that end with ".Spe" can be read into a histogram by using the function B.get_spectrum(), for example, **h1=B.get_spectrum('spectrum1.Spe') would read the data contained in spectrum1.Spe, and store the information** in h1 which is a histogram. You can plot this histogram by doing: **h1.plot()**. Histograms can also be fit with a default Gaussian function. For example **h1.fit(1.0, 1.5)** would fit the histogram h1 with a gaussian function, in the x-variable range between 1.0 and 1.5. If you want a specific color (e.g. red) to draw the fit, you can do **h1.fit(1.0, 1.5, 'color=red')** . If the fit is not shown automatically, you can use **h1.plot_fit(**). You can access the fitting parameters value and uncertainty directly from the histogram h1. For example, if you need to know the mean value, **h1.mean.value** is what you need. If you need to know the uncertainty of the mean, **h1.mean.err** is what you need.

3. Read the calibration.Spe file and store the information in the histogram called hcal (See Note 1 above). **(5pt)**

4. Show hcal in a graph. Select an appropriate range and perform a Gaussian fit on the first peak in hcal. **(10pt)**

5. obtain the mean value of the fit from the histogram, and store it as the first element of array peaks, which you created in item 2. Do not copy paste the values from the python console. **(5pt)**

6. Repeat 4 and 5 for the second and third peaks, but store the mean value to the second and third element of the array peaks. **(20pt)**
(If item 4-6 was performed in a loop, you get 10 extra points; Remember if you were to use loop here, you need to define additional arrays that's not specified by the guide)

7 Make a new graph with the energies Versus peaks, and label y and x-axis with proper names and units. **(10pt)**

8. Perform a linear fit on this the graph, and store it in fitcal. The fit line needs to be drawn on the graph. **(10pt)**

Note 2: This "fitcal" is what you need to apply to the other data files with .Spe extensions to convert them from digital channel spectrum to energy spectrum. For example, the following code
hcalnew = B.get_spectrum(**'calibration.Spe'**, calibration=**fitcal**)
would convert original digital spectrum from the file **calibration.Spe**, and converts it to energy spectrum using the linear fit stored in **fitca**l, and the resulting histogram is stored in **hcalne**w.

9. Verify that your calibration it is done correctly by plotting the hcalnew histogram (described in Note 2) in a new figure. You should see that the new histogram has exactly the same shape as the original histogram, but the x-axis range is now changed as expected. **(5pt)**

10. Set the x- axis title to 'Energies (MeV)') **(5pt)**
(Item 1-10 checklist: three graphs, totaling 80points.)

ii. **Apply Calibration to other data files measured at four different scattering angles (Refer to Note 2)**

11. Create a **four member array** with the name "**degrees"**, the four values are 30, 40, 60, 80 **(5pt)**
12. Convert the values in the array called "degrees" to radians, and store the values in another array called "**thetas**" **(5pt)**

13. Create a **four member empty array** with the name "**efs"**, to store the energies you will be determining in the following steps. **(5pt)**

14. Create a **four member empty array** with the name "**efsErr"**, to store the uncertainties of the energies you will be determining in the following steps. **(5pt)**

15. Apply calibration on the spectrum stored in the file "30_deg_target.Spe", then store the result in a histogram called htarget. **(5pt)**

16. Apply calibration on the spectrum stored in "30_deg_empty.Spe", then store the result in a histogram called hempty. **(5pt)**

17. Make a new graph that shows both htarget and hempty. **(5pt)**
18. Store the htarget-hempty in a new histogram called hcompton, and show it in a new graph. Label y-axis as "counts", and x-axis as "Energies (MeV)" (5pt) **(5pt)**

19. Inspect the graph in item 15, and choose an appropriate range to perform a Gaussian fit on hcompton. The fits need to be visible on your graph. **(10pt)**

20. Obtain the peak position (mean) from hcompton, and store it as the first array member in the array efs **(5pt)**

21. Obtain the uncertainty of the peak position from hcompton, and store it as the first array member in the array efsErr **(5pt)**
(So far you should have produced two more graphs, totaling **60 points so far for item 11-21.**)

22. Repeat the same steps from item 15 to item 21, for the data at 40, 60, and 80 degrees. If you need to define a bunch of new variable names, it's OK. But in the end, the peak positions and their uncertainties need to be store in the two arrays **efs** and **efsEr**r in the correct order. **(60pt)**
(**There are 6 more graphs for 40, 60,80 degrees**, and If you performed the items from **15 to item 22 in a loop, you get 30 extra points**)

(Item 11-22 checklist: 8 graphs, 4 Gaussian fits totaling 120points + 30 possible extra points.)

iii. Determine the electron mass

23. Compute E_0/efs, E_0 should be obtained from the array Es in item 1. The result should be stored in an array called "**eratio**" **(5pt)**
24. Make a new graph of eratio Versus cos(thetas), lable the axis appropriately. **(5pt)**
25. Perform a straight line fit to the graph above, store the fit object to fit2. **(10pt)**
26. Obtain the slope from the fit2, assign it to the variable **eslope**. **(5pt)**
27 obtain the slope uncertainty from fit2, and assign the value to the variable **dslope (5pt)**
28. Obtain the offset from the fit2, assign it to the variable **yoff**. **(5pt)**
29 obtain the offset uncertainty from fit2, and assign the value to the variable **dyoff (5pt)**
30 compute the value of electron mass, in the unit of MeV (assuming c=1 in natural unit system), according to Equation **(10pt)**
31. Compute the uncertainty of the electron mass, in the unit of MeV, according to Equation **(10pt)**
32. Print a statement that shows both the electron mass and its uncertainty, in proper format, and number of significant digits, in the graph **(10pt)**
33. Print a statement that shows both the offset and its uncertainty from fit 2, in proper format, and number of significant digits, in the graph**. The statement should include what the expected offset should be (10pt)**

(Item 1-10 checklist: three graphs, 3 Gaussian fits + 1 linear fit totaling 80points + 10 possible extra points.)

(Item 11-22 checklist: 8 graphs, 4 Gaussian fits totaling 120points + 30 possible extra points.)

(Item 23-33 should include one graph, one linear fit, and 80points)

Bonus (20pt):

34. All the Gaussian fits above in the guide did not include any background function. You could improve you results by include a some background function. For example, **h1.set_fit_list(fit = ['A', 'mean', 'sigma', 'b0', 'b1', 'b2'])** will fit the histogram that is the sum of a Gaussian function and a second order polynomial function. Then you can proceed to perform the Gaussian fits, etc. If you did this instead, you should compare your electron mass results with the different fits.