Gamma 3 Activity: Half-life analysis of thoron source gamma peak

Teal Pershing

August 12, 2019

Abstract

This document provides the overview and procedure for teaching the thoron-focused Gamma 3 Activity in the UC Davis Nuclear Analytical Techniques (NAT) Summer School. Students will use a Python package to analyze a single gamma peak over many HPGe counting datasets. The analysis will yield the approximate half-life of the isotope associated with the gamma peak. Students will then perform a chi-squared test by hand to test the goodness of fit.

1 Introduction to activity

The group will use the HPGe data collected from the thoriated fluid for a half-life analysis. The analysis will be completed using a Python package that can be found here:

`https://github.com/pershint/NAT_HLActivity.git`

For information on each directory and class used in the main script, see README.md at this git repository.

2 Making your data usable in Python

Load your .spe format HPGe data files into a directory in /data/. Using your favorite text editor, open one of the .spe files.

QUIZ TIME Is this data format easily loaded/ used by Python?

Go to the /utils/ directory and open up spe2npz.py and look at how .spe files are converted to .npz format files. Then, run

```
python spe2npz.py ../data/yourdatadir/*
```

This will convert your data into a Python friendly format: the npz file.

NOTE: There are plenty of other data formats out there you could convert to (another option I considered was JSON format). A different choice of data format would just mean writing a different data conversion tool (spe2json.py for my alternative) and possibly some different logic to load the files with the datafile and datastruct classes in lib/datastruct.py.

When you do this, you'll have to choose based on what you think is best for your application. Let's stick with managing the npz data for this activity.

3 Run your half-life analysis

We will now run the main script, which fits the counts per second as a function of time to the exponential expression:

$$E(C, \lambda, t) = Ce^{-\lambda t}$$

Where C is the counts per second expected collecting a data set starting at $t = 0$, t is the time since counting started, and λ is the decay constant for the isotope associated with the gamma peak.

The function will output the best fit initial counts per second for your sample, as well as the best fit half life.

QUIZ TIME: How is the half-life related to the decay constant term λ?

Before running the script, return back to the home directory and open main.py in your favorite text editor. You need to point the variable $DATADIR$ to where your npz data files are stored. Change the location as needed and close the text editor.

In terminal, run the main script by typing:

```
python main.py --debug
```

3.1 Questions

a. How do you best choose the peak region? How would you go about automating this process?

b. How is the background subtracted from the peak region?

c. How are the uncertainties calculated for each background subtracted data point?

d. CHALLENGE: Can you think of other ways to solve for the half life? What if you only have two data points?

4 The curve fit algorithm

The best fit function to the data is found with in this script using the scipy optimize library's function "curve_fit". Given some initial values, the function will calculate the following classifier:

$$\chi^2 = \sum_{i=1}^{N} \frac{(O_i - E(C, \lambda, t_i))^2}{\sigma_E^2}$$

Where O_i is the average counts per second observed for the run at time t_i.

Technically, this expression is not actually a "chi-squared" distribution unless the data come from a gaussian distribution in each bin. Since this often isn't the case, we refer to it as the "least squares" distribution to be pedantically correct.

For more information on the Chi Square Distribution, and how you could relate it to the goodness of your fit, here's a good "cheat sheet" resource:

`https://www.physics.ohio-state.edu/~gan/teaching/spring04/Chapter6.pdf`

The program adjusts the fit parameters a bit and re-calculates the term. Then it does it again. And again. Eventually, when the method has found the lowest χ^2 term (which correspod to the "best fit" parameters), the function returns these "best fit" parameters and a covariance matrix.

4.1 Questions

a. Why use this expression? Are there any other metrics we can use?

b. Any ideas for terms we could add to constrain/weight the fit?

5 Uncertainties on the fit

The uncertainty on the best fit parameters is important to quantify when reporting a fit to data and propagating uncertainties to any calculation that will use the best fit parameters. The uncertainties in the fit and fit parameters will depend on the spread in the data points, as well as the statistical uncertainty for each data bin.

In the output from the main script, notice that a covariance matrix is output. The covariance matrix is calculated by the algorithm and provides a measure on the uncertainties for the fit parameters. In our case, the elements of the covariance matrix are:

$$\begin{bmatrix} \sigma_C^2 & \sigma_C\sigma_\lambda \\ \sigma_\lambda\sigma_C & \sigma_\lambda^2 \end{bmatrix}$$

Where each sigma is the standard uncertainty on it's related parameter.

5.1 Questions

a. What is the difference between a statistical and systematic uncertainty?

b. What kind of systematic uncertainties could we have to account for here?