PHYS 705.01 - Fall 2020

Homework 03

Landon Buell

21 September 2020

In [1]:
```
#### IMPORTS ####

import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as opt
```

Question 1

The table lists the channel numbers x and Δx, obtained by fitting Gaussiands to photopeaks of known energues E.

In [2]:
```
#### RAW DATA PROVIDED ####

x = np.array([354.588,646.387,1055.096,716.784,810.951,413.66,519.326,323.637,779.143])
dx = np.array([0.039,0.035,0.403,0.129,0.096,0.036,0.042,0.073,0.068])
E = np.array([569.70,1063.66,1770.24,1173.24,1332.50,661.66,834.86,511.02,1274.54])
```

Part A)

Fit a straight line, s.t. $E = A + Bx$ to this data set. What are the best values for the paramaters A and B?

In [3]:
```
class LinearRegressor :
    """ Linear Regression class """

    def __init__(self,x,y):
        """ Intialize LinearRegressor Instance """
        self.x = x
        self.y = y
        self.avgX = np.mean(self.x)
        self.avgY = np.mean(self.y)

    def ComputeSlope(self):
        """ Compute Degree 1 Coefficient """
        _a = np.sum((self.x - self.avgX) * (self.y - self.avgY))
        _b = np.sum((self.x - self.avgX)**2)
        self.slope = _a/_b
        return self.slope

    def ComputeIntercept(self):
        """ Compute Degree 0 Coefficient """
        self.intercept = self.avgY - (self.slope * self.avgX)
        return self.intercept
```

In [4]:
```
# Use Regression Class Above
```

```
EnergyFit = LinearRegressor(x=x,y=E)
B = EnergyFit.ComputeSlope()
A = EnergyFit.ComputeIntercept()

# Print Results
print("Best value for A =",A)
print("Best value for B =",B)
```

```
Best value for A = -44.337084200344066
Best value for B = 1.7066164038476765
```

In [5]:
```
# Compute Uncertainty
dx_avg = np.mean(dx)
rad = 0
for i in range(len(x)):
    rad += (E[i] - A - B*x[i])**2

uncertaintyE = np.sqrt(1/(len(x)-2) * rad)

radA,sumXsq = 0,0
for i in range(len(x)):
    radA +=  (x[i]**2)
    sumXsq += (x[i]**2)

delta = len(x) * sumXsq - np.sum(x)**2
uncertaintyA = uncertaintyE * np.sqrt(radA/delta)
uncertaintyB = uncertaintyE * np.sqrt(len(x)/delta)

print("Uncertainty in A =",uncertaintyA)
print("Uncertainty in B =",uncertaintyB)
```

```
Uncertainty in A = 8.635145306757664
Uncertainty in B = 0.01298389073188239
```

The best value for A is given by -44.34 ± 8.64 and the best value for B is given by 1.71 ± 0.01. A has units of energy, [keV] and B has units of energy per channel [keV/ch] - assumine x has units of "channel"?

Part B)

Compare your values to those obtained from a linear fitting routine in Python (Thats here!)

In [6]:
```
def LinearFit (x,A,B):
    """ Produce Linear Fit Hypothesis Function """
    return A + B*x

optVals,optCov = opt.curve_fit(LinearFit,xdata=x,ydata=E,p0=[-44,2])

# Print Results
print("Best value for A =",optVals[0])
print("Best value for B =",optVals[1])
```

```
Best value for A = -44.337083000673495
Best value for B = 1.7066164010557967
```

Using the "scipy.optimizer.curve_fit()" function, with a linear fit hypothesis space, we arrive at a similar conclusion for the best possible values of A and B, being -44.34 and 1.71 respectively.

Part C)

We observe an "unknown" photopeak at $x_\mu = 688.0 \pm 1.2$. Use the results fro, part A to determine it's Energy, E_μ and error.

In [7]:
```
# We use the Linear Fit Hypothesis
x_mu = 688.0
dx_mu = 1.2
E_mu = LinearFit(x_mu,A,B)

# Compute Uncertainty
errorVector = np.sqrt((uncertaintyB/B)**2 + (dx_mu/x_mu)**2)
uncertaintyE_mu = uncertaintyA + B * x_mu * errorVector

# Print results
print("Prediction for E =",E_mu,"[KeV]")
print("Prediction error for E =",uncertaintyE_mu,"[KeV]")
```

```
Prediction for E = 1129.8150016468576 [KeV]
Prediction error for E = 17.799808962780517 [KeV]
```

Part D)

The Error in Part (c) is unreasonably large. We can reduce it by rewriting the function use for calibration. We instead choose $E = A' + B'(x - x_0)$ to fit with data. We use $x_0 = 0, 100, 200, \ldots, 1100$. What do we observe? Which value of x_0 results in the most precise calibration? Use this to recalculate E_μ.

In [8]:
```
def ModifiedLinearFit (x,x0,A,B):
    """ Produce Modofied Linear Fit Hypothesis Function """
    return A + B*(x - x0)
```

Question 2

Show that errors should be added in quadrature by adding two Gaussians and examining the resulting distribution: add a Gaussian centered at x with wdith σ_x to a Gaussian centered at y with width σ_y. Show resulting distribution has width of $\sqrt{\sigma_x^2 + \sigma_y^2}$.

The general form of a normalized Gaussian function over the domain r is given by:

$$N(\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma_r^2}} e^{-\frac{(r-\mu)^2}{2\sigma^2}}$$

Thus for the above condition over the domain r, we have the sum of the two distributions: $N(x, \sigma_x) + N(y, \sigma_y)$ is given by:

$\frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{(r-x)^2}{2\sigma_x^2}} + \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(r-y)^2}{2\sigma_y^2}}$ The uncertainty in each is given by σ_x or σ_y

Question 3

Let $f(a, b)$ be a product of the parameters to some power, $f(a, b, c) = a^m b^n c^p$.

Part A)

Show that the relatice uncertainity of f is given by: $\left(\frac{\Delta f}{f}\right)^2 = \left(m\frac{\Delta a}{a}\right)^2 + \left(n\frac{\Delta b}{b}\right)^2 + \left(p\frac{\Delta c}{c}\right)^2$

From uncertainties in multiplication, we know the following relationship holds:

$$\frac{\Delta f}{f} = \sqrt{\frac{\Delta a}{a} + \frac{\Delta b}{b} + \frac{\Delta c}{c}}$$

as also shown in the question above. From exponentiation, such as $f(a) = a^m$, we also know:

$$\frac{\Delta f}{f} = |m|\frac{\Delta x}{x}$$

Thus combing the two uncertainties resulting from the multiplication of exponentiation results in:

$$\left(\frac{\Delta f}{f}\right) = \sqrt{\left(m\frac{\Delta a}{a}\right)^2 + \left(n\frac{\Delta b}{b}\right)^2 + \left(p\frac{\Delta c}{c}\right)^2}$$

or

$$\left(\frac{\Delta f}{f}\right)^2 = \left(m\frac{\Delta a}{a}\right)^2 + \left(n\frac{\Delta b}{b}\right)^2 + \left(p\frac{\Delta c}{c}\right)^2$$

Part B)

Demonstate that this relation enable s quick estimate if $m = n = p$ or at least approximately equal, and one parameter has a much larger relative error than the others.

If m = n = p, we can simplfy the equation,

$$f(a, b, c) = (abc)^m$$

And we can simply the error:

$$\left(\frac{\Delta f}{f}\right)^2 = m^2\left[\left(\frac{\Delta a}{a}\right)^2 + \left(\frac{\Delta b}{b}\right)^2 + \left(\frac{\Delta c}{c}\right)^2\right]$$

Thus if one paramater has a larger error than the others, it dominates all of the other components. For example if $\Delta a >> \Delta b$ and $\Delta a >> \Delta c$. We can then approcimate the error in f as:

$$\left(\frac{\Delta f}{f}\right)^2 = m^2\left(\frac{\Delta a}{a}\right)^2$$

```
In [ ]:
```