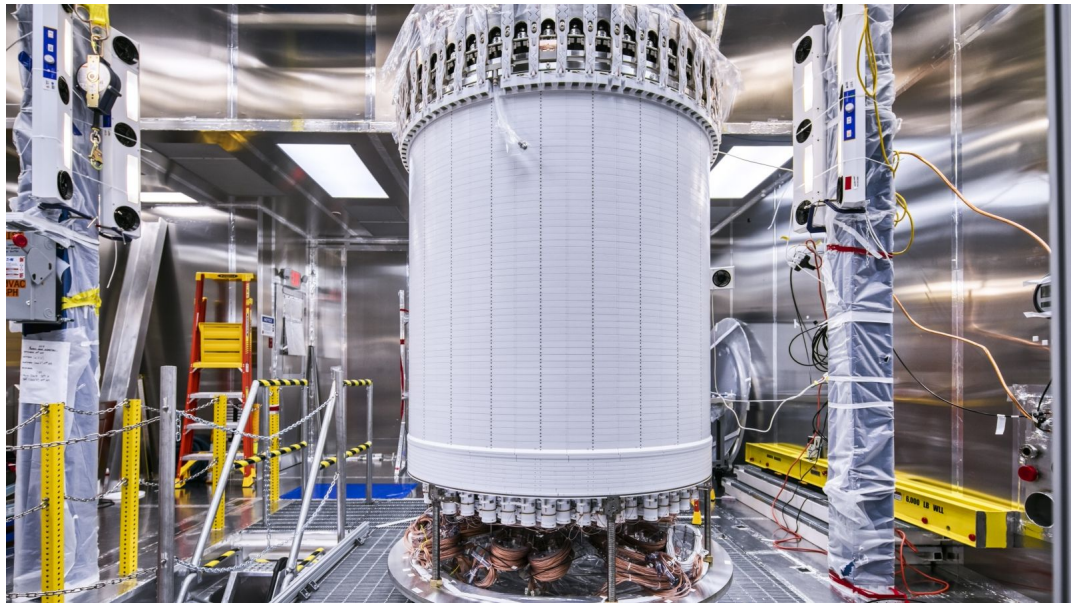
RADON EMANATION ANALYSIS

Seth Bendigo

South Dakota School of Mines and Technology

MOTIVATION - LZ

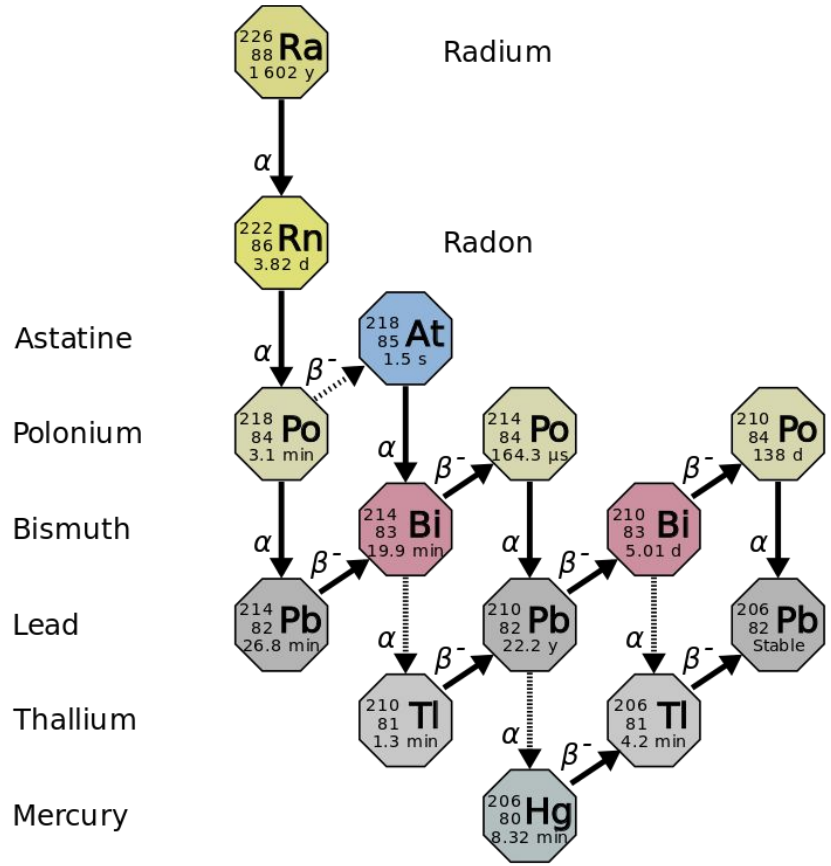
- LZ will look for dark matter events occurring with its detector.
- For a detector such as this, it is important to find the backgrounds and reduce them where necessary.
- Being a noble gas, radon likes to diffuse into the environment.
  - Such as the liquid xenon itself.
- As a result, radon granddaughter Pb-214 is a dominant background.
- It is important to fully understand this background and try to reduce it wherever possible.



LZ Time Projection Chamber - Photo by Matthew Kapust Sanford Underground Research Facility.
https://www.sanfordlab.org/article/lz-time-projection-chamber-assembly-completed

RADON GRANDDAUGHTER PB-214

- Beta decay of Lead-214 is the main issue.
  - Also known as the “naked” beta.
- 9.2% of the time this decay will occur without a gamma ray.
- This can lead to it being discerned as a dark-matter event.
- Which is why we need to measure and reduce radon background activity where possible.
- We can measure radon emanation of objects using the radon emanation system at SDSM&T.



RADON EMANATION SYSTEM

1. Place sample to be measured in a radon emanation chamber.

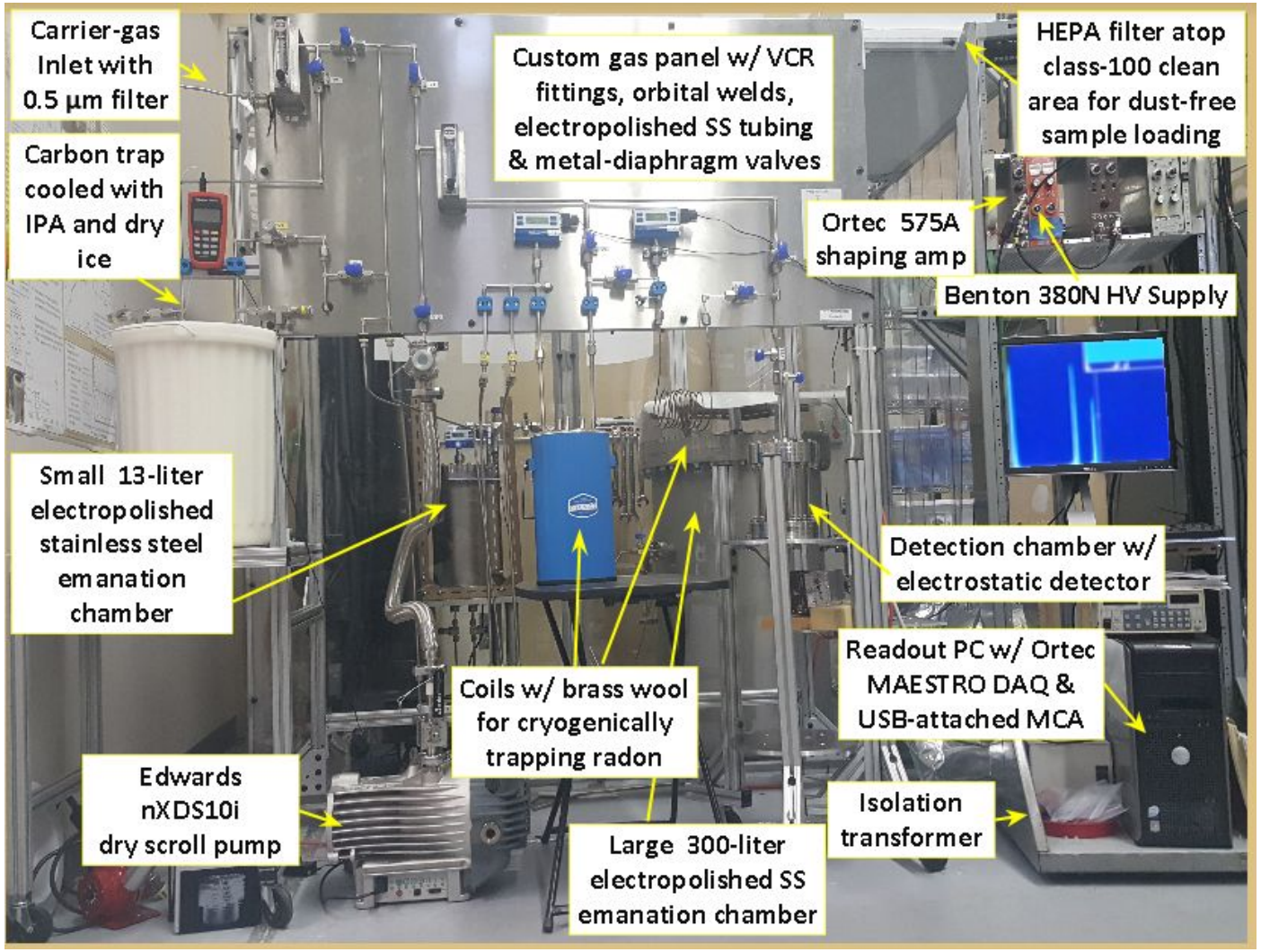
2. Rinse chamber with low-Rn N2 to remove residual air from when the source was placed inside the chamber.

3. Allow sample to emanate radon for ~1 week.

4. Let low-Rn carrier gas into the emanation chamber through cooled carbon trap.

5. Open valves and pump on carrier gas and radon to condense the radon on the first of two brass wool traps, cooled using LN2.

6. Warm first brass-wool trap and repeat step 5 using a second (smaller) trap.



Located in the Dakota Building at SDSM&T

Instructions and picture from a 2017 poster by R. Leonard

BACK TO THE DECAY CHAIN

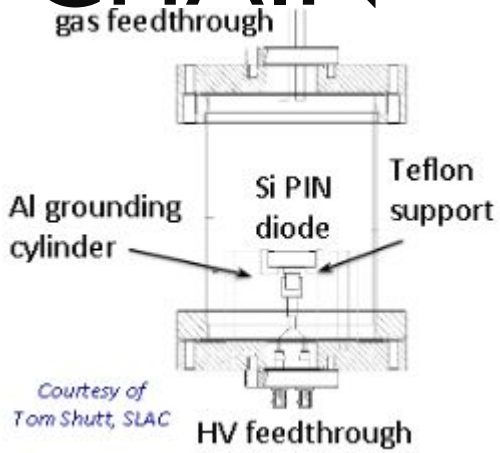
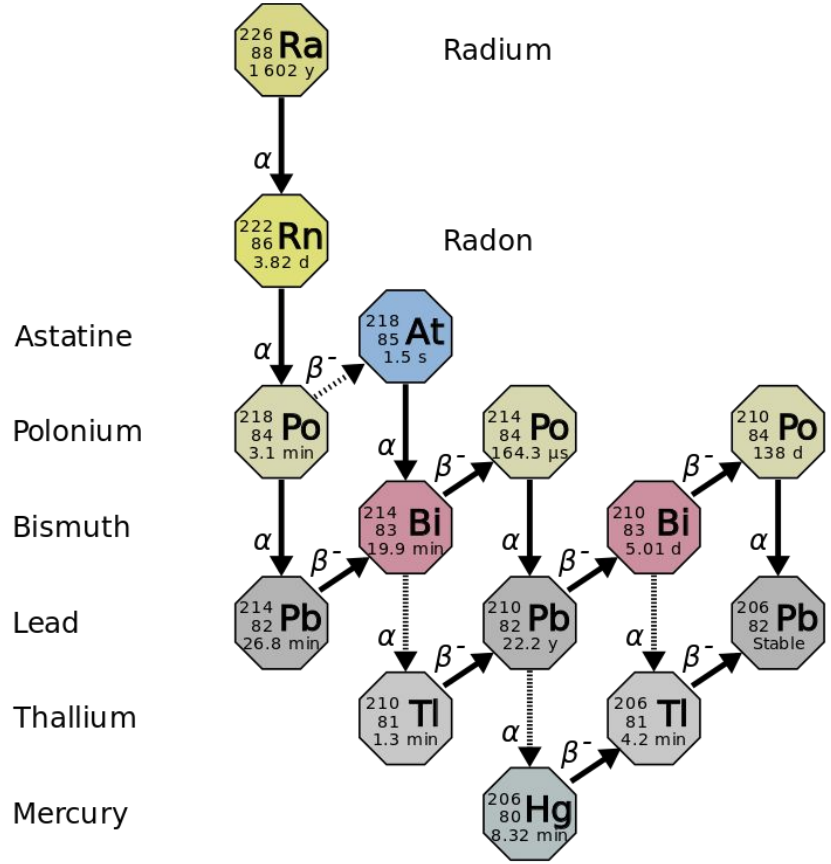


Diagram of the electrostatic detector which uses a silicon pin diode to detect alpha particles of varying energy channels.

- The electrostatic detector, relative to the walls, is kept at -2000 V.
- Po-218 is positively charged 88% of the time, which draws it to the detector.
- Collection efficiency is about 45%.
- Alpha decays of the subsequent Po-218, Po-214, and Po-210 are all detected and then used in the data analysis.
- Po-210 events are most often used as a calibration signal.
- Po-218 and Po-214 are then used to determine the emanation rate.



By User:Tosaka - File:Decay chain(4n+2, Uranium series).PNG, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=33293646

RADON EMANATION ANALYSIS

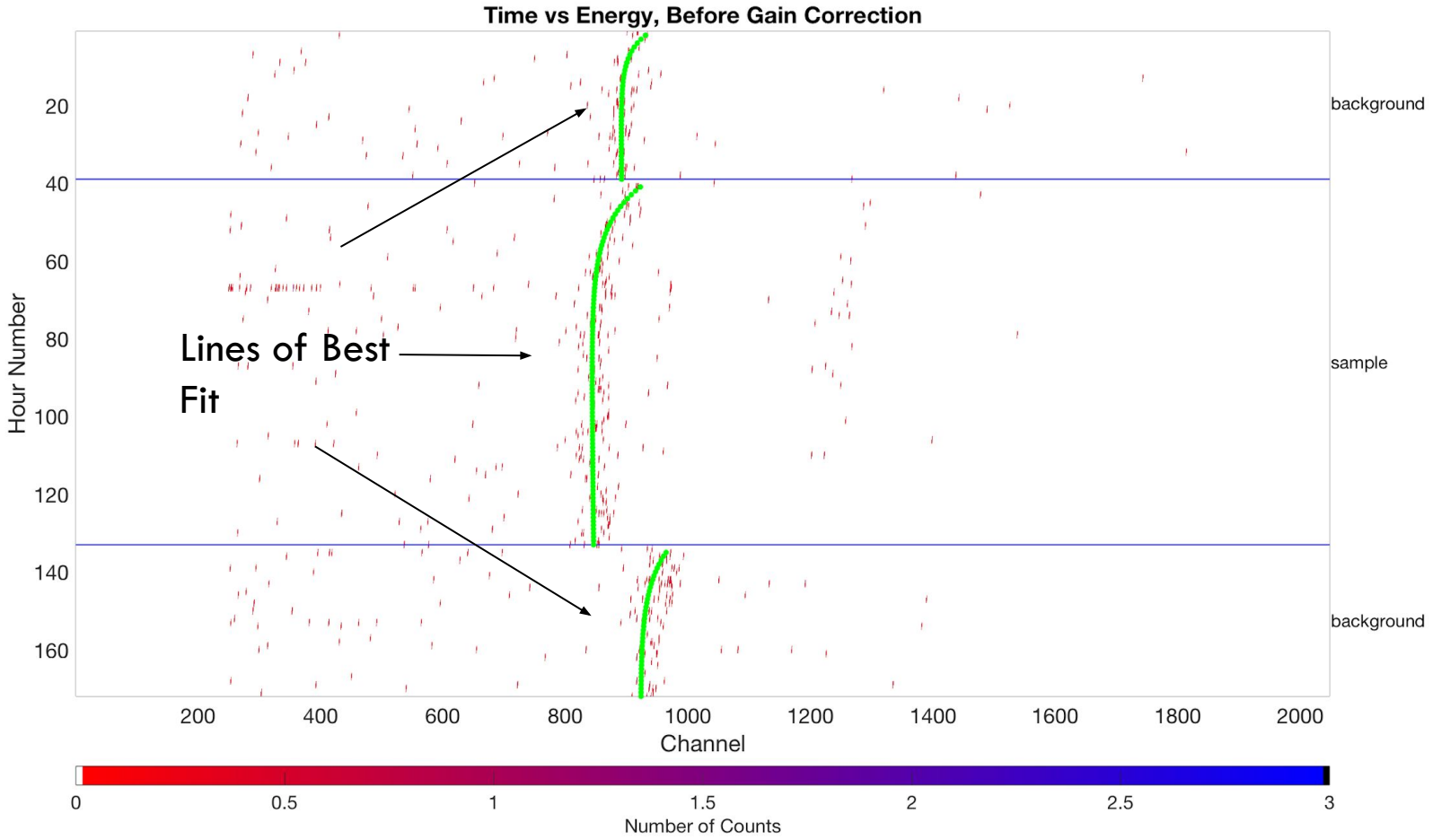
Currently, the code for the emanation analysis is written in MATLAB. However, we are in the process of converting the code into Python.

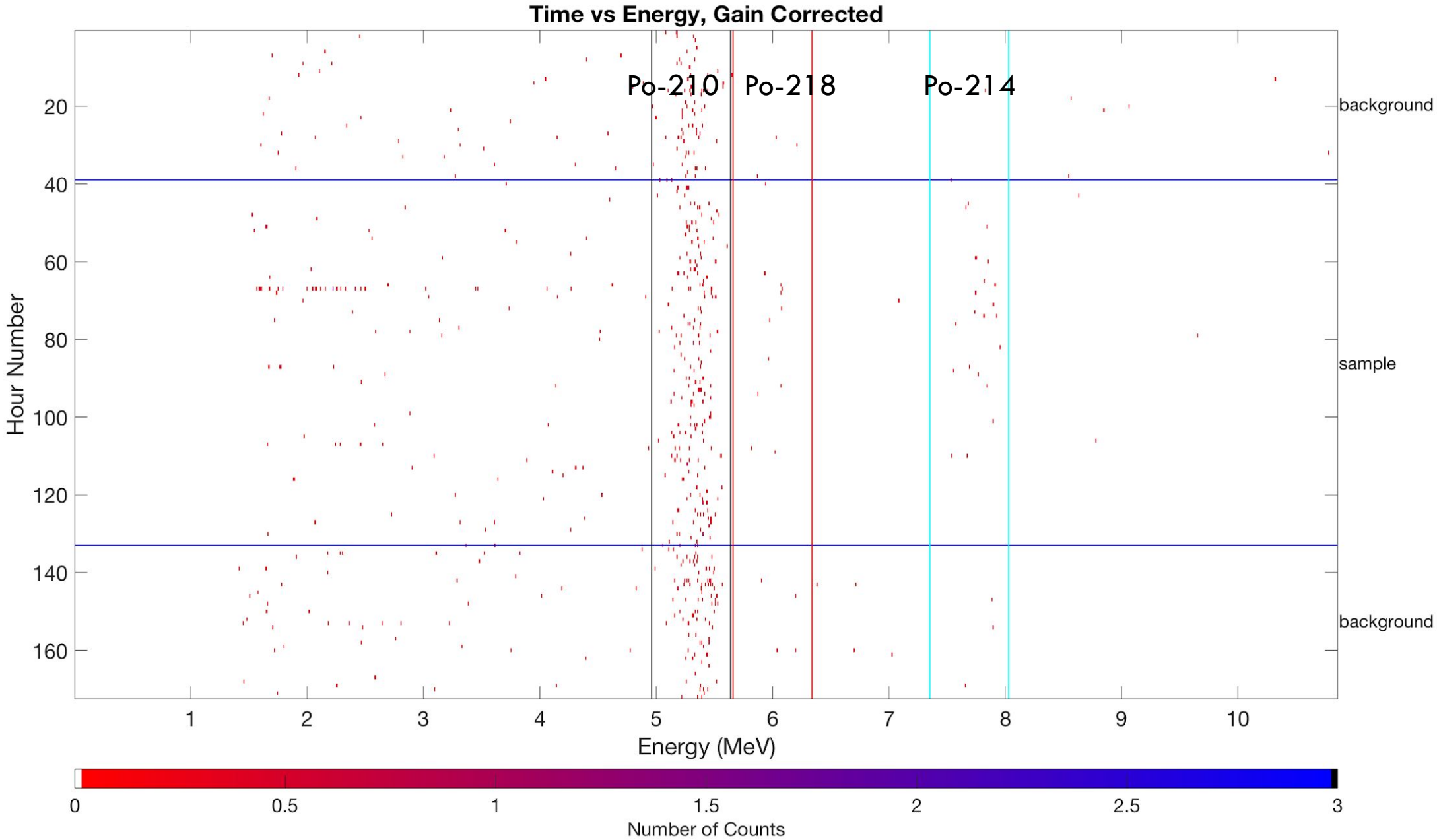
- Older codebase
- Unused code
- Need a license
- A more difficult to read language

- No license needed
- Built from the ground up to be more modular
- Easier to add new features
- Easy to read language

GAIN CORRECTI

- Our detector has a known issue where the detection is not consistent.
- However, we know Po-210 events will occur at a known energy of about 5.4 MeV which allows us to correct for this error.
- The subsequent slides show graphs of two runs with data before and after gain correction.




Time vs Energy, Gain Corrected
Po-210
Po-218
Po-214
background
sample
background
Hour Number
20
40
60
80
100
120
140
160
1
2
3
4
5
6
7
8
9
10
Energy (MeV)
0
0.5
1
1.5
2
2.5
3
Number of Counts

PEAK FINDING

Instead of using a function defined by a best fit line to determine the peaks, we can have the program find the peaks for us. Example:

Number of Counts: 11141

Channel: 12345

Sum → 1+1+1+4+1 = 8

Weighted Sum → (1)1+(2)1+(3)1+(4)4+(5)1 = 27

$$\frac{\text{Weighted Sum}}{\text{Sum}} = \frac{27}{8} = 3.375 \xrightarrow{\text{Round}} 3$$

This tells that the algorithm found the peak to be at channel 3.

```
def peakFinder(countMatrix,lowerHourLimit,upperHourLimit,peakPreviousHour):

    listOfPeaks = list()
    listOfCounts = [peakPreviousHour]
    sumOfCountsMultipliedByChannel = 0
    delta = 50 #default channel range to search for peak

    for i in range(lowerHourLimit+1,upperHourLimit):
        channelRange = [peakPreviousHour - delta, peakPreviousHour + delta]
        if channelRange[0] < 0 or channelRange[1] < 0:
            channelRange = [0, 2 * delta]
        if channelRange[1] > len(countMatrix[i]):
            channelRange = [len(countMatrix[i]) - 2 * delta, len(countMatrix[i])]

        listOfCounts = countMatrix[i:i+1,channelRange[0]:channelRange[1]]

        sumOfCountsMultipliedByChannel = 0
        for j in range(channelRange[0],channelRange[1]):
            sumOfCountsMultipliedByChannel = sumOfCountsMultipliedByChannel + j * listOfCounts[0][j - channelRange[0]]

        if np.sum(listOfCounts) != 0:
            peakPreviousHour = int(sumOfCountsMultipliedByChannel / np.sum(listOfCounts))

        listOfPeaks.append(peakPreviousHour)

    return listOfPeaks
```

Define a Range Channels to Search.

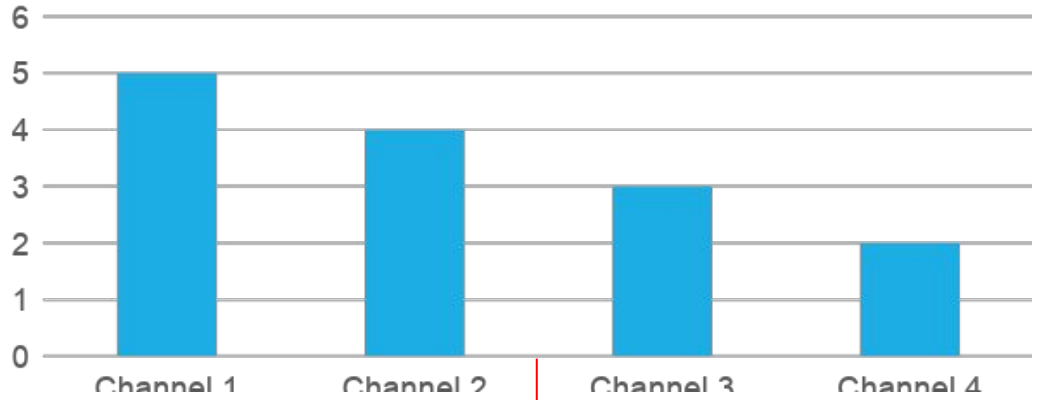
Make a List of Counts Within Range.

Take the Weighted Sum.

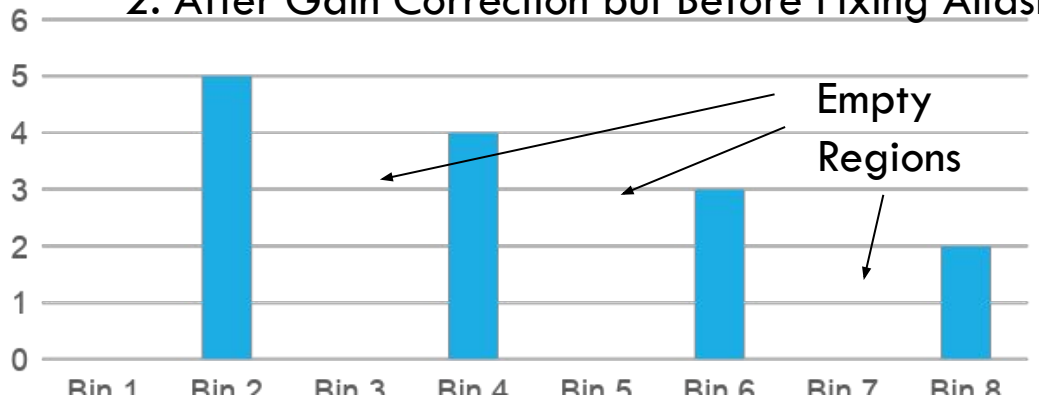
Take the Sum and then Divide.

GAIN CORRECTING AND FIXING ALIASING

1. Before Gain Correction and Fixing Aliasing



2. After Gain Correction but Before Fixing Aliasing

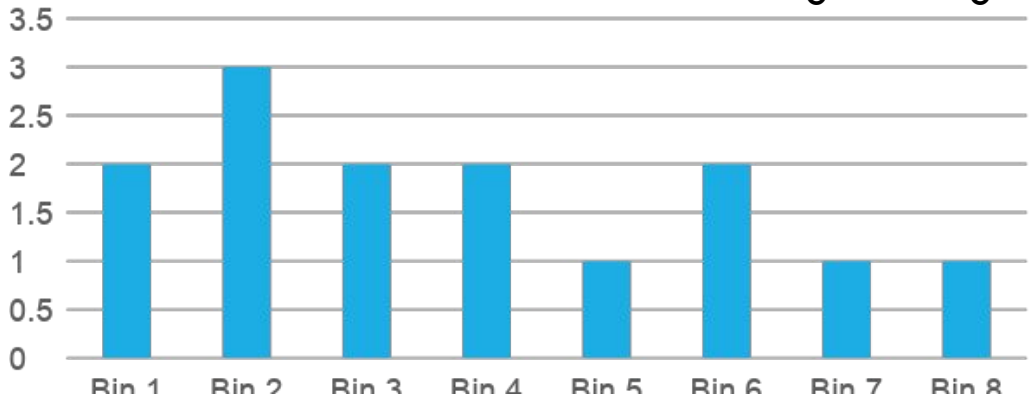


When correcting the gain, we are changing the values of the bins from channels, to energy.

Because of this, we move the data to a matrix that has a larger number of bins.

This leads to "aliasing" or empty regions between data points. We fix the aliasing by redistributing the data.
A rudimentary example of this is shown here.

3. After Gain Correction and Fixing Aliasing



```
def fixAliasing(oldRow, newRow, gainFactor):

    oldRowLen = len(oldRow)
    newRowLen = len(newRow)

    for i in range(oldRowLen):

        lowerLimit = 1 + floor((i - 1) * gainFactor)
        upperLimit = 1 + floor(i * gainFactor)

        for j in range(lowerLimit,upperLimit):
            overlap = min(j,upperLimit-1) - max(j - 1, lowerLimit-1))
            newRow[j] = oldRow[i] * overlap * floor(1 / gainFactor) + newRow[j]

    return newRow
```

Defining Region as a Function of the Gain Factor

Defining Overlap, and Then Moving Data to the Correct Bin.