6-2022

Development of Quantitative Methods to Study PFAS Using Proton Induced Gamma-Ray Emission

Colin Langton
Union College - Schenectady, NY

Development of Quantitative Methods to study PFAS Using Proton Induced Gamma-Ray Emission

By

Colin Langton

* * * * * * * * *

Submitted in partial fulfillment

of the requirements for

Honors in the *Department of Physics and Astronomy*

UNION COLLEGE

June, 2022

Abstract

LANGTON, COLIN Development of Quantitative Methods to study PFAS Using Proton Induced Gamma-Ray Emission. Department of Physics and Astronomy, June 2022

ADVISOR: Michael Vineyard

Per- and polyfluoroalkyl substances (PFAS) are man-made chemicals that have become a major environmental concern. They can be found in a broad range of everyday products and pose a significant risk to the public due to their adverse health effects. They are persistent, bioaccumulate and do not break down in the environment. This project specifically aims to determine the concentration of Fluorine, a key identifier of PFAS, in environmental samples. To do this, we employ proton induced gamma-ray emission (PIGE) to screen for Fluorine within our samples. PIGE is performed at the Union College Ion Beam Analysis Laboratory using a 1.1-MV tandem Pelletron accelerator. Samples are bombarded in an ex-vacuo setup with an incident energy of 1.8 MeV and emitted gamma-rays are detected with a high-purity Ge detector. This research defines two quantitative methods for PIGE analysis. The first uses standards of known concentrations to compare to collected samples. The second uses a developed python application to directly compute concentration values based on nuclear theory. We present preliminary results on the accuracy of both methods.

Contents

List of Figures

Chapter 1

Introduction

Per- and polyfluoroalkyl substances (PFAS) are man-made chemicals that have become a major environmental concern [1]. They can be found in a broad range of products including food packaging, stain- and water-repellent fabrics, nonstick products, makeup, fire-fighting foams, and electronics. PFAS pose a significant risk to the public due to their adverse health effects [2, 3]. While the extent of these effects is still being researched, studies have revealed connections between high exposure and diseases such as high cholesterol, thyroid disease, pregnancy-induced hypertension, ulcerative colitis, and kidney and testicular cancer [4]. In addition to this, they are persistent, bioaccumulate, and do not break down in the environment.

As a result of these properties of persistence and bioaccumulation, it has been shown that PFAS accumulate in the bloodstream of most United States citizens in detectable concentrations. The presence of PFAS in human blood has been actively monitored since 1999 by the National Health and Nutrition and Examination Survey (NHANES) conducted by the Centers for Disease Control and Prevention [5]. Fortunately, since the survey's initial report, there has been a dramatic decrease in the level of PFAS in human blood due to policy change and an active effort to replace the use of PFAS within consumer products. As recently as 2014, NHANES has shown that the average blood PFAS level for the four most common groups of PFAS have each dropped below 10 micrograms per liter [5]. These results are shown in Figure 1.1. PFOS (green) and PFOA (blue) are the two most widely

used PFAS compounds and have seen the most progress in being phased out. However, it will be crucial to phase out all forms of PFAS to completely protect human health.

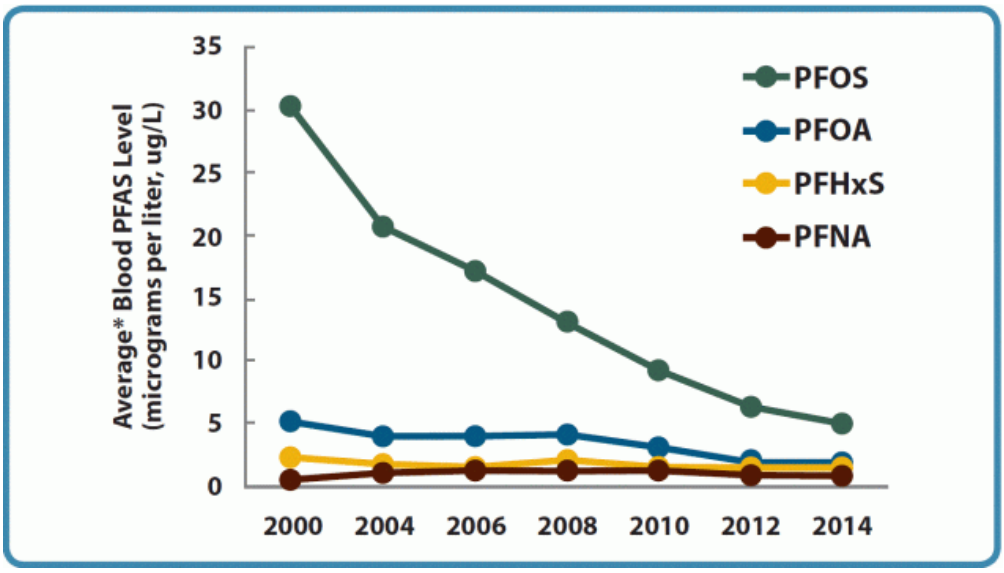


Figure 1.1: Average PFAS Blood Level 2000-2014 [5]

Even with this drop, the EPA continues to work to reduce the concentration of PFAS within potential sources of human exposure, most notably drinking water supplies. The agency has set the limit of PFAS within drinking water supplies to be 70 parts per trillion [3]. However, the Environmental Working Group (EWG) says that to protect human health, this limit should be 1 part per trillion [6]. A study conducted by the group in 2019 reveals that for drinking water samples taken from 44 different locations across 34 states, 41 exceed the EWG advisory and 2 exceed the advisory set by the EPA [6].

To further reduce PFAS exposure to the general public, it is important to understand the prevalence and concentration of PFAS within a variety of samples and how it may spread throughout the environment and into areas of human exposure. Understanding where and how PFAS spreads will allow for further policy to reduce human exposure.

Traditionally, methods of determining the concentration and prevalence of PFAS have been limited to chemical approaches using liquid chromatography/tandem mass spectroscopy

which aims to detect specific compounds of the PFAS family [9]. Currently, the EPA has defined 7 of these methods to screen for specific compounds in a wide range of sources including potable water, non-potable water, and source air emissions. These methods are highly sensitive to specific PFAS compounds, but they can be time-consuming and are destructive to the sample of interest. Currently, the EPA is in the process of creating new methods to expand the variety of samples that can be screened for high levels of PFAS. One area of interest is to be able to quantify large groups of PFAS efficiently and accurately within environmental samples. The EPA has designated this kind of method as one of great interest and is currently working to develop the procedure to quickly determine the presence or absence of PFAS.

One proposed method for quantifying a large group of PFAS quickly and accurately is through the use of a tandem particle accelerator. This method uses low-energy nuclear reactions to identify the presence and concentration of Fluorine within a sample which is a key identifier for all PFAS. Although Fluorine is naturally abundant in soil samples, natural levels range between 150 and 400 ppm [7]. We expect PFAS contamination to be on the order of 10000 ppm or more. However, it is important to note that Fluorine contamination can also be caused by phosphoric fertilizers [8]. While this must be taken into account in studying certain areas such as farmland, a significant amount of fluorine otherwise can be attributed to PFAS. This can be confirmed by performing chemical analysis on a single sample to determine if PFAS compounds are present. While this research requires highly specific lab equipment, samples can be screened in less than 10 minutes without destroying the sample. This process has been shown to be especially effective at the University of Notre Dame under Dr. Graham Peaslee [10].

At the Union College Ion Beam Analysis Laboratory (UCIBAL) we have shown the method of Proton Induced Gamma-Ray Emission (PIGE) to be highly effective at qualitatively detecting the presence of PFAS within a variety of samples including soil, makeup,

powders, and small everyday items (e.g. dental floss). It is then our goal at the UCIBAL to quantitatively determine the concentration of fluorine within PFAS affected samples to determine the concentration of PFAS. Quantitative concentration methods have been presented by the International Atomic Energy Agency utilizing the method of PIGE most notably through using a group of standards with known amounts of fluorine [11]. In addition to this, there has been work done by Manteigas et al at the University of Lisbon to use computational methods and known nuclear cross-sections to simulate experimental data and extract concentration values [12]. At current, we have been unable to implement this computational approach effectively to produce results that match up with experimental data.

Our goal at the UCIBAL is to implement both a standards-based method and a computational approach to deduce quantitative concentrations. The standards approach will aim to compare unknown samples to a group of created samples with known concentrations. The calculational approach will aim to directly calculate concentrations using equations derived from theoretical nuclear interactions. Through these methods, we aim to understand how PFAS spreads through the environment as well as key sources of human exposure. One area of great interest is the impact when fire fighting foam is released into the environment in significant quantities. It is known that fire fighting foam contains a high concentration of PFAS and over time has been actively released into the environment for various reasons including large-scale accidents. To understand the greater effects of these types of incidents, it will be essential to have accurate quantification methods. The goal of this thesis is to establish the quantification procedures and computational method necessary to carry out the steps towards understanding the large-scale effect of PFAS within the environment.

Chapter 2

Methods

2.1 Proton Induced Gamma-Ray Emission

In our research, we utilize Proton Induced Gamma-Ray Emission (PIGE) to screen for light elements such as fluorine. PIGE works by bombarding a sample with a beam of protons in the energy range of a few MeV. These protons are generally produced in a low-energy tandem accelerator [11]. As the protons penetrate a sample, there are instances where the protons inelastically collide with a nucleus leaving it in an excited state. When the nucleus de-excites it emits a gamma-ray which can be detected. This process is shown in Figure 2.1. The energy of the gamma-ray identifies the element and the intensity of gamma-rays at a particular energy can be used to determine the concentration of the element. Specifically for our research, we will be looking at the characteristic fluorine peaks which have energies of 110 and 197 keV.

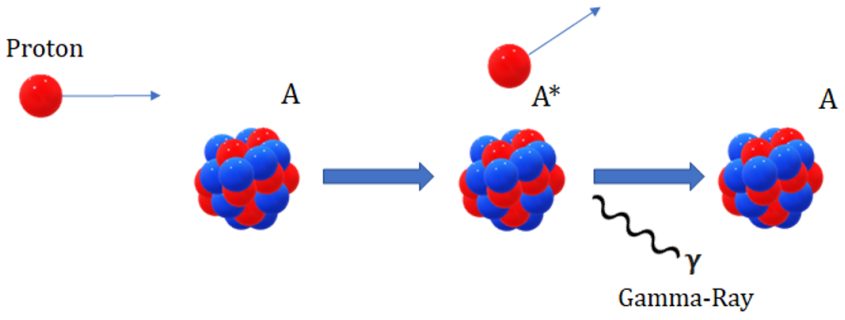


Figure 2.1: Illustration of PIGE Reaction.

2.2 Experimental Methods

At the Union College Ion-Beam Analysis Laboratory (UCIBAL), we use a 1.1-MV tandem Pelletron accelerator shown in Figure 2.2. In our lab, we accelerate protons up to energies of 2.2 MeV in vacuum. However, in order to use our Canberra Ge detector to measure gamma-rays, it is necessary to bring the proton beam out into the air before colliding with the target. To conduct these experiments, we have constructed an external beam facility to hold our targets fixed in each trial. Samples are placed 2 centimeters away from the beam pipe at a 45-degree angle relative to the proton beam. Then our Canberra Ge detector is placed 6.5 centimeters directly beneath the target. In order to reduce noise due to partial gamma-ray detection and background radiation, we have placed a circular lead collimator on the top of our detector to block unwanted photons. This entire external beam facility setup is shown in Figure 2.3. In this external beam setup, we calculated the beam loses 0.4 MeV of energy due to interaction with the 7.5 μm Kapton vacuum window in our beam pipe and the 2 centimeters of air. This means that the average beam energy on target is 1.8 MeV with incident currents of up to 10 nA.

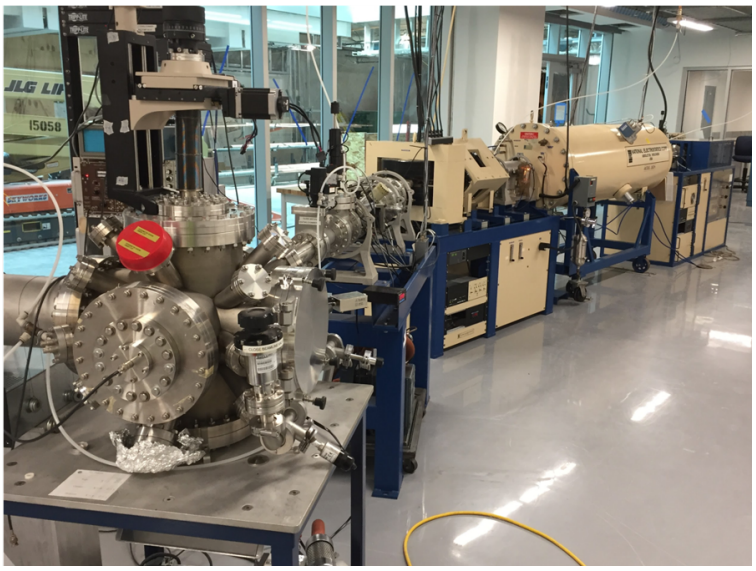


Figure 2.2: The 1.1 MV Tandem Pelletron Accelerator at the UCIBAL.

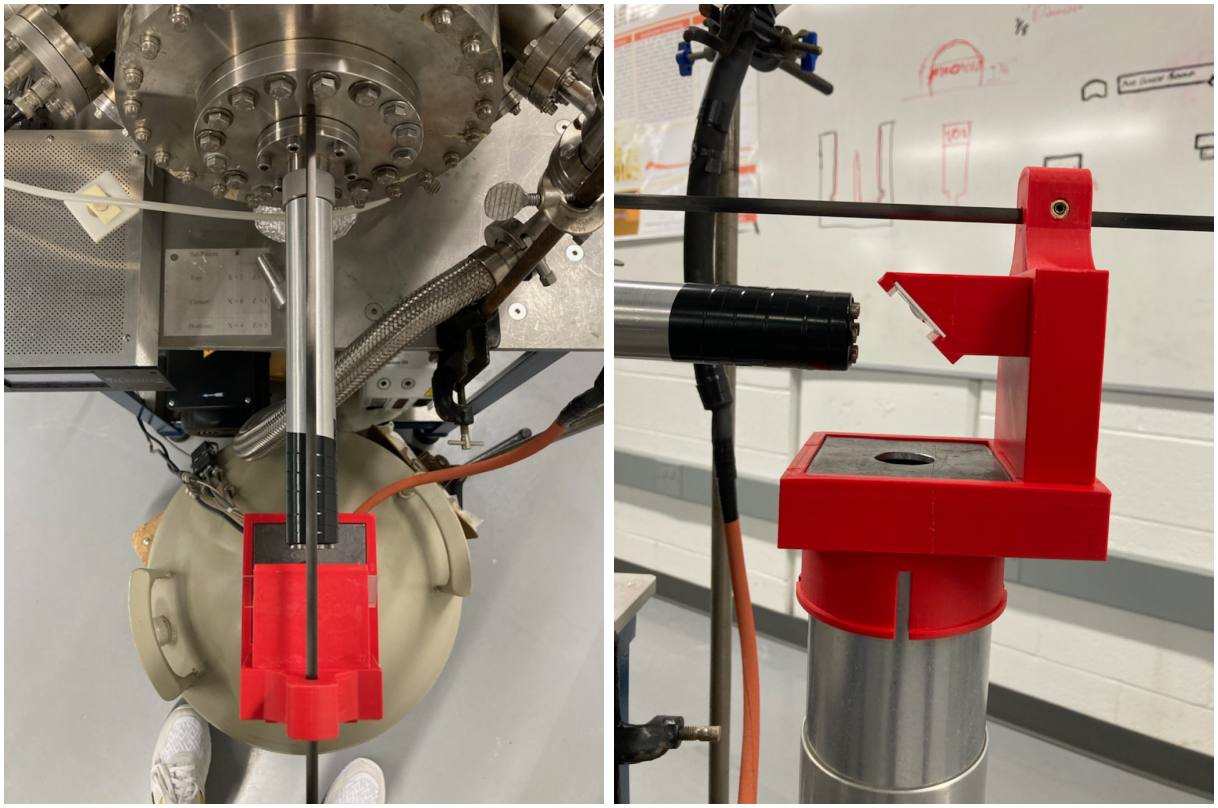


Figure 2.3: Ex-vacuo PIGE setup at the UCIBAL. Facility is mounted on high purity Germanium detector 90 degrees relative to proton beam. Pictured is the external beam from the 1.1 MV tandem accelerator lining up with the target holder.

2.3 Sample Creation

In order to achieve accurate and consistent results, we require that our targets are solid and thick. This reference to being solid simply refers to whether the target is able to be placed in our external beam facility. On the other hand, this requirement of being thick has to do with our ability to calculate concentration values. To simplify any future calculations and to reduce extraneous interactions, it is necessary that our beam is absorbed by the sample and does not penetrate and hit the back of our sample holder. Using SRIM, we are able to calculate the required thickness for any target sample to ensure we are not corrupting our data [13]. Unfortunately, many of our samples of interest do not conform to these two requirements. Powdery substances including soil samples require additional preparation in

order to meet the standards of our external beam facility. For these samples, we create pellets of diameter 1cm with the required thickness to absorb our beam. To do so, we combine 0.5 grams of our sample of interest with a binding agent of cellulose or polyvinyl alcohol. With the use of cellulose, it is necessary to mix until the two powders are completely homogeneous. The sample and binding agent are then poured into our pellet-making chamber. A metal cylinder is placed on top of the sample and then the entire chamber is placed in a lever-driven hydraulic press where pressure is applied to the metal cylinder. Once this specific pressure is reached, we let it sit under constant pressure for 1 minute before releasing the press. The sample is then ready for use within our external beam facility.

2.4 Data collection

As mentioned in our experimental methods section, data is collected using a Canberra High Purity Ge detector. The resulting gamma-rays are processed and counted using Ortec's MAESTRO software which sorts gamma-ray energies into channels [14]. Samples are run on for 5 minutes each in batches of 3 pellets. After collecting data for 3 pellets, a Faraday cup is placed on the end of the beam pipe to measure charge over the same period of 5 minutes. This charge value is then saved along with the raw data files of the three targets for analysis. This charge integration is required for future calculations and allows us to normalize yield values. Unfortunately, as a result of inconsistencies in beam current, it is required that this charge integration is conducted multiple times throughout data collection and may differ by a few nA's at any time. The uncertainty in charge integration is around 10%. While this process is standard for most of the work done at the UCIBAL with thick targets, it is easy to modify for a variety of targets and experimental controls. Time can be shortened or lengthened as long as the corresponding charge integration time is equal in length.

Chapter 3

Data Analysis

3.1 Data Processing

In order to determine concentrations from experimental data, raw data files must be processed and calibrated and yield counts must be extracted. The data received from the detector and processed by MAESTRO records counts in arbitrary channel numbers which are not calibrated to the necessary energy values. To calibrate the channel numbers to units of keV, we use the characteristic 110 keV and 197 keV Fluorine peaks as well as the background electron-positron annihilation peak at 511 keV. In each of our standards, we know there exist significant peaks at 110 and 197 keV. Furthermore, in our lab setup, we always see a peak at 511 keV corresponding to electron-positron annihilation due to background beta$^+$ decay. We expect the spacing between energy values to be linear, so a linear fit can be deduced using the center of the 110, 197, and 511 keV peaks and their associated channel number. This fit can then be applied to every data file collected in the same collection run.

It is also important to normalize our data to the charge values associated with each raw data file. While our beam current is generally consistent, this step is necessary in order to be able to compare data collected over a wide variety of collection dates. This is because gamma-ray yield is proportional to charge and while our beam is generally pretty consistent, small changes over different runs can have a drastic effect on our results.

Raw data files are converted into text files using a python code. The code removes the header and footer generated by MAESTRO and calibrates the channel values to energy values using the calibration equation included in the raw data file footer. The code also allows for the input of a background file to strip background counts from the experimental data. This background data can be obtained by running PIGE on just air or a sample with no fluorine. In this process, each yield count is divided by the charge value extracted from the file name including the background file. Finally, the background counts at each energy step are subtracted from the experimental accounts at the corresponding energy. The formatted data is finally saved as a text file with two columns: the first storing the energy values in keV and the second storing the yield counts per collected charge.

3.2 Extracting Yields

After the data is properly processed and normalized, yield counts can be extracted for specific energy peaks. While we are concerned with the counts at a precise energy, due to detector resolution, the counts may be spread across more than one energy channel. To account for this, we consider counts between the range of 108 keV and 112 keV. Since in this process we have already removed the background, almost all of the counts in this range should be a result of the 110 keV peak due to the nuclear interaction corresponding to Fluorine 19. The reason we choose the 110 keV peak of Fluorine 19 is that Fluorine 19 is the only naturally abundant source of Fluorine and the 110 keV peak is the most significant in terms of total counts. This yield is used in both methods to determine concentration.

Chapter 4

Quantitative Methods

4.1 Computational Method

The IAEA has outlined a calculational method utilizing the theoretical nuclear reaction excitation function for proton-induced gamma-ray emission [11]. To make the computation a little bit easier, we have chosen to utilize only thick targets in which all proton energy is absorbed within the sample. This function is given as:

$$Y(E) = \epsilon_{abs}(E_\gamma) \cdot \left(\frac{Q}{e}\right) \cdot f_m \cdot f_i \cdot N_{av} \cdot A^{-1} \cdot \int_0^{E_0} \sigma(E)/S_m(E)\, dE, \tag{4.1}$$

where $Y(E)$ is the yield at a specific ion beam energy, ϵ_{abs} is the absolute efficiency of the detector at the energy of our gamma ray, $\frac{Q}{e}$ is the total number of protons on target, f_m, f_i, and A are the mass fraction, isotopic abundance and atomic mass respectively, of the relevant element, N_{av} is a Avogadro's number, E_0 is the incident energy of the beam, $\sigma(E)$ is the nuclear cross section for the relevant element, and $S_m(E)$ is the stopping power of the material expressed in energy per areal units.

Our work aims to solve for the mass fraction f_m for fluorine at its characteristic energy of 110 keV. The yield value $Y(E)$ and collected charge Q are experimental values and are the main input into the equation. The detector efficiency and incident energy are characteristic of our experimental setup and remain constant for our purposes. Avogadro's number is

also a known quantity. Finally, isotopic abundance and atomic mass are well-known for any element of interest and can be simply inputted accordingly. This leaves us with finding values for nuclear cross-section and the stopping power of a material. Both values will be discussed further below.

4.1.1 SRIM Calculations

Stopping power is a parameter that takes into account how the proton beam interacts with the sample of interest. The complex nuclear and electromagnetic reactions that take place once the proton beam penetrates the target are too difficult to computationally simulate due to a large number of collisions and reactions. Thus, there must be a way to characterize the effects on a large scale. One of the key factors that go into this general calculation is that the beam is going to lose energy every time it interacts with something in the sample. This means that the gamma-ray reactions that we are concerned with don't all occur at our incident energy on target. The effect of stopping power is going to be heavily reliant on the makeup of the sample. A denser sample of heavy elements will cause a lot of reactions to occur very close to the surface while a less dense material of light elements will allow higher energy protons to reach further. Fortunately, this calculation can be simulated using the program SRIM as shown in Figure 4.1 [13]. SRIM allows for the input of the material of interest and particle type as well as incident energy. Outputted stopping powers in units of keV per area density can be used in equation 4.1 above. Since equation 4.1 requires the integral of this value up to the target incident energy, numerical integration methods must be implemented in the final method. Unfortunately, SRIM does not output the results in even increments, so it will be necessary to interpolate the results to match the energy steps of the nuclear cross-section data.

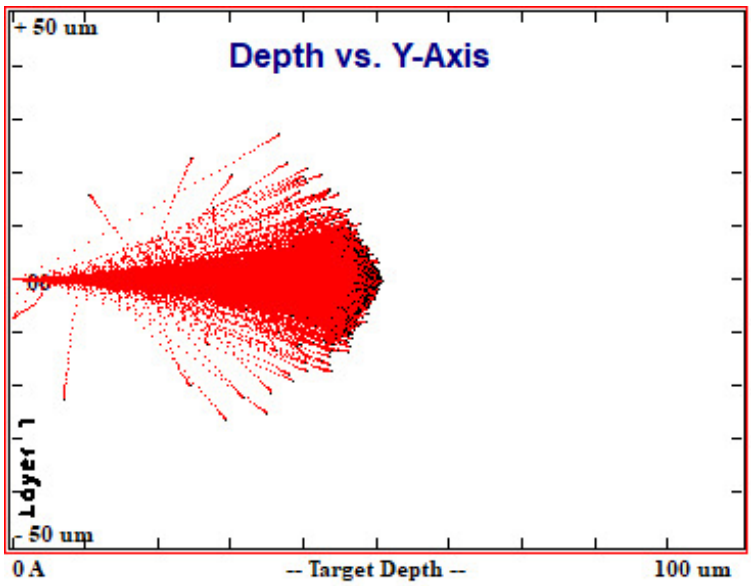


Figure 4.1: Simulation of protons with incident energy of 1.8 MeV in NaF and Cellulose Target

4.1.2 Nuclear Cross-sections

Nuclear cross-sections refer to the probability of a specific nuclear reaction occurring. This reaction is unique both to an element and the specific beam energy. Specifically, in our work, this refers to the probability of a proton exciting a Fluorine 19 nucleus and causing a 110 keV gamma-ray to be transmitted. Cross-section is measured in units of millibarns. This quantity has been the crucial piece of understanding PIGE reactions. Unfortunately, current results are mostly experimental which makes it hard to apply them to different experimental setups. While some programs try to calculate what the cross-section of a specific reaction should look like, we will attempt to implement experimental results from the Ion Beam Analysis Nuclear Data Library (IBANDL) [15]. The experimental cross-section as a function of beam energy for the 110 keV characteristic peak of Fluorine 19 is shown in Figure 4.2 [16]. The peaking structure of the cross-section is a result of transitions between quantum states and energy levels which further complicate any calculation. Fortunately, these results allow for the use of beam energies well above the UCIBAL's 1.8 MeV incident energy which will allow for use on a wide variety of low energy particle accelerators. Since equation 4.1

requires the integration of the nuclear cross-section up to the incident energy on target, a numerical integration method such as Simpson's Rule can be implemented to calculate the necessary value for a wide range of incident energies.

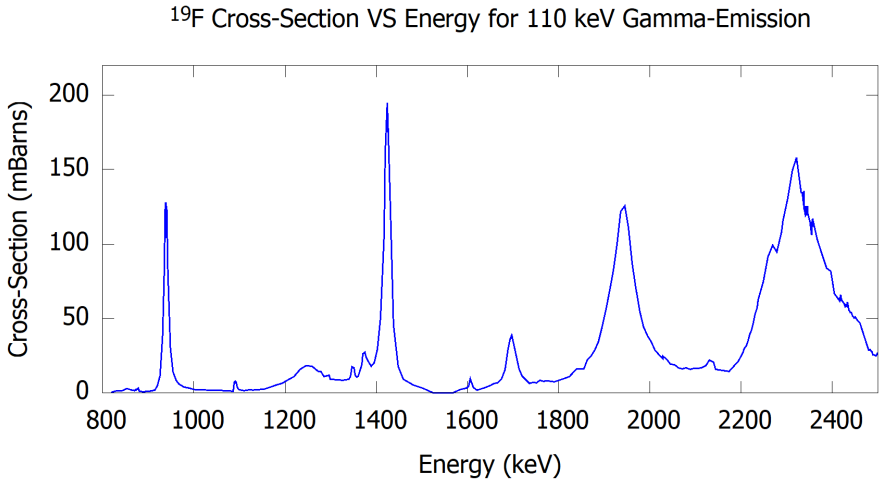


Figure 4.2: Experimental Cross-Section of Fluorine 19 at 110 keV as a function of beam energy

4.1.3 Detector Efficiency

Understanding the absolute efficiency of our high purity Ge detector is critical to calculating the mass fraction of our element of interest. This quantity accounts for the total number of gamma-rays emitted in the nuclear reaction between the proton beam and our sample that are counted by our detector. Since gamma-rays are equally likely to be emitted in all directions, our detector can't count all of them. Furthermore, there is an inherent error in the number of gamma-rays that do hit the detector that is accounted for. This latter value is the relative efficiency of our detector which is energy-dependent. The value is detector specific and varies based on the energy of the detected gamma-ray. To calculate the absolute efficiency in our experimental setup, there are two methods described below.

1. The first method is to calculate the absolute efficiency as a product of the relative efficiency multiplied by the solid angle of our experimental setup. The solid angle can

be calculated as the surface area of the detector divided by the distance from the target to the detector squared or equation 4.2.

$$\Omega = \frac{\pi r^2}{R^2} \tag{4.2}$$

where r is the radius of the detector and R is the distance from the detector to the target. The estimation for absolute efficiency is then calculated using equation 4.3.

$$\epsilon_{abs} = \epsilon_{rel} \cdot \Omega \tag{4.3}$$

2. The second method is to use an active source to determine the absolute efficiency. By placing a gamma-ray emitting source with a known activity rate in the place of a target we can see how many of the total gamma-rays emitted are counted by the detector. The absolute efficiency at the energy of the source is then computed using equation 4.4.

$$\epsilon_{abs} = \frac{\text{Yield of Detected Gamma-Rays}}{\text{Number of Gamma-Rays Emitted}} \tag{4.4}$$

4.1.4 Python Code

The goal of the quantitative procedure is to efficiently and quickly calculate concentrations of unknown samples. However, the necessary information and calculations require a significant amount of time for just one sample. To reduce run time, it will be beneficial to use a code to compute the necessary values to find the concentration in parts per million.

The created python code is able to compute all pieces of equation 4.1. The code takes inputs of experimental yield, charge collected, incident energy on target, sample stopping power, and element of interest. Yield, charge collected, and incident energy are simple input fields that are extracted from experimental data. Stopping power and the cross-section of the

element of interest are the complicated pieces that the code aims to simplify. Stopping power values are computed using the software SRIM [13]. This requires a precise understanding of the major elements in the sample. If the composition is unknown, Proton-Induced x-ray Emission (PIXE) can be performed to determine the heavy elements of the sample. Otherwise, samples can be combined with a known base to lessen the significance of the unknown materials on the stopping power values.

The computed SRIM values are saved and inputted directly into the python code for calculating concentrations. The calculation of concentration also requires values associated with an element at a specific reaction energy. This includes abundance, atomic weight, and most importantly nuclear cross-section. To manage these values, the code stores the information as singular element objects that include the necessary information. The objects are then pickled using the python pickle package and can be saved indefinitely. The specific information for Fluorine at 110 keV and 197 keV are already stored in the code and new elements can be added as long as the required information is known.

However, the most critical and hardest piece of the calculation is integrating the ratio of the nuclear cross-section to sample stopping power from 0 to the incident energy on target. Since both values of nuclear cross-section and stopping power values are sampled data, we require the use of a numerical integration method. The IAEA has defined a summation equation, shown in equation 4.5, to calculate this integral as long as the nuclear resonances are well-defined [11]. Using cross-section data from IBANDL, we can further fit the resonance by interpolation. This is shown in Figure 4.3 where an interpolation is created to fit the 110 keV nuclear cross-section associated with Fluorine-19 up to an incident energy of 1800 keV.

Furthermore, since stopping power is simulated using SRIM, we can also interpolate the calculated values as well in order to match the numerical energy values with the experimental

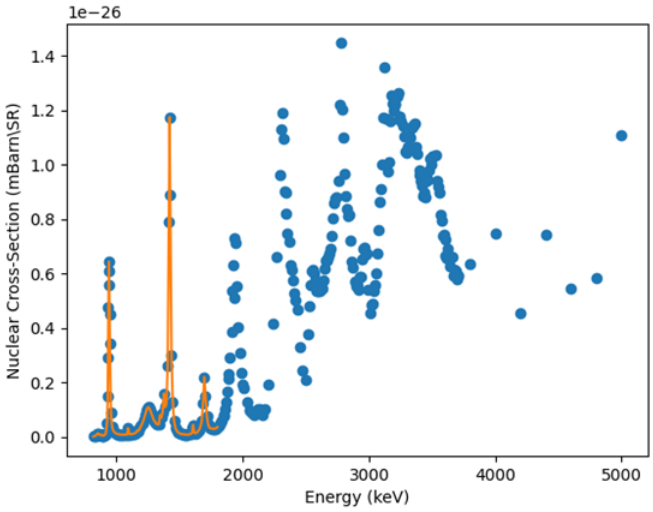


Figure 4.3: Interpolation (Orange) of the 110 keV Fluorine-19 nuclear cross-section (Blue)

cross-sections. These values can be used in equation 4.5.

$$\int_0^{E_0} \sigma(E)/S_m(E) = \sum_k [(1/(S_{mk}) \cdot \sigma(E_{k,k+1}) \cdot (E_{k+1} - E_k)] \tag{4.5}$$

Where $\sigma(E_{k,k+1})$ is the mean between σ_k and σ_{k+1}. With the calculation of this integral, it becomes simple to reorganize equation 4.1 and isolate the mass fraction. The mass fraction can be converted to parts per million by using the conversion factor of $1\% = 10000$ppm. A screenshot of the user interface of the program is included in Figure 4.4.

4.2 Standards Method

The standards method outlines the procedure for creating a group of samples with known concentrations of fluorine to compare to samples with unknown concentrations. By creating a set of samples with varying concentrations that cover the range of potential concentration values, experimental gamma-ray yields at the 110 keV and 197 keV peaks of the unknown sample can be compared to the standard data to estimate concentration. This

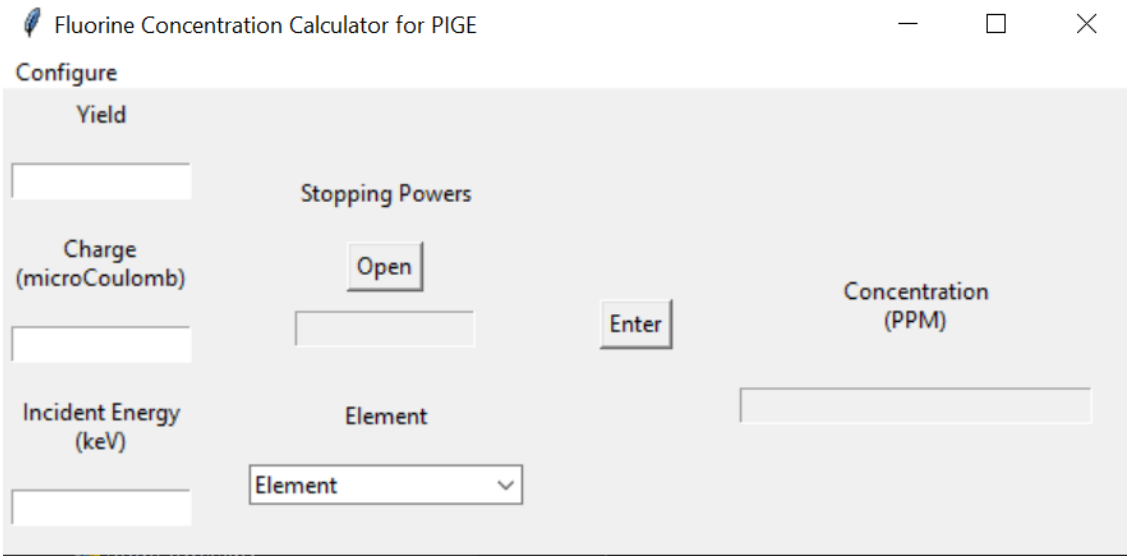


Figure 4.4: Screenshot of python application for calculating fluorine concentration.

method requires a process to create standards, prepare samples for use with the standards, and finally a way to compare standards and samples of interest. These methods are detailed below along with the creation of a set of sodium fluoride (NaF) and cellulose standards.

4.2.1 Creation of Standards

For our standards method, we needed to develop a group of samples with known fluorine concentrations that covered a range similar to what we should expect of our samples of interest. To create the samples we weighed out known amounts of NaF and Cellulose binder using a high sensitivity scale to total 0.5 grams. The equation to calculate the concentration of fluorine using sodium fluoride is shown in equation 4.6.

$$\frac{\frac{19}{42}(\text{Mass NaF})}{\text{Total Sample Mass}} \cdot (1 \times 10^6) \tag{4.6}$$

These two powders were then homogeneously mixed over a period of 10 minutes. This mixing was conducted by shaking the samples by hand but future standards will be mixed using an electronic mixer. We then applied our pellet creation method described above.

In total, we created 11 standards with concentrations ranging from 10,000 to 75,000 ppm Fluorine. Standard specification along with the uncertainty in the concentration values are included in table 4.1. Uncertainty was calculated using relative standard uncertainty shown in equation 4.7.

$$\text{Uncertainty} = \left(\frac{1000 \cdot 10^{-6}}{m_{NaF}} + \frac{1000 \cdot 10^{-6}}{m_{total}} \right) \cdot \text{Concentration} \tag{4.7}$$

Mass NaF (g)	**Total Mass (g)**	**Concentration F (PPM)**	**Uncertainty in Concentration (PPM)**
0.0222	0.9997	10046	463
0.03427	1.00762	15386	464
0.0441	0.9989	19972	473
0.06642	0.93374	30042	484
0.06642	0.93374	30042	484
0.07717	1.00719	34661	484
0.07717	1.00719	34661	484
0.08854	1.00104	40012	492
0.08854	1.00104	40012	492
0.10008	1.00333	45124	496
0.10008	1.00333	45124	496
0.11062	1.00018	50033	502
0.11062	1.00018	50033	502
0.16628	1.01204	74327	520

Table 4.1: NaF and Cellulose Standard Specifications

4.2.2 Samples of Interest

As is shown in the general formula for calculating nuclear yield through PIGE, the number of emitted gamma-rays is inversely proportional to the integration of the stopping power of the material. This means the composition of the sample has a tangible effect on the experimental yield. In the standards method, this result is important to keep in mind since the goal is to compare the experimental yield of an unknown sample to a fixed group of standards. As a result, it is crucial that the composition of the standard is nearly the same as

the composition of the sample of interest. This can be done in two ways. First, the standard can be created using an uncontaminated version of the sample of interest. For example, a dirt sample from a site that is unaffected by PFAS can be used to create standards to compare to other dirt samples. The second method is to combine a sample of interest with a similar base as the standard. This is the approach more commonly used at the UCIBAL as it allows us to use a singular set of standards for a wider variety of samples.

4.2.3 Linear fit of cellulose-based standards

The created standards were experimentally run on to determine gamma-ray yields at the respective 110 keV and 197 keV Fluorine peaks. The resulting counts were normalized to collected charge over the same period and the background was removed using experimental results of a purely cellulose target. Uncertainty was calculated for the experimental yields using a fixed 10% assumed error due to charge integration. The experimental results are included in table 4.2.

Concentration F (PPM)	**Uncertainty in Concentration (PPM)**	**Yield/μC**	**Uncertainty in Yield**
10046	462	437	43
15386	464	551	55
19972	472	865	86
30042	484	1426	142
30042	484	1262	126.
34661	483	1400	140
34661	483	1842	184
40012	491	2085	208
40012	491	1926	192
45124	495	2217	221
45124	495	1991	199
50033	502	2361	236
50033	502	2436	243
74327	520	3767	376

Table 4.2: PIGE data taken on NaF and Cellulose Standards

The results were then graphed as concentration in ppm to normalized gamma-ray yield. A linear fit was deduced and shown in Figure 4.5. An assumed intercept of 0 was used since 0 yield should correspond to a concentration of 0. The linear fit provides a formula for concentration as a function of counts/μC given by

$$\text{Concentration} = 20.77 \cdot (\text{Counts}/\mu C). \tag{4.8}$$

The equation fits the data well with an associated R^2 term of 0.992. While the error bars for the standards are higher than we would like, this is due to error throughout our experimental setup but, most notably due to our method of charge integration. We will discuss methods to reduce this error in future experiments in the results section.

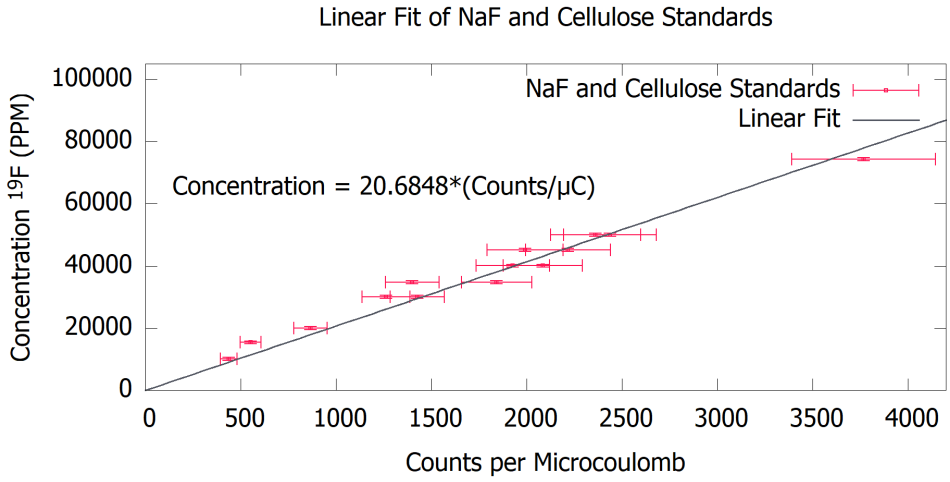


Figure 4.5: Plot of Table 4.2 as Concentration versus Yield per microcoulomb with linear fit of data.

4.2.4 Use of standards to predict future concentrations

The standards method can be used to compare samples of unknown concentrations with the same stopping power value. While the method cannot provide exact results, this method can accurately and efficiently produce a range of possible concentration values. For example,

an equation similar to equation 4.8 can be deduced to determine concentration ranges for a wide variety of samples. The limit of this method is our ability to create accurate standards as well as the inherent error of 10% that we associate with our experimental results. However, this method suits our goal of understanding how PFAS spreads throughout the environment. By being able to understand the range of PFAS concentration over a wide area, we will be able to make recommendations about the potential impact as well as understand how the chemicals spread over time. In many cases, a precise result will not be required and a general range will suffice to understand the extent of the problem.

Chapter 5

Results

5.1 Comparison to Standards

Preliminary results are described using the computational python code to predict the concentration of the NaF and cellulose standards. The goal was to compare the linear fit of the standards to a fit of the concentration values calculated using computational methods and the parameters of our experimental setup. For this initial test, we will use the yields presented in table 4.5, a SRIM calculation for cellulose $(C_6H_{10}O_5)_n$ and fluorine, and a detector efficiency calculated using the formula shown in equation 4.3. For calculating the absolute detector efficiency, we assume a relative efficiency of 20% at 110 keV. Unfortunately, these results were taken before the implementation of the new external beam facility so the solid angle must be calculated using the old setup where the targets were placed 6.5 cm away from the detector with a detector diameter of 45 mm. This gives us a solid angle of

$$\Omega = \frac{\pi(0.045/2)^2}{(0.065)^2} = 0.3764\, Sr.$$

Thus our calculated absolute detector efficiency is

$$\epsilon_{abs} = 0.2 \cdot 0.3764 = 0.07529\, Sr.$$

Plugging the necessary values into the application gives us the predicted concentration values shown in table 5.1.

Actual Concentration (PPM)	**Predicted Concentration (PPM)**
10046	8351.275
15386	10531.9
19972	16535.7998
30042	27324.894
30042	24102.71
34661	26738.137
34661	35189.298
40012	39822.44
40012	36797.597
45124	42351.108
45124	38043.785
50033	45104.37
50033	46536.58
74327	71963.404

Table 5.1: Predicted Fluorine concentration for NaF and cellulose standards using python application.

These results show that the code is predicting the concentration of fluorine within the samples relatively well. These calculations also match the linear fit deduced in figure 4.5. A comparison of the two linear fits is included in figure 5.1.

Systematically, it seems the code is underestimating the results. However, this could be a result of many factors including our value of detector efficiency. Improving these results will be discussed in the next section. A further test of the complete standards and computational methods will be conducted in the future for more samples and standards with a wide variety of compositions. While this preliminary study provides a great understanding of how well the computational code works, there are a few aspects of the necessary parameters that will be required for more precise results.

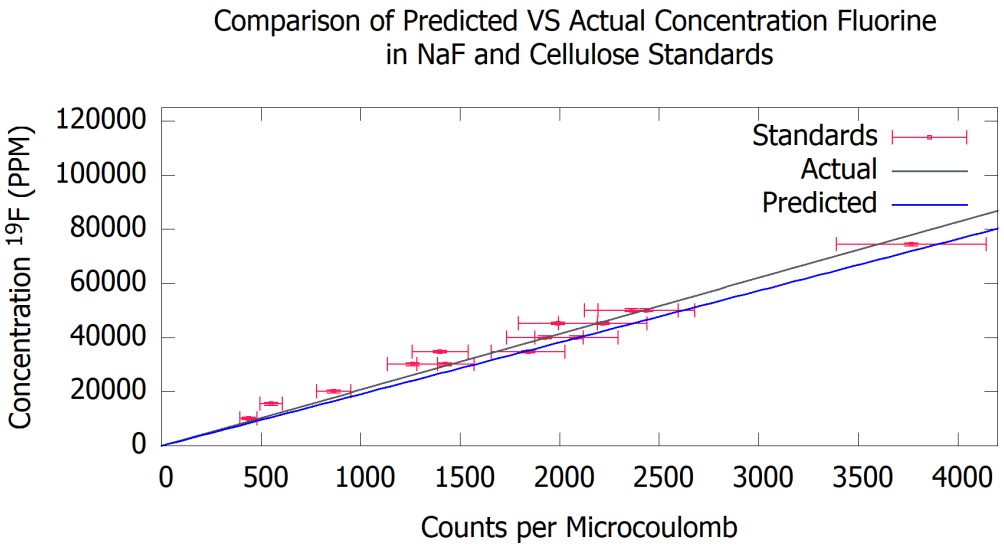


Figure 5.1: Comparison of predicted fluorine concentration using python code and linear fit of NaF and cellulose standards.

5.2 Improving Results in the Future

First, it will be important to have an exact value for detector efficiency rather than the estimation calculated using the relative efficiency. This calculation includes a relative efficiency that is not completely known for all possible energy values. It will be more beneficial to determine absolute efficiency using a source that emits gamma-rays with energy around 110 keV. The second area of improvement will be to implement the new external beam facility for data collection. The new facility aims to increase the accuracy of detecting gamma rays and eliminate potential sources of background. This will make experimental results more reliable for calculating concentrations. The final improvement will rely on the continued research in understanding nuclear cross-sections. Currently, nuclear cross-sections are mostly experimental and do not cover a wide energy range; most often covering higher energies well above 1800 keV. Another issue with these results is that the results are angular dependent. This means they rely on the relative angle between the proton beam and the detector. In our experimental setup, the relative angle is 90 degrees. However, the cross-

section data available uses an angle of 135 degrees. While we found that by looking at other results the angle dependence had a small effect on the result, it is possible that the difference could have a significant effect on our data. A current tool called SigmaCalc aims to calculate nuclear cross-sections for any arbitrary angle. However, currently, the tool can only produce rough cross-sections up to about 1700 keV for Fluorine 19, just below our incident energy [17]. As more precise cross-sections can be calculated, especially for the 110 keV fluorine peak, the more accurate the computational method of calculating concentration can become. Even so, the computational code in its current state looks to be able to work alongside the standards-based method in order to make accurate recommendations on PFAS within the environment through examining fluorine concentrations.

Chapter 6

Summary and Applications

Per- and polyfluoroalkyl substances remain a threat to both environmental well-being and human health. Humans are directly exposed to this group of man-made chemicals as they can be found in an extensive range of day-to-day goods and spread across environmental areas. More concerning, they are prevalent within the bloodstream of a significant number of U.S. citizens [5]. To better understand how to reduce the concentration of PFAS within humans, it will be vital to know where PFAS is present in high concentrations. While chemical methods have been described by the EPA to determine concentrations of specific chemicals, there is a need to quantify concentrations of large groups of PFAS quickly and effectively [9]. PIGE has been shown as an effective tool to accomplish this goal by detecting total fluorine, a key identifier of PFAS [10, 11]. Our goal at the UCIBAL is to develop methods of quantifying the total concentration of fluorine in order to aid in the screening process of PFAS affected areas in New York State and the greater northeast region.

In this thesis, we have described all parts of the necessary methods to collect and format data, create samples and standards, and compute concentrations of fluorine within samples of interest. Fluorine concentrations are calculated using two methods: the standards method and the computational method. Both methods work for a variety of samples and all compounds of the PFAS family. The preliminary results show that both the standards-based method and computational method are in agreement and are ready to be tested for full use at the UCIBAL. While this agreement is promising, we hope to continue improving

upon these methods in order to have more precise results. This includes improvements in our calculation of the detector efficiency as well as keeping up with the work being done to improve nuclear cross-section results. The defined procedure will allow us to begin work towards screening samples from PFAS impacted areas with high fidelity.

As PFAS continues to serve as a threat to the environment and human health, it will be important to understand the sources of PFAS and methods to protect humans from potential exposure. At the UCIBAL, one of our main areas of future interest is PFAS exposure due to fire foam. Fire foam is known to contain a high concentration of PFAS and is widely used to combat high-intensity fires. While the product serves as a great method to suppress large-scale fires, there have been numerous recorded times where significant amounts have been released into the environment due to training exercises and accidents. Specifically, one of our future projects hopes to quantify the spread of PFAS due to the release of fire fighting foam at Bradley International Airport in 2019. In this accident, 25000 gallons of fire fighting foam was released and ended up in the Farmington River [18]. Once in the river, we believe the foam was spread along the banks and could potentially have lasting impacts on a wide stretch of adjacent parks and walking paths. With the development of the quantitative methods, it will be our goal to quantify the spread and determine if PFAS are still detectable and if so, are the levels hazardous to humans.

REFERENCES

[1] See for example: Julie Turkewitz, "Toxic 'Forever Chemicals' in Drinking Water Leave Military Families Reeling." The New York Times, 2019.

[2] Evelyn E. Ritter, et al., "PIGE as a screening tool for Per- and polyfluorinated substances in papers and textiles," Nuclear Instruments and Methods in Physics Research B **407**, 47–54 (2017).

[3] United States Environmentally Protection Agency, "Basic Information on PFAS," EPA.gov, 2019.

[4] E.M. Sunderland, X.C. Hu, C. Dassuncao, et al. "A review of the pathways of human exposure to poly- and perfluoroalkyl substances (PFASs) and present understanding of health effects," J Expo Sci Environ Epidemiol **29**, 131–147 (2019).

[5] United States Agency for Toxic Substances and Disease Registry, "PFAS in the U.S. Population", atsdr.cdc.gov, 2020.

[6] Sydney Evans, David Andrews,Tasha Stoiber, and Olga Naidenko, "PFAS Contamination of Drinking Water Far More Prevalent Than Previously Reported", Environmental Working Group, ewg.org, 2020.

[7] Hong BD, Joo RN, Lee KS, Lee DS, Rhie JH, Min SW, Song SG, Chung DY. 2016. Fluoride in soil and plant. Korean Journal of Agricultural Science **43** 522-536 (2016).

[8] E. Bombik, A. Bombik, and K. Rymuza. The influence of environmental pollution with fluorine compounds on the level of fluoride in soil, feed and eggs of laying hens in Central Pomerania, Poland. Environ. Monit. Assess. **192**, 178 (2020).

[9] United States Environmental Protection Agency, "PFAS Analytical Methods Development and Sampling Research",EPA.gov, 2020.

[10] Graham Peaslee, "Developing PIGE into a Rapid Field-Screening Test for PFAS", Environmental Restoration SERDP and ESTCP, serdp-estcp.org, 2019.

[11] INTERNATIONAL ATOMIC ENERGY AGENCY, "Development of a Reference Database for Particle Induced Gamma Ray Emission (PIGE) Spectroscopy", TECDOC Series, 2017.

[12] V. Manteigas, J. Cruz, M. Fonseca, A.P. Jesus, "ERYA–Profiling: A code for quantitative PIGE analysis of in-depth heterogeneous samples," Nucl. Instrum. and Methods in Phys. Res. B **502**, 142-149 (2021) .

[13] James F. Ziegler, The Stopping and Range of Ions in Matter (SRIM), SRIM.org, 2013.

[14] ORTEC, MAESTRO Multichannel Analyzer Emulation Software, ortec-online.com.

[15] IAEA, Ion Beam Analysis Nuclear Data Library (IBANDL), 2019.

[16] D.Bachiller Perea, Nucl. Instrum. Methods in Phys Res. B, **406**, 161 (2017) data retrieved from the IBANDL database, IAEA, 2019 at http://www-nds.iaea.org/ibandl/.

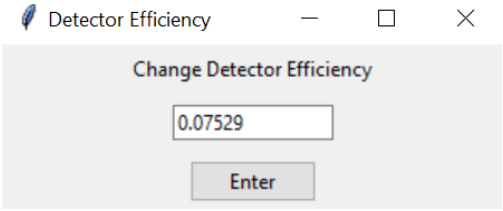
[17] Alexander Gurbich, Sigma Calc, Obninsk Institute for Nuclear Power Engineering, sigmacalc.iate.obninsk.ru.

[18] GREGORY B. HLADKY, "CONNECTICUT Firefighting foam spill in Farmington River has Connecticut residents concerned, worried about future contamination," Hartford Courant, 2019.

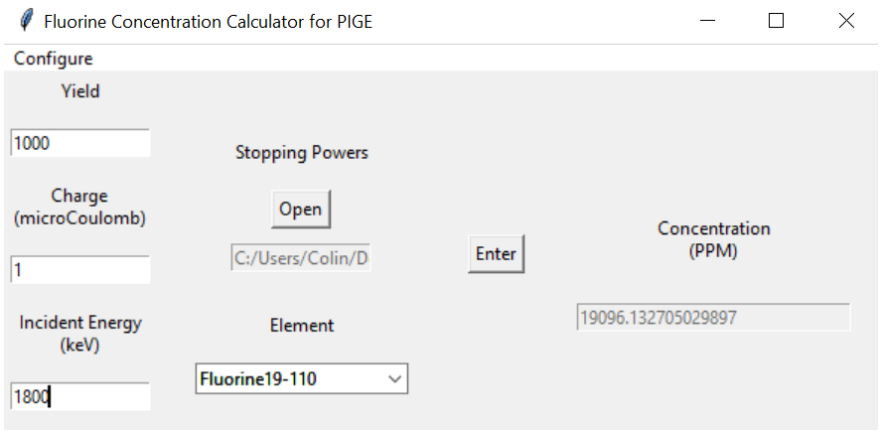
Python Code

.1 Instructions

The created python code is ready to be implemented as soon as it's downloaded. Upon opening, it is first necessary to configure detector efficiency to the desired value. To do so, open the configure menu, click on detector efficiency and write the desired value in the entry field. Hitting enter will save the value for future uses.

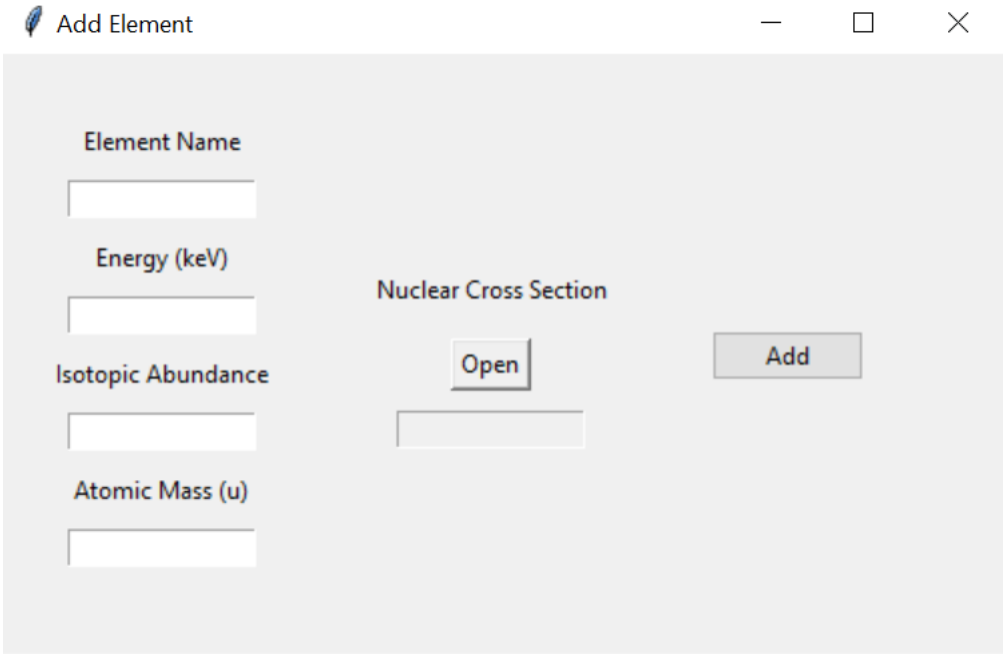


The next step is to input the necessary values of Yield, Charge, and Incident Energy. A valid number must be entered into each of these fields for the code to run. After entering valid numbers, a stopping power file must be entered from SRIM. The saved stopping powers from SRIM can be directly inputted by navigating to the correct location in the file menu. Finally, an element and its corresponding energy must be chosen from the element drop-down menu. If done correctly, hitting enter will result in a numerical value appearing in the concentration field. Using a detector efficiency of 0.07529, 1000 yield per microcoulomb at an incident energy of 1800 keV, a SRIM file of $C_6H_{10}O_5$ and fluorine, and finally Fluorine 19 at 110 keV, a concentration of 19096 should be produced.



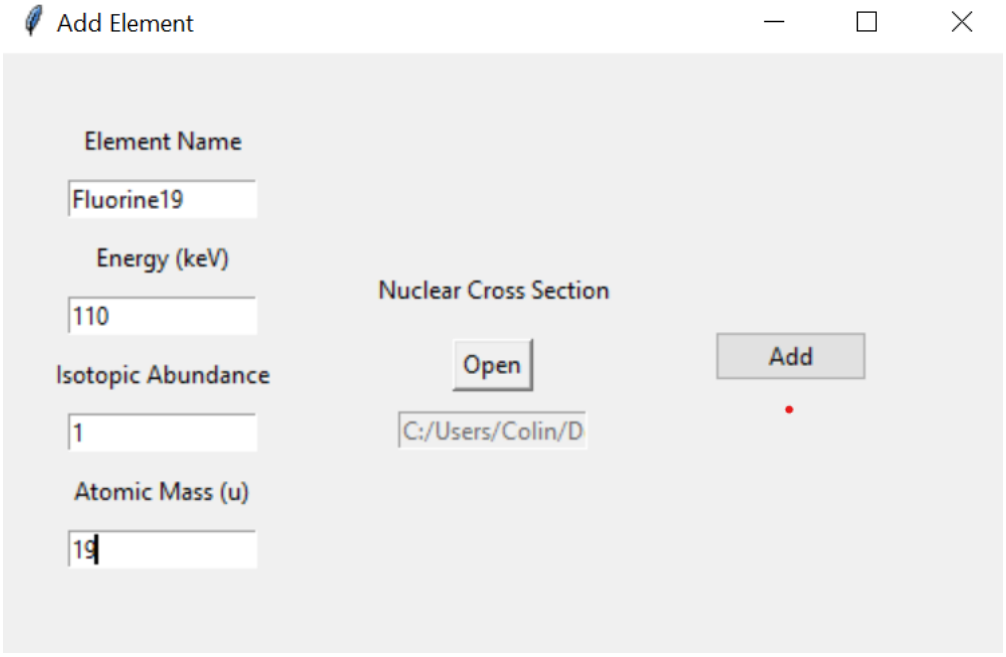
.2 Adding Element

In order to add an element to the code's database, open the configure menu and hit add element. This will bring up the field to add an element.



Enter the necessary information for element name, the gamma-ray energy of interest, the isotopic abundance of the element, and its atomic mass. Then, navigate to the location of the nuclear cross-section. The nuclear cross-section file should be formatted to just include

numerical values with energy (MeV) in the first column and cross-section (mb/sr) in the second column. Adding elements will save the element in the code's database. For the new element to show up in the element drop-down list of the calculator, the application must be closed and reopened. The new element will be available indefinitely for future use. An example for adding Fluorine-19 at 110 keV is included.



.3 Code

Included below is the raw python code for the concentration calculator. The application is available through contact with the UCIBAL.

File - C:\Users\Colin\PycharmProjects\pythonProject\GUI.py

```
 1 import pickle
 2 import tkinter as tk
 3 from tkinter import *
 4 from tkinter import ttk
 5 from tkinter import filedialog as fd
 6 from tkinter.messagebox import showinfo
 7
 8 import element
 9 import calcConc
10
11
12 class GUI(tk.Tk):
13     def __init__(self):
14         #Creates frame for GUI
15         super().__init__()
16         self.title('Fluorine Concentration Calculator
   for PIGE')
17         self.geometry = ('1000x500')
18         self.frame = Frame(self)
19         self.frame.pack()
20
21         global det_eff
22         det_eff = pickle.load(open("detEff.p", "rb"))
23
24         menubar = MenuBar(self)
25         self.config(menu=menubar)
26
27         #Creates frame for left most items
28         leftFrame = Frame(self)
29         leftFrame.pack(side=LEFT)
30
31         #Creates field for Yield input
32         label = Label(leftFrame, text="Yield")
33         label.pack()
34         self.yiel = tk.StringVar()
35         yieldEntry = Entry(leftFrame, width=15,
   textvariable = self.yiel)
36         yieldEntry.pack(padx=5, pady=15)
```

File - C:\Users\Colin\PycharmProjects\pythonProject\GUI.py

```
37
38          #Creates field for Charge input
39          label = Label(leftFrame, text="Charge\n(
    microCoulomb)")
40          label.pack()
41          self.chargeEntry = Entry(leftFrame, width=15)
42          self.chargeEntry.pack(padx=5, pady=15)
43
44          #Creates field for incident energy input
45          label = Label(leftFrame, text="Incident Energy\
    n(keV)")
46          label.pack()
47          self.incEn = tk.StringVar()
48          incEntry = Entry(leftFrame, width=15,
    textvariable = self.incEn)
49          incEntry.pack(padx=5, pady=15)
50
51          #Frame for middle items
52          middleFrame = Frame(self)
53          middleFrame.pack(side=LEFT)
54
55          #Creates field to open stopping power file
56          label = Label(middleFrame, text="Stopping
    Powers")
57          label.pack(padx=25, pady=15)
58          open_button = Button(
59              middleFrame,
60              text='Open',
61              command=self.stopFile
62          )
63          open_button.pack(expand=True)
64          self.stopField = tk.StringVar()
65          stopEntry = Entry(middleFrame, width=15, state=
    DISABLED,textvariable = self.stopField)
66          stopEntry.pack(pady=10)
67
68          #Creates menu to choose element
69          label = Label(middleFrame, text="Element")
```

File - C:\Users\Colin\PycharmProjects\pythonProject\GUI.py

```
 70         label.pack(padx=25, pady=15)
 71
 72         database = pickle.load(open("elements.p", "rb"
    ))
 73         self.listEle = list(database.values())
 74         listVals = [(val.getName() + "-" + str(val.
    getEnergy())) for val in self.listEle]
 75         self.Combo = ttk.Combobox(middleFrame, values=
    listVals, state="readonly")
 76         self.Combo.set("Element")
 77         self.Combo.pack(padx=25)
 78
 79         #frame for enter button
 80         enterFrame = Frame(self)
 81         enterFrame.pack(side = LEFT)
 82
 83         enter_button = Button(
 84             enterFrame,
 85             text='Enter',
 86             command=self.enter
 87         )
 88         enter_button.pack(expand=True,padx = 15)
 89
 90         #Frame for output
 91         rightFrame = Frame(self)
 92         rightFrame.pack(side = LEFT)
 93
 94         label = Label(rightFrame,text = "Concentration
    \n(PPM)")
 95         label.pack(padx = 25, pady= 25)
 96
 97         #output field
 98         self.concentration = tk.StringVar()
 99         concEntry = Entry(rightFrame, width = 30,
    state = DISABLED,textvariable = self.concentration)
100         concEntry.pack(padx = 20)
101
102
```

File - C:\Users\Colin\PycharmProjects\pythonProject\GUI.py

```
103     #Opens file explorer
104     def stopFile(self):
105         filetypes = (
106             ('text files', '*.txt'),
107             ('All files', '*.*')
108         )
109         self.filename=fd.askopenfilename(
110             title='Open a file',
111             initialdir='/',
112             filetypes=filetypes)
113 
114         showinfo(
115             title='Selected File',
116             message=self.filename
117         )
118         self.stopField.set(self.filename)
119 
120 
121 
122 
123     #Computes concentration on enter input
124     def enter(self):
125         elemPos = self.Combo.current()
126 
127         try:
128             yielval = float(self.yiel.get())
129             charge = float(self.chargeEntry.get())
130             incidentEner = float(self.incEn.get())
131 
132             if(self.stopField.get() != ""):
133                 self.concentration.set(calcConc.
    calculateMassF(yielval, self.listEle[elemPos], self.
    filename, incidentEner, charge, det_eff))
134         except ValueError:
135             self.concentration.set("Error:189E084")
136 
137 
138 #Creates drop down menu for configuring detector
```

File - C:\Users\Colin\PycharmProjects\pythonProject\GUI.py

```
138 efficiency and element database
139 class MenuBar(tk.Menu):
140     def __init__(self, parent):
141         tk.Menu.__init__(self, parent)
142 
143         fileMenu = tk.Menu(self, tearoff=False)
144         self.add_cascade(label="Configure",underline=0
    , menu=fileMenu)
145         fileMenu.add_command(label="Detector
    Efficiency",underline=1, command=self.detEff)
146         fileMenu.add_command(label = "Add Element",
    underline=1,command=self.addEle)
147 
148     #Opens detector efficiency window
149     def detEff(self):
150         window = Window(self)
151         window.grab_set()
152 
153     #Opens add element window
154     def addEle(self):
155         window = Window2(self)
156         window.grab_set()
157 
158 #New window to change detector efficiency
159 class Window(tk.Toplevel):
160     def __init__(self, parent):
161         super().__init__(parent)
162 
163         self.geometry('300x100')
164         self.title('Detector Efficiency')
165 
166         ttk.Label(self,text="Change Detector
    Efficiency").pack(expand=True)
167 
168         self.detEff = tk.StringVar()
169         changeDet = ttk.Entry(self,width = 15,
    textvariable=self.detEff)
170         changeDet.insert(END,str(det_eff))
```

File - C:\Users\Colin\PycharmProjects\pythonProject\GUI.py

```
171         changeDet.pack(expand=True)
172 
173         ttk.Button(self,
174                 text='Enter',
175                 command=self.changeDetEff).pack(expand
    =True)
176 
177     def changeDetEff(self):
178         global det_eff
179         try:
180                 det_eff = float(self.detEff.get())
181         except ValueError:
182             pass
183         pickle.dump(det_eff, open("detEff.p", "wb"))
184         self.destroy()
185 
186 #New window to add elements to the database
187 class Window2(tk.Toplevel):
188     def __init__(self, parent):
189         super().__init__(parent)
190 
191         self.geometry('500x300')
192         self.title('Add Element')
193         self.frame = Frame(self)
194         self.frame.pack()
195 
196         leftFrame = Frame(self)
197         leftFrame.pack(side=LEFT)
198 
199         ttk.Label(leftFrame,text = "Element Name").
    pack(padx = 25)
200         self.elName = tk.StringVar()
201         entname = Entry(leftFrame,width = 15,
    textvariable=self.elName)
202         entname.pack(pady = 10)
203 
204         ttk.Label(leftFrame, text="Energy (keV)").pack
    (padx=25)
```

File - C:\Users\Colin\PycharmProjects\pythonProject\GUI.py

```
205         self.ener = tk.StringVar()
206         enerEn = Entry(leftFrame, width=15,
    textvariable=self.ener)
207         enerEn.pack(pady=10)
208 
209         ttk.Label(leftFrame, text="Isotopic Abundance"
    ).pack(padx=25)
210         self.isoAb = tk.StringVar()
211         isoAbEn = Entry(leftFrame, width=15,
    textvariable=self.isoAb)
212         isoAbEn.pack(pady=10)
213 
214         ttk.Label(leftFrame, text="Atomic Mass (u)").
    pack(padx=25)
215         self.mass = tk.StringVar()
216         massEn = Entry(leftFrame, width=15,
    textvariable=self.mass)
217         massEn.pack(pady=10)
218 
219         middleFrame = Frame(self)
220         middleFrame.pack(side = LEFT)
221 
222         ttk.Label(middleFrame, text="Nuclear Cross
    Section").pack(padx=25, pady=15)
223         open_button = Button(
224             middleFrame,
225             text='Open',
226             command=self.crossFile
227         )
228         open_button.pack(expand=True)
229         self.crossField = tk.StringVar()
230         crossEn = Entry(middleFrame, width=15, state=
    DISABLED, textvariable=self.crossField)
231         crossEn.pack(pady=10)
232 
233 
234         rightFrame = Frame(self)
235         rightFrame.pack(side = LEFT)
```

File - C:\Users\Colin\PycharmProjects\pythonProject\GUI.py

```
236
237         ttk.Button(rightFrame,
238                 text='Add',
239                 command=self.addElement).pack(padx =
    25)
240
241     def crossFile(self):
242         filetypes = (
243             ('text files', '*.txt'),
244             ('All files', '*.*')
245         )
246         self.filename=fd.askopenfilename(
247             title='Open a file',
248             initialdir='/',
249             filetypes=filetypes)
250
251         showinfo(
252             title='Selected File',
253             message=self.filename
254         )
255         self.crossField.set(self.filename)
256
257     def addElement(self):
258         try:
259             energy = float(self.ener.get())
260             isoAbundance = float(self.isoAb.get())
261             mass = float(self.mass.get())
262
263             element.addElement(self.elName.get(),
    energy,self.crossField.get(),isoAbundance,mass)
264
265         except ValueError:
266             print("error")
267             pass
268         self.destroy()
269
```

File - C:\Users\Colin\PycharmProjects\pythonProject\main.py

```
1 from GUI import GUI
2
3 if __name__ == "__main__":
4     app = GUI()
5     app.mainloop()
```

File - C:\Users\Colin\PycharmProjects\pythonProject\reader.py

```
 1 def readLines(fileLoc):
 2     with open(fileLoc) as f:
 3         contents = f.readlines()
 4     return contents
 5
 6 def read(fileLoc):
 7     with open(fileLoc) as f:
 8         contents = f.read()
 9     return contents
10
11
12
13
14
15
```

File - C:\Users\Colin\PycharmProjects\pythonProject\element.py

```
 1 import CrossSections
 2 import pickle
 3
 4
 5 def addElement(nameElement,energy,crossLoc,abundance,
   mass):
 6     newElement = element(nameElement,energy,crossLoc,
   abundance, mass)
 7     elements = pickle.load( open( "elements.p", "rb"
    ) )
 8     nameEner = str(newElement.getEnergy())+ newElement.
   getName()
 9     if nameEner not in elements:
10         elements[nameEner] = newElement
11         pickle.dump(elements,open("elements.p","wb"))
12     else:
13         print("Element already in database")
14
15 class element:
16     CrossSectionData = []
17     isotopicAbundance = 0
18     atomicMass = 0
19     name = ""
20     energy = 0
21
22     def __init__(self,name,energy,crossLoc,abundance,
   mass):
23         self.CrossSectionData = CrossSections.
   formatCrossList(crossLoc)
24         self.isotopicAbundance = abundance
25         self.atomicMass = mass
26         self.name =  name
27         self.energy = energy
28
29     def getCross(self):
30         return self.CrossSectionData
31     def getIsoAbund(self):
32         return self.isotopicAbundance
```

File - C:\Users\Colin\PycharmProjects\pythonProject\element.py

```
33      def getAtmM(self):
34          return self.atomicMass
35      def getName(self):
36          return self.name
37      def getEnergy(self):
38          return self.energy
39
40
41
```

File - C:\Users\Colin\PycharmProjects\pythonProject\calcConc.py

```
 1 import NumericalIntegrator
 2 import StopPowers
 3
 4 nAV = 6.02214*(10**(23))
 5 #charge of electron in Coulombs
 6 electronCharge = 1.60217662*(10**(-19))
 7
 8 #Calculates concentration
 9 def calculateMassF(nucYield,element,stopLoc,incEner,
   charge,detEff):
10     NP = chargeToNP(charge*10**(-6))
11     stopPow = StopPowers.formatSrimData(stopLoc)
12     crossSec = element.getCross()
13     mass = element.getAtmM()
14     isoAbun = element.getIsoAbund()
15     sumIntegr = NumericalIntegrator.calcIntegr(1800,
   stopPow, crossSec)
16     fm = (nucYield*mass)/(isoAbun*nAV*NP*sumIntegr*
   detEff)
17
18     return fm*100*10000
19
20
21 def chargeToNP(charge):
22     return charge/electronCharge
23
24
25
26
27
```

File - C:\Users\Colin\PycharmProjects\pythonProject\StopPowers.py

```
 1 import numpy as np
 2 import reader
 3
 4 MEV = 1000
 5 def formatSrimData(stopLoc):
 6     strData = reader.read(stopLoc)
 7     start = strData.find('--------------')+73
 8     end = strData.find(
   "----------------------------------------------------
   -----")
 9     strData = strData[start:end - 1]
10     listOfLineVals= [line.split() for line in strData.
   split('\n')]
11     x = []
12     y = []
13     for line in listOfLineVals:
14         x.append(formatEnergyVal(line[0],line[1]))
15         y.append((formatStopVal(line[2])+formatStopVal(
   line[3]))*10**(6))
16     xarray = np.array(x)
17     yarray = np.array(y)
18     return xarray,yarray
19
20 def formatStopVal(val):
21     exp = val.split("E")
22     return(float(exp[0])*(10**float(exp[1])))
23
24 def formatEnergyVal(val,unit):
25     if unit == "keV":
26         return float(val)
27     if unit == "MeV":
28         return float(val) * MEV
```

File - C:\Users\Colin\PycharmProjects\pythonProject\CrossSections.py

```
 1 import numpy as np
 2
 3 import reader
 4
 5 def formatCrossList(crossLoc):
 6     x = []
 7     y = []
 8     text = reader.read(crossLoc)
 9     strData = text
10     listOfLineVals = [line.split() for line in strData.
   split('\n')]
11     for line in listOfLineVals:
12         x.append(float(line[0])*1000)
13         y.append(float(line[1])*10**(-27))
14     xarray = np.array(x)
15     yarray = np.array(y)
16
17     return xarray,yarray
```

File - C:\Users\Colin\PycharmProjects\pythonProject\NumericalIntegrator.py

```
 1 import numpy as np
 2 from scipy import interpolate
 3
 4 def closestPos(x, energyVal):
 5     index = np.argmax(x >= energyVal)
 6     return index
 7
 8
 9 def inter(minEn,incEn,x,y):
10     index = closestPos(x, incEn)
11     temp = interpolate.interp1d(x[:index+1],y[:index +
   1])
12     xNew = np.arange(minEn,incEn,1)
13     yNew = temp(xNew)
14
15     return xNew,yNew
16
17
18 def calcIntegr(incEner,stopPow,crossSec):
19     stopX, stopY = inter(crossSec[0][0], incEner,
   stopPow[0], stopPow[1])
20     crossX, crossY = inter(crossSec[0][0], incEner,
   crossSec[0], crossSec[1])
21     sumIntegr = 0
22     for x in range(len(stopX)-1):
23         toAdd = (1/stopY[x]) * ((crossY[x]+crossY[x+1
   ])/2) * (crossX[x+1]-crossX[x])
24         sumIntegr = sumIntegr + toAdd
25     return sumIntegr
26
27
28
29
```