Gamma Ray Production Cross Sections of Proton Bombardment of Fluorine for Light Element Analysis and Depth Profiling

by

Monica Pham

SUBMITTED TO THE DEPARTMENT OF NUCLEAR SCIENCE AND ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN NUCLEAR SCIENCE AND ENGINEERING

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2019

Signature of Author:_____

Monica Pham
Department of Nuclear Science and Engineering
May 18, 2018

Certified by:_____

Zachary Hartwig
John C. Hardwick Assistant Professor of Nuclear Science and Engineering
Thesis Supervisor

Accepted by:_____

Michael Short
Assistant Professor of Nuclear Science and Engineering
Chairman, NSE Committee for Undergraduate Students

Gamma Ray Production Cross Sections of Proton Bombardment of Fluorine for Light Element Analysis and Depth Profiling

by

Monica Pham

Submitted to the Department of Nuclear Science and Engineering on May 18, 2018 in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Nuclear Science and Engineering

Abstract

Due to the interdependence of fusion reactor performance and the surface conditions of the plasma facing components (PFC), diagnostic techniques that monitor the conditions of PFCs are useful. One ion beam analysis (IBA) diagnostic technique utilizes particle induced gamma emission (PIGE) analysis to diagnose the conditions of PFCs. Gamma production differential cross section data is essential to PIGE because it provides a way predict and interpret data. Fluorine may be a candidate for this technique because it is low-Z and not typically intrinsic to fusion reactors. However, differential cross section measurements for ^{19}F(p, $\alpha\gamma$)^{16}O do not exist yet. Differential cross section measurements for the ^{19}F(p,$\alpha\gamma$)^{16}O reaction have not been measured primarily because of the difficulty of deconvolving overlapped peaks in the gamma ray spectra of the reaction. This work presents, for the first time, the 6.13 MeV gamma production differential cross section for the ^{19}F(p,$\alpha\gamma$)^{16}O reaction in the 840 to 1520 keV energy range. Although there exists gamma yield data for this reaction in Dababneh et al. [1], yield measurements can only be used as a comparison or for relative measurements. Additionally, previous yield measurements relied on integrating the entire spectrum with approximate bounds of 4.5 to 9 MeV likely due to overlap of peaks. In this work, yield of the 6.13 MeV full energy peak was obtained by deconvolving overlapping peaks by utilizing the constant ratio of the double escape and the single escape peaks. The subsequent yield curve was converted to differential cross section. The presented differential cross section measurements expands current knowledge and contributes to diagnostic techniques that utilize gamma production reactions.

Thesis Supervisor: Zachary Hartwig

Title: John C. Hardwick Assistant Professor of Nuclear Science and Engineering

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The performance of a fusion reactor heavily depends on the surface conditions of plasma facing components (PFCs). The surface of PFCs are affected by erosion, redeposition, fuel retention, surface impurity accumulation, and material mixing due to plasma-material interactions. The resulting high-Z contaminants in the surfaces of the PFCs create radiation loses in the plasma. Because of the interdependence between plasma interactions and surface conditions, it is important to monitor the surface conditions of PFCs. Historically, surface diagnostic techniques including ion beam analysis (IBA) methods have only been ex situ due to geometry restrictions inside a fusion reactor. Accelerator-based in situ Materials Surveillance (AIMS) is a diagnostic technique developed at MIT designed to study the material evolution of of surface layers of PFCs [3]. This technique is shown in Fig. 1-1

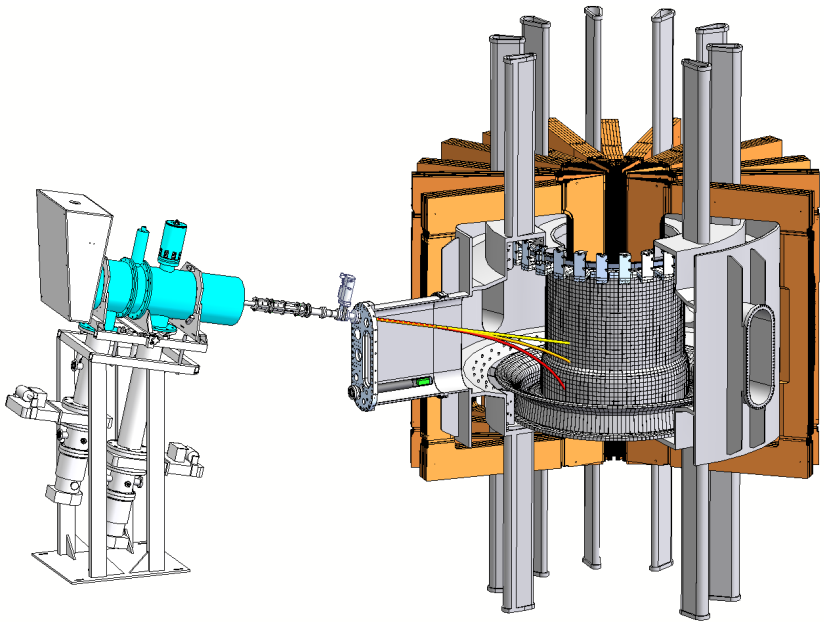

Figure 1-1: This simulated model of AIMS shows the accelerator-based technique used in situ to diagnose PFC surface conditions from Hartwig et al. [3].

A new technique based on AIMS being developed at MIT utilizes in situ ion beam analysis and ion implantation in PFCs to measure high-Z changes in PFCs [4]. This is a novel adaption of IBA tracks implanted depth markers through gamma-producing nuclear reactions. It uses a linear accelerator and gamma detectors to measure changes in PFC surface composition. This technique requires low-Z and gamma-producing isotopes that are not intrinsic to fusion reactor designs [5]. Fluorine may be an element of interest because it is low-Z making it suitable for existing ion beam analysis techniques such as particle induced gamma emission (PIGE) analysis and is not often found in the material components of fusion reactors.

PIGE is an IBA technique that can be used to detect low Z isotopes in a surface [2]. Material analysis using PIGE relies on measuring gamma production from a sample and comparing this to an expected yield. This technique is advantageous because it has better resolution than other IBA techniques. However, this technique is limited by the knowledge of differential cross section data [6].

Differential cross section data has not been recorded for ^{19}F(p,$\alpha\gamma$)^{16}O, a potentially important fluorine reaction that can be used in PIGE analysis. Without knowl-

edge of the differential cross section, PIGE is unable to be performed without another standard and this reaction is unable to be modeled. Although yield measurements for this reaction have been measured by several sources including Dababneh et al. [1], yield measurements cannot be applied to situations that vary in background, detector geometry, experimental setup, and the high energy continuum in the gamma spectrum. Measuring differential cross sections for this reaction would enable use of this reaction in varying situations and experimental setup. This paper presents differential cross section measurements for ^{19}F(p,$\alpha\gamma$)^{16}O for the proton energy range of 840 to 1520 keV. This energy range is corresponds to the energy range at which in situ diagnostic measurements would be performed. Additionally, the most proton beams are approximately 1 to 2 MeV.

Chapter 2

Background

2.1 Ion Beam Analysis Techniques

Nuclear reaction analysis (NRA) is a group of techniques that utilize nuclear reactions between a target and beam particle. NRA is most suitable for determining the concentration and depth profiling of light elements. The benefits of NRA include high isotopic selectivity, enhanced sensitivity for many nuclides, and nondestructive depth profiling [6].

Particle induced gamma emission (PIGE) is an NRA technique in which prompt gamma rays are detected from nuclear reactions induced by accelerated ions. Gamma yield from a known reaction can be measured when a sample is bombarded with a beam of ions at a fixed energy. PIGE is typically used when the chemical composition of the sample is known and can be used to detect low-Z isotopes in a surface [2],[6]. Changes in concentration and depth profile can be detected by measuring gamma yields of known reactions of an implanted layer underneath the surface [6]. A diagram of PIGE is shown in Fig. 2-1.

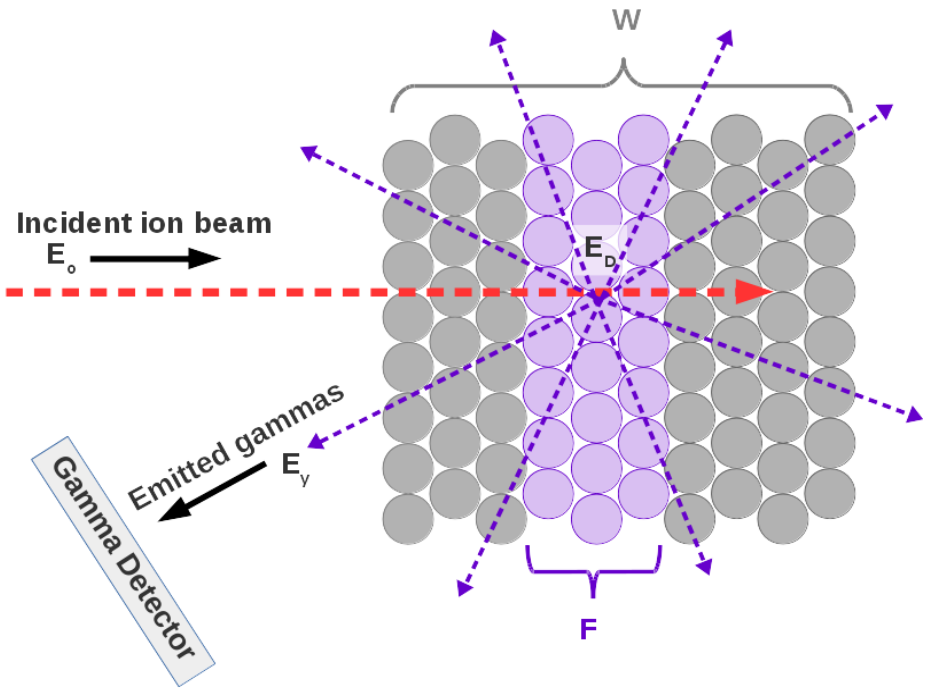

Figure 2-1: This diagram demonstrates PIGE performed on a layer of fluorine implanted in a tungsten target. The incident particle beam with energy E_0 reaches the fluorine layer at an energy of E_D. This particle beam induces emission of gamma rays of energy E_γ. The subsequent gamma production is detected. This figure is from Kesler [4].

PIGE is favorable over other NRA techniques due to the potential for depth profiling with better resolution by utilizing strong, narrow spikes in gamma yield called resonances. Resonances are useful to PIGE because they occur at discrete energies. This technique is also possible in situ because of the greater penetration of gamma rays than that of charged particles because gamma rays do not experience coulombic interactions. However, gamma rays do not provide direct information about the depth so gamma yields are typically compared to a proper standard. Because of this, the use of PIGE is limited by the knowledge of physical properties such as stopping power and differential cross sections [6].

2.2 Protons on Fluorine Reaction and Excited Oxygen States

The ^{19}F(p, $\alpha\gamma$)^{16}O reaction is an rearrangement reaction that results in alpha and gamma emission and ^{16}O in an excited state. This reaction includes three of five exit channels of the ^{19}F + p reaction. The three alpha channels are followed by gamma emission caused by nuclear transition directly to the ground state in ^{16}O [1].

Resonance reactions such as ^{19}F(p, $\alpha\gamma$)^{16}O can be useful for depth profiling. Resonances are a large and typically narrow increase in gamma yield at a discrete incident proton energy. Tracking the change in the center of the resonance can provide information about the depth of the target element is from the surface. A plot of gamma yield vs. incident proton energy displays the location resonances resulting from a resonance reaction. Yield data extracted from Dababneh et al. [1] is shown in Fig. 2-2.

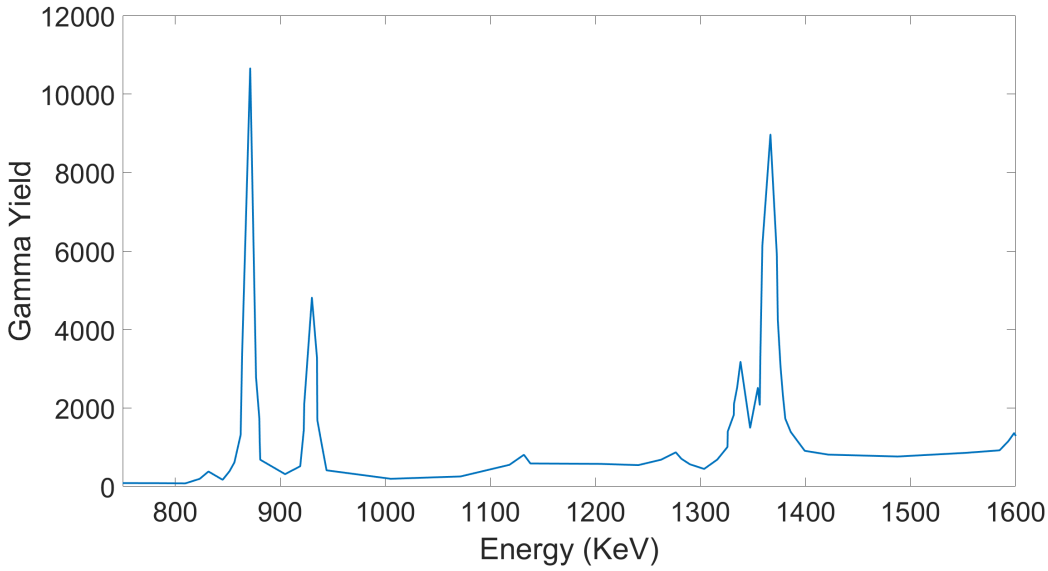

Figure 2-2: Thin target gamma yield data was extracted from Dababneh et al. [1] via WebPlotDigitzier and replotted

The gamma yield in Fig. 2-2 was obtained by integration of the entire spectrum from energies of approximately 4.5 to 9 MeV is given in Dababneh et al. [1]. An

example spectrum of ^{19}F(p,$\alpha\gamma$)^{16}O with these integration bounds is shown in Fig. 2-3.

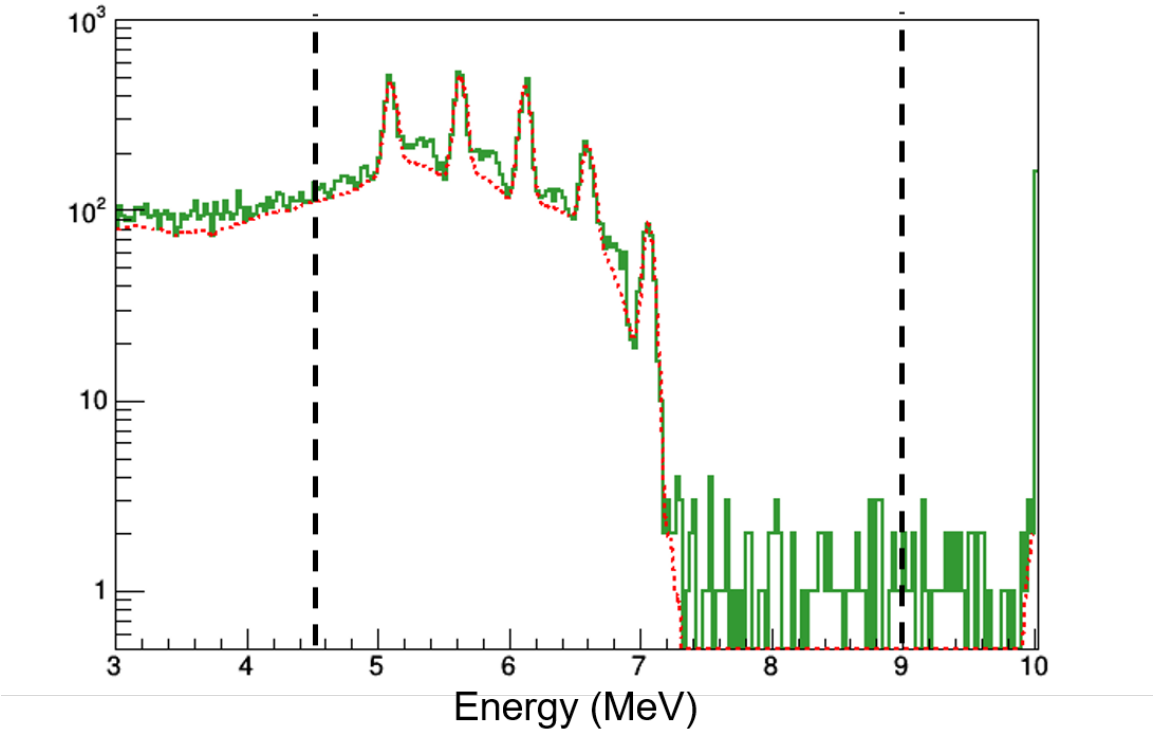

Figure 2-3: This example spectrum from ^{19}F(p,$\alpha\gamma$)^{16}O reaction is part of the data to be presented in Chapter 5. Approximate Dababneh et al. [1] bounds of integration of 4.5 to 9 MeV are shown by the black dashed lines.

Fig. 2-3 displays common features of spectra produced by ^{19}F(p,$\alpha\gamma$)^{16}O reaction. In this spectrum, there are five peaks visible. These peaks are defined and discussed in Section 4.4.1 and Fig. 4-2. Each of these peaks produces a Compton continuum which is smooth yield at a range of energy below that of the peak. This feature results from Compton scattering of characteristic gamma rays inside the detector. Above approximately 7.3 MeV is the high energy continuum. There are typically very few counts here because most of the counts result from cosmic radiation and neutrons from other nuclear reactions associated with the proton beam.

The bulk integral from 4.5 to 9 MeV does not account for changes in detector geometry and changes in background and therefore may not be able to be applied to varying situations. However, measuring the yield of the individual peaks resulting

from ^{19}F(p,$\alpha\gamma$)^{16}O is necessary to calculate differential cross section values. This will be discussed in further sections.

The energies of the characteristic gamma rays and relative dominance of the characteristic gamma full energy peaks are given in the following table [2].

Table 2.1: Gamma Energy of ^{19}F(p, $\alpha\gamma$)^{16}O and relative intensities from Barbour et al. [2]

	Relative Intensities (Percent)		
Gamma Energy (MeV)	γ_1 =6.13 MeV	γ_2 =6.92 MeV	γ_3 =7.12 MeV
Proton Energy (keV)			
668	81	0.3	19
872.1	68	24	8
902	>90	<5	<5
936	76	3	21
1283	74	8	18
1348	55	14	31
1371	87	8	5

The yield corresponding to each gamma line, γ_1 =6.13 MeV, γ_2 =6.92 MeV, and γ_3 =7.12 MeV, has a different intensity relative to the total yield of ^{19}F(p,$\alpha\gamma$)^{16}O at varying proton energies. For example, at a proton energy of 936 keV, the relative intensity of the 6.13 MeV gamma is 76 percent while the relative intensities of the 6.92 MeV and 7.12 MeV characteristic gamma rays are 3 percent and 21 percent respectively. Because the relative intensity of the 6.13 MeV characteristic gamma is the highest, it has the highest peak dominance. Table 2.1 shows that the 6.13 MeV characteristic gamma has the highest relative intensity and therefore the highest dominance at all energies listed.

Chapter 3

Differential Cross Section

Differential cross section describes the intrinsic rate at which scattered particles can be produced at a given angle expressed units of $[\frac{cm^2}{sr}]$. This measurement is similar to the likelihood or probability of a reaction occurring at a given solid angle and therefore can be used to determine the yield of a reaction. The differential cross section can be used to determine information about the target if information about the scattered particle is known. Knowledge of these values is fundamental to NRA and PIGE techniques.

3.1 Gamma Yield and Differential Cross Section

The number of detected particles or the gamma yield is given by the following relation:

$$Y = \frac{d\sigma}{d\Omega} \frac{t}{cos(\alpha)} \frac{Q}{q_p} N \epsilon \tau \Omega \tag{3.1}$$

where $\frac{d\sigma}{d\Omega}$ is the differential cross section in $[\frac{cm^2}{sr}]$, Q is the collected charge in $[C]$, and q_p is the charge of the accelerated particle in $[C]$, N is the volumetric density of of the target isotope in $[cm^{-3}]$, ϵ is the efficiency of the detector, τ is the transmission of the gamma rays through the material between, Ω is the solid angle in $[sr]$, t is the thickness of the target $[cm]$, and α is the angle between the incident beam and the normal to the target surface. In this equation, $\frac{t}{cos(\alpha)}$ is the path length of the

accelerated particle beam through the target. By rearranging Eq. (3.1), differential cross section can be calculated if the yield is known. This rearrangement is shown in Eq. (3.2).

$$\frac{d\sigma}{d\Omega} = \frac{Y\, cos(\alpha)}{N\, t\, \epsilon\, \tau\, \Omega} \frac{q_p}{Q} \tag{3.2}$$

In an experimental setup in which geometry and the chemical composition of the target is known, gamma yield can be measured and differential cross section can be calculated using Eq. (3.2).

Yield (Y) of a certain reaction is measured by utilizing gamma spectroscopy. Resulting particles or radiation is measured using a detector. Yield of a reaction can be obtained by integration of characteristic peaks. Error in this measurement is due to statistical error as well as errors in analysis methods. For example, if peaks are fit with Gaussian functions, the error in the fit contributes to the error in the yield measurement.

The efficiency (ϵ) and transmission (τ) are characteristic of the detector used to measure yield. Typically, efficiency and transmission curves are found by fitting a function to experimentally collected data points. Error in these values results from statistic error and error in fit.

Error in charge collection (Q) is due to errors in the collection method. These errors could result from false collection by software or experimental equipment.

Values of sample number density (N) and sample thickness (t) are determined by sample preparation. These values can be controlled. Error in these values may be due to error in sample preparation such as contamination and degradation of the sample over time.

Values of α and solid angle (Ω) are determined by experimental setup. These values can also be controlled. Error in these measurements result from the accuracy of placement of experimental equipment.

The charge of the accelerated particle (q_p) is known from the reaction. This value can be found in databases. The error in this value is small compared to experimental

error.

3.2 Challenges in Measuring Cross Sections for ^{19}F(p, $\alpha\gamma$)^{16}O

Measuring differential cross section data for ^{19}F(p, $\alpha\gamma$)^{16}O is difficult due to the close proximity of the characteristic gamma rays and their subsequent single and double escape peaks. Escape peaks occur when one or both of the 0.511 MeV gamma rays resulting from electron-positron pair annihilation from pair production escapes the detector and is therefore not absorbed. A single escape peak occurs when one 0.511 MeV gamma ray escapes resulting in a peak at an energy of $E_\gamma - 0.511MeV$. A double escape peak occurs when both 0.511 MeV gamma rays escapes resulting in a peak at an energy of $E_\gamma - 1.022MeV$. Full energy, single escape, and double escape energies are given for the ^{19}F(p, $\alpha\gamma$)^{16}O reaction in Table 3.1 [7].

Table 3.1: Gamma Energies of Resulting from Excited States of Oxygen in ^{19}F(p, $\alpha\gamma$)^{16}O

Peak	Full Energy Peak [MeV]	Single Escape [MeV]	Double Escape [MeV]
γ_1	6.13	5.62	5.11
γ_2	6.92	6.41	5.90
γ_3	7.12	6.61	6.10

As shown in Table 3.1, all characteristic gamma rays are high energy and very close in energy to each other. The high energy of the characteristic gamma rays is advantageous in spectroscopy and diagnostic techniques because there are no background peaks at this energy range. However, because they are all close in energy, it is necessary to use a detector with sufficient enough resolution to resolve each full energy peak and its escape peaks. Additionally, because detector efficiency is typically low at the high energy range of the characteristic gamma rays, the detector must also have sufficient efficiency to collect data on realistic timescales. The detector used to measure gamma yields for ^{19}F(p, $\alpha\gamma$)^{16}O needs to have high enough resolution to be able to resolve the peak while having high enough efficiency to be able to get

good statistics on each measurement. Three detectors were considered to take these ^{19}F(p,$\alpha\gamma$)^{16}O gamma yield measurements: sodium iodide (NaI) scintillation detector, lanthanum(III) bromide ($LaBr_3$) scintillation detector, and a High Purity Germanium (HPGe) detector. The HPGe's very high resolution would be able to resolve all the necessary peaks. However, the crystal size is small due to limitations in crystal growth of high purity germanium resulting in low efficiency. Because of the low efficiency, measurements would need to be lengthy. The NaI detector was not suitable for these measurements because although it has high efficiency, its poor resolution results in its inability to resolve the peaks of interest. A comparison of spectra taken with an NaI detector and $LaBr_3$ is shown in Fig. 3-1.

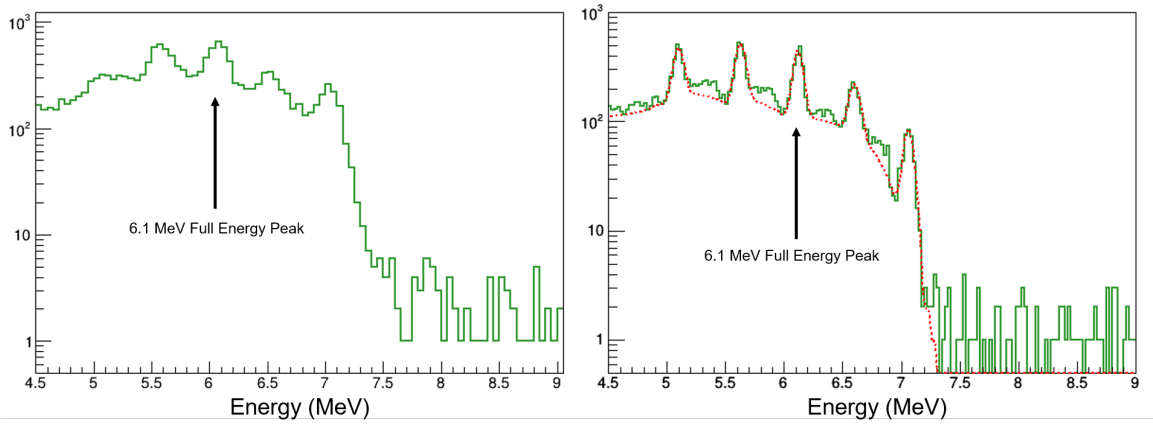

Figure 3-1: These spectra shows a comparison of ^{19}F(p, $\alpha\gamma$)^{16}O spectra taken with an NaI detector and an $LaBr_3$ taken at the same proton energy of 1180 keV. The spectrum taken with an NaI detector is pictured on the left. The spectrum taken with the $LaBr_3$ detector is pictured on the right. In the NaI spectrum, many of the full energy peaks and escape peaks are not resolved. The $LaBr_3$ is able to resolve all necessary peaks.

A $LaBr_3$ scintillation detector is suitable to collect gamma yield data for this reaction due to its high resolution and suitable efficiency. As shown in Fig. 3-1, the high resolution allowed for the peaks of interest to be well resolved from each other.

3.3 Double Escape Peak to Single Escape Peak Ratio and Deconvolution of Peaks

The double escape of the 7.12 MeV gamma is extremely close in energy to the 6.13 MeV gamma. These two peaks are virtually impossible to resolve with any existing detector. Because of this, it is necessary to deconvolve the peaks. Deconvolution is possible due to the fact that the ratio of the counts in the double escape and the counts in the single escape, R, is independent of the primary energy of the gamma ray and is only a function of detector geometry [8],[9]. This is because the 0.511 MeV gamma rays', resulting from subsequent positron-electron annihilation after a pair production event, attenuation losses are the same in the crystal of the detector. R is given by the following equation.

$$R = \frac{Quanta\ in\ the\ Double\ Escape\ Peak}{Quanta\ in\ the\ Single\ Escape\ Peak} \tag{3.3}$$

For a singular detector, R should be constant at all energies. R can be exploited to calculate the yield of the 7.12 MeV double escape peak. The peaks needed to determine counts in the 7.12 MeV double escape peak are shown in Fig. 3-2.

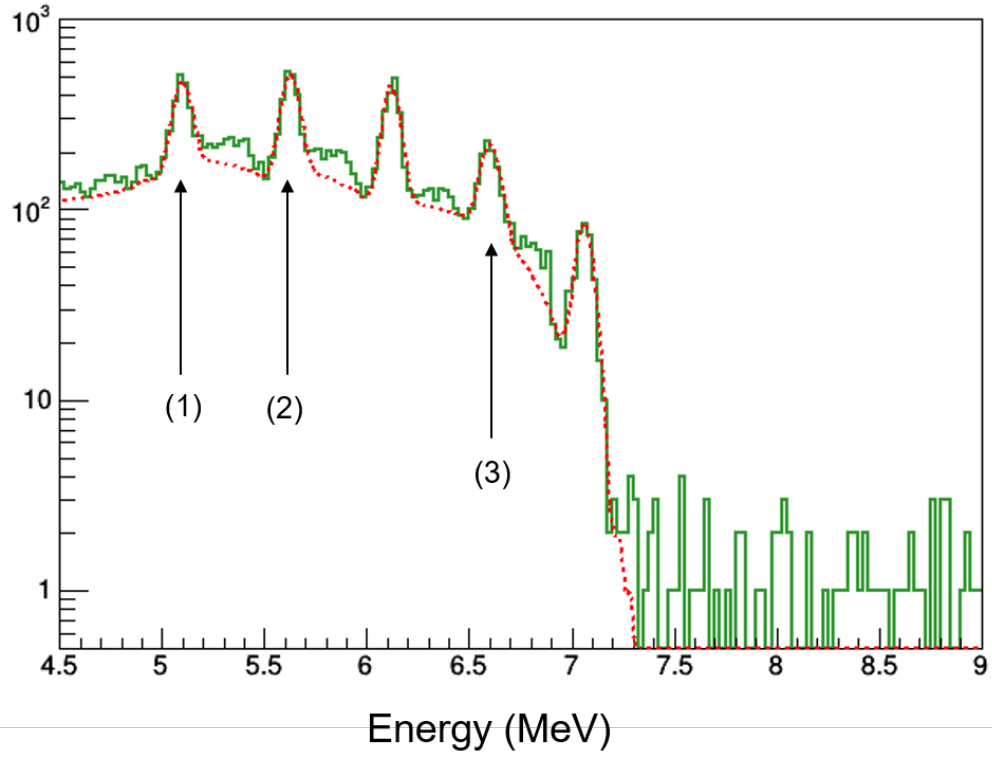

Figure 3-2: The labelled peaks are as follows (1) double escape peak of the 6.13 MeV gamma, single escape peak of the 6.13 MeV gamma, and (3) single escape peak of the 7.12 MeV gamma.

The yields of the peaks shown in Fig. 3-2 are needed to deconvolve the 6.13 MeV full energy peak and the 7.12 MeV double escape peak. This deconvolution method is possible because the 7.12 MeV single escape peak is unconvolved. If it was convolved, the counts of the 7.12 MeV double escape peak could not be calculated using the ratio, R.

The constant ratio, R, can be calculated using Eq. 3.3 and the yields of the 6.13 MeV double and single escape peaks. The ratio can be calculated using the double and single escape peaks of the 6.13 MeV gamma because they are unconvolved. Using this ratio, the number of counts of the 7.12 Mev double escape peak can be calculated with Eq. 3.4.

$$Y_{7.12\ MeV DoubleEscape} = R\, Y_{7.12\ MeV SingleEscape} \tag{3.4}$$

Using 3.4, the actual yield of the 6.13 MeV full energy peak can be calculated. The actual yield of the 6.13 MeV full energy peak is given by Eq. 3.5

$$Y_{actual} = Y_{measured} - R\, Y_{7.12\ MeV\ Single\ Escape} \tag{3.5}$$

where $Y_{measured}$ is the measured counts in the 6.13 MeV full energy peak, R is the ratio of counts of the double escape peak to counts of the single escape peak, and $Y_{7.12\ MeV\ Single\ Escape}$ is the measured yield in the 7.12 MeV single escape peak.

Chapter 4

Methods and Materials

4.1 Experimental Equipment

The Cambridge Laboratory of Accelerator Studies of Surfaces (CLASS) contains a General Ionex 1.7 MV tandem ion accelerator located at MIT. For consistency in this document, this accelerator will be referred to as the CLASS accelerator. The CLASS accelerator is a solid-state ion accelerator based on a parallel-fed cascade generator with a Cockroft-Walton voltage multiplier.

The CLASS accelerator uses two different ion sources. One of the sources is a National Electrostatics Corporation Source of Negative Ions Cesium Sputter (SNICS), which utilizes a solid cathode to produce ion beams. The second source used by the CLASS accelerator is the National Electrostatics Corporation RF Charge Exchange Negative ion source, which is also known as an Alphatross source. The Alphatross source produces negative ions from gaseous species by utilizing RF generated plasma to extract ions through a Rubidium charge exchange chamber. Though both sources can be used to produce protons beams, the sputter source was chosen because it can produce higher current.

Ultimately, an $LaBr_3$ detector was chosen to take yield measurements due to its high resolution. Waveforms were digitized using CAEN DT5790M Desktop Digitizer. Current collected on the target was measured using RBD 9103 USB Autoranging picoammeter. To ensure accurate current collection measurement, electron suppression

voltage of -500 V was used.

4.2 Experimental Setup

The sputter source using a TiH cathode was used to obtain a proton beam to perform PIGE analysis. Using the CLASS accelerator, a proton beam 3 mm in diameter was focused on a LiF target at an incident angle of 90° ($\alpha = 0°$) in an high vacuum chamber at pressure of approximately 10^{-6} torr. The $LaBr_3$ scintillation detector was placed at an angle of 90° with respect to the incident proton beam flush against a quartz viewport of the chamber. This setup is illustrated in Fig. 4-1.

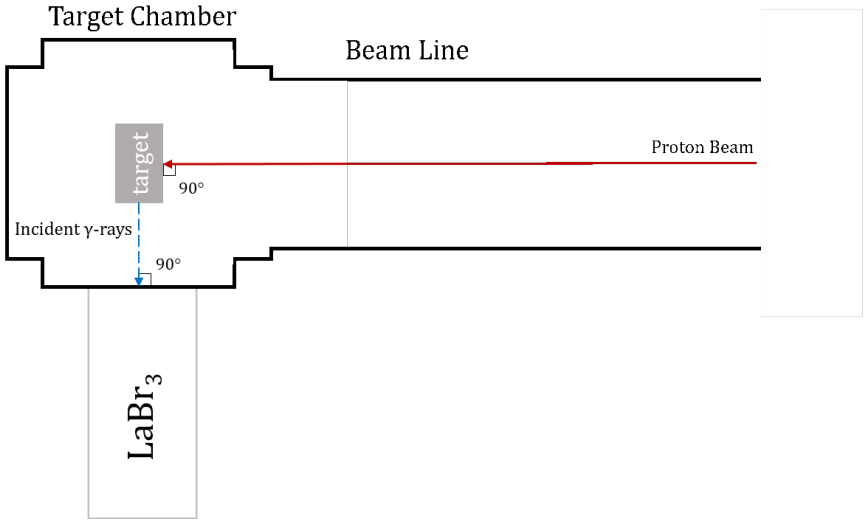

Figure 4-1: The experimental setup of detector, target, and beam is shown. A proton beam was at an incident angle of 90° with respect to the target. The target was 50 nm of LiF on the surface of a 3 mm thick molybdenum target.

The position of the detector was chosen because it optimized the number of quanta detected by the $LaBr_3$ detector. Due to restrictions outside the high vacuum chamber, the detector was placed against the viewport 20.59 cm away from the target. In this experimental setup, gamma yield was measured from proton beam energies of 840 keV to 1520 MeV. This energy range was chosen because it is an ideal range to perform ion beam analysis for fusion materials diagnostics techniques because it contains the energies of the resonances.

4.3 Data Acquisition and Processing

Data was collected using ADAQAcquisition, an acquisition software built on the ROOT framework [10] for spectroscopy. ADAQAcquisition provides a graphical user interface to handle control and readout of digital data acquisition (DAQ) systems [11].

4.4 Analysis Methods

4.4.1 Spectrum Analysis

Gamma yield of the ^{19}F(p,$\alpha\gamma$)^{16}O reaction has been measured previously by integrating spectra from energies of approximately 4.5 to 9 MeV in a study by Dababneh et al. [1]. Integration of entire spectra from 4.5 MeV to 9 MeV was likely done due to the overlap of the characteristic peaks that are close in energy. However, this technique does not account for changes in experimental setup and background. Yield measurements can only be used if the experimental setup is exactly the same as the experimental setup used for the original yield measurements or as a comparison. Resulting spectra were also analyzed using the same analysis technique as described in Dababneh et al. [1] to compare and verify yield results.

The 6.13 MeV gamma production differential cross section values for the ^{19}F(p,$\alpha\gamma$)^{16}O reaction were calculated because this peak is the most easily distinguishable due to the high relative dominance over the entire energy range. To do so, yield of the 6.13 MeV alone would need to be measured. Yield spectra resulting from data taken at varying proton energies were analyzed using the ROOT framework [10]. Spectra were dynamically calibrated in the high energy range using the 6.13 MeV full energy peak, single escape peak, and double escape peak. The peaks that were analyzed are shown in Fig 4-2.

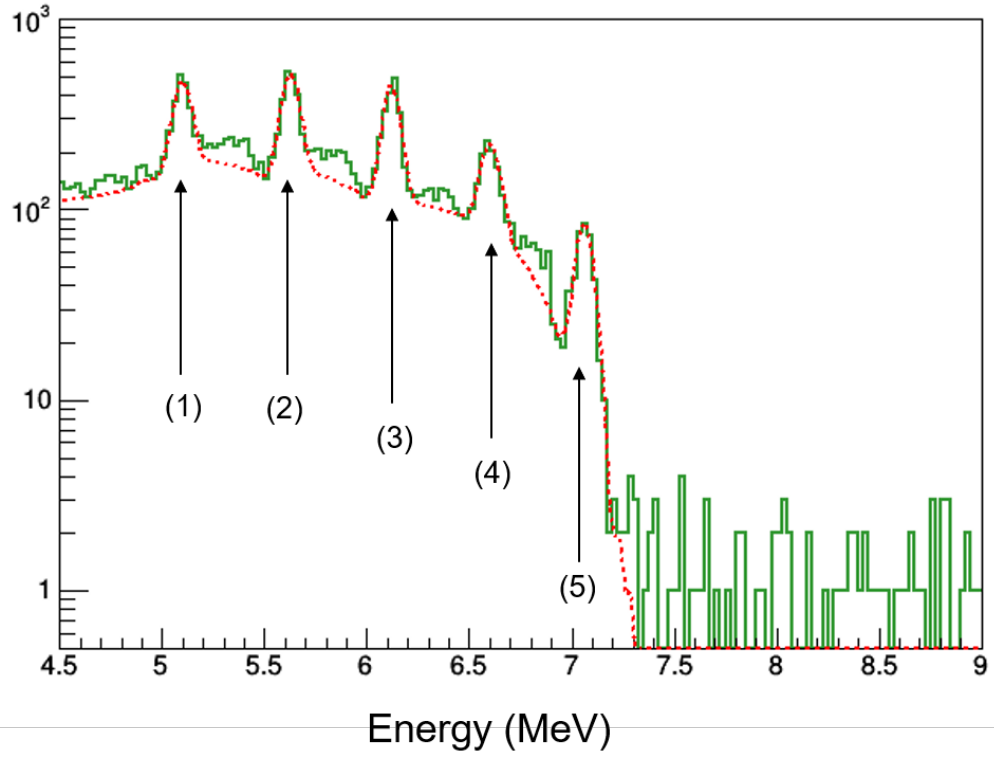

Figure 4-2: This spectrum displays the peaks that were integrated to accurately calculate the cross section data of ^{19}F(p, $\alpha\gamma$)^{16}O. (1) 6.13 MeV double escape peak, (2) 6.13 MeV single escape peak, (3) 6.13 MeV full energy peak, (4) 7.12 MeV single escape peak, (5) 7.12 MeV full energy peak. The 6.92 full energy peak and its escape peaks are not able to be seen in this spectrum due to its low peak dominance at this proton energy.

Each peak shown in Fig. 4-2 was fit with a Gaussian and integrated to obtain the respective yields. The background and continuum were subtracted to reduce variability in yield measurements. The peaks analyzed in addition to the 6.13 MeV full energy peak were needed to obtain the actual yield of the 6.13 MeV full energy peak. The raw yield of the 6.13 MeV characteristic gamma was corrected by accounting for intrinsic efficiency and transmission at the gamma energy and by deconvolving the 7.12 MeV double escape peak. These corrections are described in the following sections.

4.4.2 Intrinsic Efficiency

Intrinsic efficiency as a function of energy was measured using calibrated check sources with characteristic gamma rays of varying energy. The intrinsic efficiency is defined in Eq. 4.1 [7].

$$\epsilon = \frac{number\ of\ pulses\ recorded}{number\ of\ radiation\ quanta\ incident\ on\ detector} \tag{4.1}$$

Each check source was counted for 10 min. Spectrum were analyzed to obtain yield of the check sources in the way described in Section 4.4.1. These sources are given in Table 4.1.

Table 4.1: Calibrated Check Sources and their Energies

Source	Energy (MeV)
^{137}Cs	0.662
^{54}Mn	0.8348
^{60}Co	1.173, 1.332

4.4.3 Transmission

Because gamma attenuation coefficients vary with energy, transmission as a function of energy was first calculated theoretically with attenuation coefficients from Hubbell and Seltzer [12] using the following equation,

$$\tau = \frac{I}{I_0} = e^{-(\frac{\mu}{\rho})\rho t} \tag{4.2}$$

where $\frac{\mu}{\rho}$ is the density adjusted attenuation coefficient, ρ is the density, and t is the thickness of the material between the source and detector. This function was confirmed experimentally by counting four sources for 20 min inside the experimental chamber and then for 20 min outside the experimental chamber. The sources and their energies are given in Table 4.2.

Table 4.2: Calibrated Check Sources and their Energies

Source	Energy (MeV)
^{137}Cs	0.662
^{54}Mn	0.8348
^{60}Co	1.173, 1.332
AmBe	4.4

4.4.4 Single to Double Escape Ratio

The ratio, R, as described in Section 3.2, was confirmed to be constant with energy using an AmBe source with characteristic gamma energy of 4.4 MeV and the 6.13 MeV gamma from ^{19}F(p, $\alpha\gamma$)^{16}O. The AmBe source was used to confirm the ratio because it exhibits significant pair production compared to other check sources. This ratio was also calculated for each spectrum at different proton energies, and the mean was calculated. Using Eq. (3.5), the actual yield was measured for each yield measurement at each energy.

Chapter 5

Results and Discussion

5.1 Intrinsic Efficiency Measurements

Following the procedure in Section 4.4.2, counts emitted from the sources were measured and using the integrated yield, efficiency as a function of energy was calculated. The resulting efficiency measurements and fit are shown in Fig. 5-1

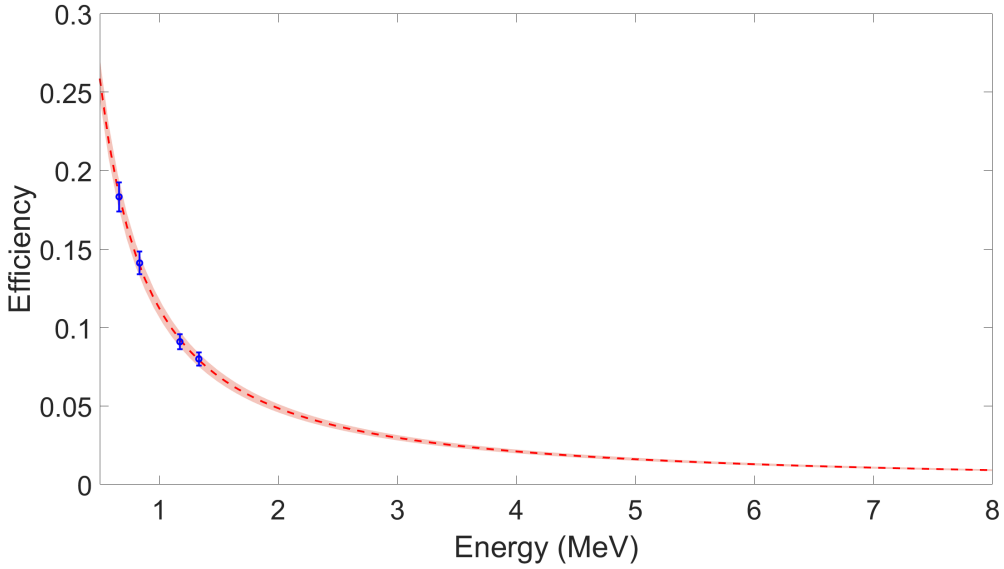

Figure 5-1: Intrinsic efficiency was calculated at 4 energies using check sources described in Section 4.4.2. Data points are plotted with error bars. These data points were fit with a log-log fit shown by the red dashed curve. The error in the fit is shown by the red shading.

The log-log fit for intrinsic efficiency shown in Fig. 5-1 was used to obtain the intrinsic efficiency at incident gamma energies. These efficiency values were used to calculate differential cross section values independent of intrinsic efficiency at differing incident gamma energies.

5.2 Transmission Measurements

Transmission as a function of energy was calculated using Eq. 4.2. The experimental and theoretical resulting transmission measurements and fit are shown in Fig. 5-2

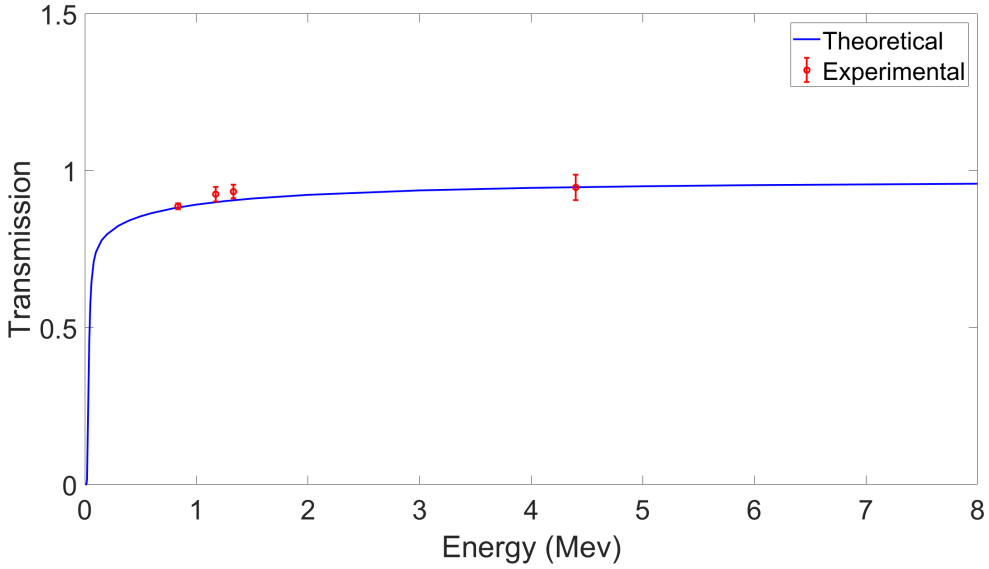

Figure 5-2: This plot shows the experimentally taken transmission data as well as the function found using the theoretical equation of transmission given by 4.2 and attenuation coefficients from Hubbell and Seltzer [12].

The theoretical transmission curve shown in Fig. 5-2 was used to correct yield measurements for varying transmission at different incident gamma energy. The experimentally taken data points confirm the theoretical model. confirming the model. Additionally, in the high energy range, transmission becomes nearly constant.

5.3 Double to Single Escape Ratio

Using Eq. 3.3, the double escape to single escape ratio was calculated using yield data from ^{19}F(p, $\alpha\gamma$)^{16}O spectra taken at varying proton energies. The ratio plotted at different proton energies is shown in Fig. 5-3.

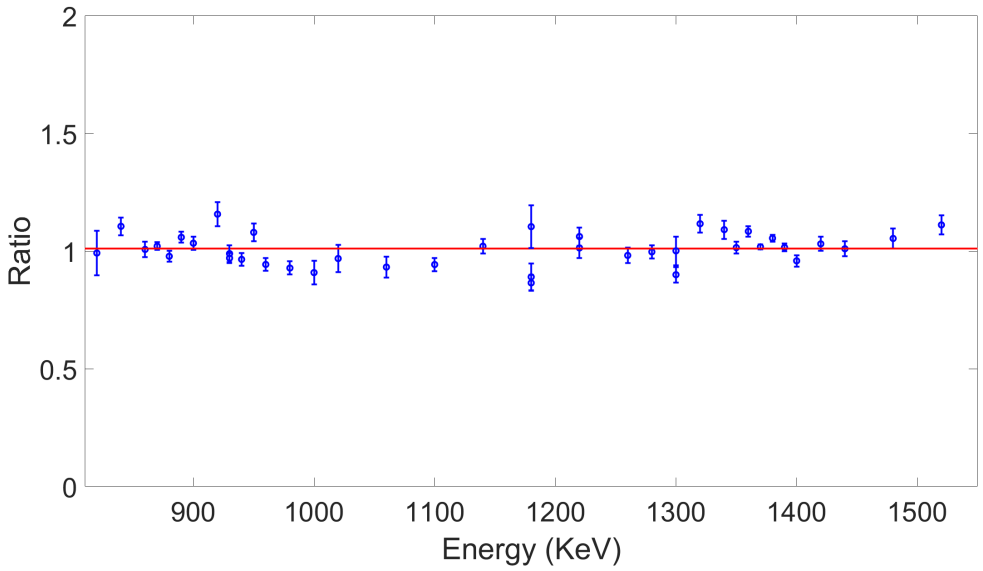

Figure 5-3: The double to single escape peak ratio was calculated for each proton energy spectrum. The mean of the ratio is shown in the red line.

As displayed in 5-3, the mean ratio was found to be R = 1.0104 ± 0.0665. This ratio was used to deconvolve the yield of the 6.13 MeV full energy peak and the yield of the 7.12 MeV double escape peak using Eq. 3.5.

5.4 Yield Measurements

The bulk integral was found by integrating the spectra from energy bounds of 4.5 to 9 MeV shown in Fig. 2-3. This yield measurement was used as a comparison to the gamma yield data from Dababneh et al. [1]. The comparison of the two gamma yield measurements are shown in Fig. 5-4.

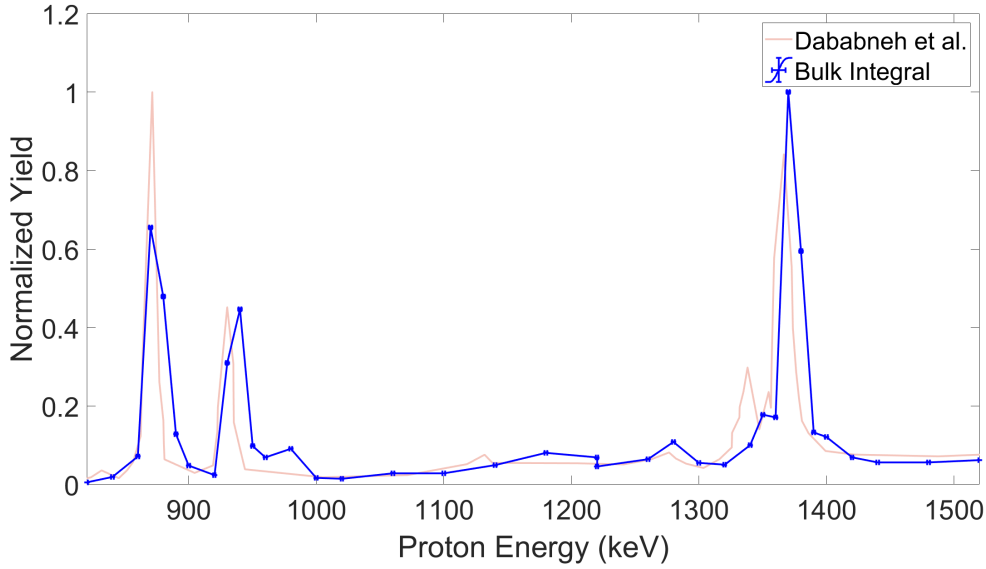

Figure 5-4: Gamma yield resulting from the integration of the entire spectra from 4.5 to 9 MeV was normalized to compare the shape and location of resonances to normalized gamma yield from Dababneh et al. [1]. Data from Dababneh et al. [1] was extracted using WebPlotDigitizer.

As shown in Fig. 5-4, the shape of the bulk integral yield matches the shape of the yield data from Dababneh et al. [1]. However, there is a variable shift in the resonances between data sets. This shift may be due to two potential problems with the experimental equipment. Although an $LaBr_3$ detector has sufficient efficiency, it has relatively low efficiency compared to that of a NaI detector. Because of this, data points taken at energies not on resonance must be taken for longer, almost twice or thrice as long, than data points on resonance. The long data collections lead to the detector electronics warming up resulting in a calibration drift. The calibration drift may affect data collection of several points. Additionally, there may be issues with the terminal potential. The terminal potential may be unsteady, resulting in changes in the gamma yield measurements. Because the resonances of ^{19}F(p, $\alpha\gamma$)^{16}O are extremely narrow, any small changes or movement in terminal potential, resulting in a change in proton energy, can result in a dramatic change in gamma yield while on resonant energy. This is likely the cause in the shift in resonances. The shift of

the locations of the resonances could be corrected by directly comparing data points to data points from Dababneh et al. [1] and shifting data accordingly. However this correction was not taken in this paper because causes of the terminal potential jumps are currently being investigated.

The corrected gamma yield of the 6.13 MeV full energy peak was calculated at each proton energy using the yield of the 7.12 MeV single escape peak, the double escape to single escape ratio found in Section 5.3, and Eq. 3.5

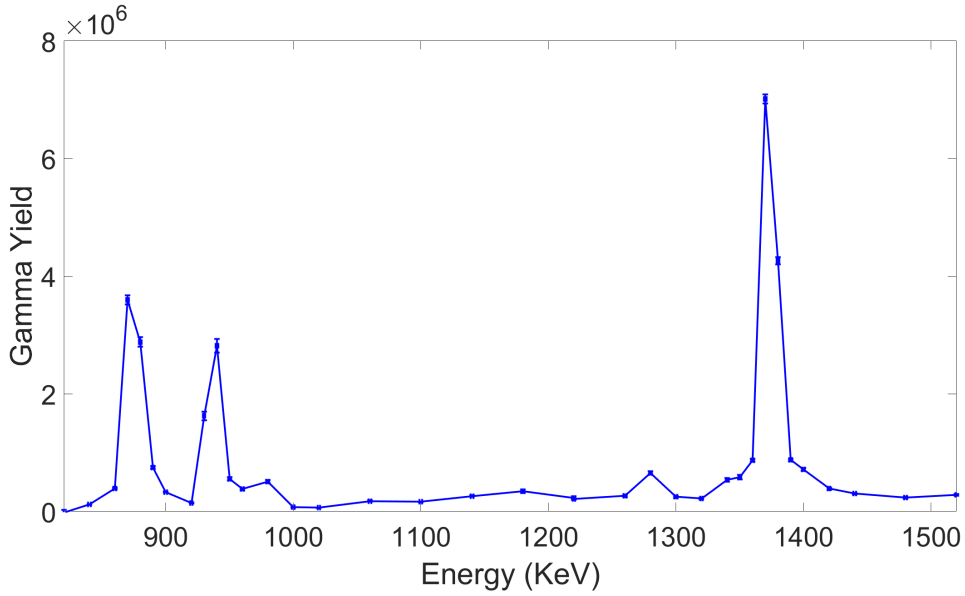

Figure 5-5: The yield of the 6.13 MeV full energy peak was used to calculate the differential cross section data in the following sections.

As shown in Fig. 5-5, the shape of the corrected yield of the 6.13 MeV full energy peak vs. proton energy is consistent with the shape of the bulk integral and yield data from Dababneh et al. [1]. This yield was used to calculate the differential cross section measurements in the following section.

5.5 Differential Cross Section Measurements

Differential cross section was calculated using Eq. 3.2. A plot of the differential cross section values vs. proton energy is shown in Fig. 5-6.

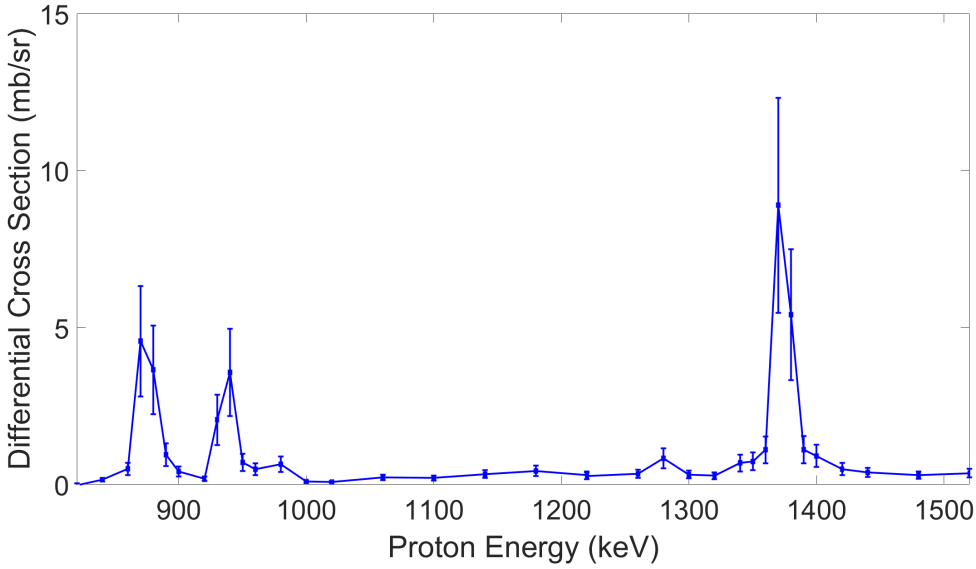

Figure 5-6: Differential Cross Section Values of from Proton Energy of 840 keV to 1520 keV

The values plotted in Fig 5-6 are tabulated in Table 5.1. Possible sources of error in the cross section measurements may result from slight changes in the target and experimental setup between data collection and error in yield due to analysis method and possible calibration shift to the detector electronics warming.

Error resulting from the experimental setup can be reduced if all data is taken at once. This way, changes in experimental setup will likely not occur.

Errors in yield measurements are likely the largest source of error due several contributing factors. Because 6.13 MeV full energy peak is deconvolved with the 7.12 MeV double escape peak to determine the actual yield, error in the yield is affected by the error in the apparent yield of the 6.13 MeV full energy peak, yield of the 7.12 MeV double escape peak, and the ratio of the double escape to single escape peak. All contributing error results from statistical error and error in fit and integration methods. Ensuring the determination of fit and integration methods are robust over all proton energies may reduce error.

Data points may also be affected by calibration shift. This error could reduced by more rigorous analysis methods in which spectra are broken up in uniform time

intervals and analyzed. For example, a data collection lasting 20 minutes could be split into intervals of 10 minutes. The first spectrum consists of data taken in the first 10 minutes, while the second consists of data taken in the second 10 minutes. The subsequent two spectra would be analyzed using the original analysis method.

Table 5.1: Gamma Production Differential Cross Section Values for ^{19}F(p, $\alpha\gamma$)^{16}O was well as their statistical error (1σ)

Proton Energy ± 1 [keV]	**Differential Cross Section [mb/sr]**
840	0.15 ± 0.06
860	0.50 ± 0.19
870	4.56 ± 1.76
880	3.65 ± 1.41
890	0.95 ± 0.37
900	0.42 ± 0.16
920	0.18 ± 0.07
930	2.06 ± 0.80
940	3.57 ± 1.38
950	0.70 ± 0.27
960	0.48 ± 0.19
980	0.64 ± 0.25
1000	0.09 ± 0.04
1020	0.08 ± 0.03
1060	0.22 ± 0.09
1100	0.21 ± 0.08
1140	0.33 ± 0.13
1180	0.43 ± 0.17
1220	0.29 ± 0.12
1220	0.27 ± 0.11
1260	0.34 ± 0.13
1280	0.83 ± 0.32
1300	0.32 ± 0.12
1320	0.28 ± 0.11
1340	0.68 ± 0.26
1350	0.74 ± 0.29
1360	1.10 ± 0.43
1370	8.89 ± 3.43
1380	5.40 ± 2.08
1390	1.11 ± 0.43
1400	0.91 ± 0.35
1420	0.49 ± 0.19
1440	0.38 ± 0.15
1480	0.30 ± 0.12
1520	0.36 ± 0.14

Chapter 6

Conclusion

This work presents differential cross section values for the gamma production reaction ^{19}F(p,$\alpha\gamma$)^{16}O. Although gamma yield measurements were taken in the prior studies, yield measurements alone can only be used to make relative measurements. Differential cross section values can be used in situations of varying geometry, detector, background, and experimental setup. Differential cross section values can be used in a variety of applications. Knowledge of differential cross section values is fundamental in ion beam analysis (IBA) techniques such as nuclear reaction analysis (NRA) and particle induced gamma emission (PIGE) analysis because these values are fundamental to PIGE analysis. In addition to expanding databases of differential cross sections, these differential cross section values will contribute to development of diagnostic techniques that utilize NRA and PIGE. Specifically, a diagnostic technique for plasma facing component conditions that utilizes resonant reactions including ^{19}F(p,$\alpha\gamma$)^{16}O is currently being developed at MIT [4].

While these results are valuable, future work is needed to make these values more accurate. Future work could include study of different gamma production reactions to expand differential cross section databases as well as study of the same reaction at different proton energies. Further work may also include more rigorous analysis of yield data by identifying and accounting for calibration drift over time and the unsteady terminal potential.

Bibliography

[1] S. O F Dababneh, K. Toukan, and I. Khubeis. Excitation function of the nuclear reaction ^{19}F(p, $\alpha\gamma$)^{16}O in the proton energy range 0.3-3.0 MeV. *Nuclear Inst. and Methods in Physics Research, B*, 83(3):319–324, 1993. ISSN 0168583X. doi: 10.1016/0168-583X(93)95849-Z.

[2] J. Charles Barbour, Carl J. Maggiore, and James W. Mayer. *Handbook of Modern Ion Beam Materials Analysis*. Materials Research Society, Pittsburgh, 1st edition, 1995.

[3] Z.S. Hartwig, H.S. Barnard, B.N. Sorbom, R.C. Lanza, B. Lipschultz, P.W. Stahle, and D.G. Whyte. Fuel retention measurements in Alcator C-Mod using Accelerator-based In-situ Materials Surveillance. *Journal of Nuclear Materials*, 463(0022-3115):73–77, 2015.

[4] L.A. Kesler. *Development and testing of an in situ method of ion beam analysis for measuring high-Z erosion inside a tokamak using an AIMS diagnostic*. In progress, MIT, 2018.

[5] L.A. Kesler, B.N. Sorbom, Z.S. Hartwig, H.S. Barnard, G.M. Wright, and D.G. Whyte. Initial results of tests of depth markers as a surface diagnostic for fusion devices. *Nuclear Materials and Energy*, 0(2016):1–5, 2016. ISSN 23521791. doi: 10.1016/j.nme.2016.11.013. URL http://dx.doi.org/10.1016/j.nme.2016.11.013{%}5Cnhttp://linkinghub.elsevier.com/retrieve/pii/S2352179116300345.

[6] M Kokkoris. Nuclear Reaction Analysis (NRA) and Particle-Induced Gamma-Ray Emission (PIGE). *Characterization of Materials*, 2012.

[7] Glenn F. Knoll. *Radiation Detection and Measurement*. John Wiley & Sons, Inc., Hoboken, NJ, 4th edition, 2010.

[8] J. Kopecký, W. Ratyński, and E. Warming. Curves for the response of a Ge(Li) detector to gamma rays in the energy range up to 11 MeV. *Nuclear Instruments and Methods*, 50(2):333–339, 1967. ISSN 0029554X. doi: 10.1016/0029-554X(67)90061-4.

[9] John R. Johnson and Kenneth C. Mann. Single And Double Escape Peaks in Ge(Li) Gamma-Ray Spectra. *Nuclear Instruments and Methods*, 112:601–602, 1973.

[10] Rene Brun and Fons Rademakers. ROOT - An object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1-2):81–86, 1997. ISSN 01689002. doi: 10.1016/S0168-9002(97)00048-X.

[11] Zachary S. Hartwig. The ADAQ framework: An integrated toolkit for data acquisition and analysis with real and simulated radiation detectors. *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 815:42–49, 2016. ISSN 01689002. doi: 10.1016/j.nima.2016.01.017. URL http://dx.doi.org/10.1016/j.nima.2016.01.017.

[12] J. H. Hubbell and S. M. Seltzer. Tables of X-Ray Mass Attenuation Coefficients and Mass Energy-Absorption Coefficients from 1 keV to 20 MeV for Elements Z = 1 to 92 and 48 Additional Substances of Dosimetric Interest. Technical report, 1989. URL https://www.nist.gov/pml/x-ray-mass-attenuation-coefficients.

Appendix A

Data Analysis Scripts

analyzeF.cc outputs spectral analysis

```
1  #include <TFile.h>
2  #include <TTree.h>
3  #include <TH1F.h>
4  #include <time.h>
5  #include <dirent.h>
6  #include <sys/stat.h>
7
8  // Load ADAQ headers
9  #include "ADAQReadoutInformation.hh"
10
11 // Load the ADAQReadout library change path depending on the ...
       machine
12 R__LOAD_LIBRARY(/usr/local/adaq/ADAQ/lib/x86_64/libADAQReadout.so)...
       ;
13
14
15 void analyzeF(string data_directory, string target, string ...
       terminal, string beam, string description, string iteration)
16 {
17   /*
18 This section sets up the code, creating the filenames that need to...
```

```
      be analyzed and the filenames of the output data files.
19
20    */
21
22   //Make filenames
23   //change the directory path to my directory path
24   string ADAQ_Filename, calibration_filename, current_filename;
25   mode_t mode0 = S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH;
26   string prefix ="/home/Monica/Data/experiment/"+data_directory;
27   DIR* dir = opendir(prefix.c_str());
28   if (dir)
29     {
30     closedir(dir);
31     }
32   else
33     {
34     /* opendir() failed for some other reason. */
35     cout << "\nError! Finle name not found\n"
36      << endl;
37     return;
38   }
39
40   // Print location of experimental data
41   //creates yield and current files, prints out that it is making ...
       file names
42   stringstream SSadaq, SScalibration, SScurrent;
43   SSadaq << prefix << "/" << target << "_yield_" << terminal <<"...
       kV_"<<beam<<"_"<<description<<"_"<<iteration<<".adaq.root";
44   ADAQ_Filename=SSadaq.str();
45   cout << "Opening " << ADAQ_Filename<< endl;
46   SScalibration << prefix << "/" << target << "_calibration_" << ...
       terminal <<"kV_"<<iteration<<".acal";
47   calibration_filename=SScalibration.str();
48   cout << "Calibrating with " << calibration_filename << endl;
49   SScurrent << prefix << "/" << target << "_current_" << terminal ...
       <<"kV_"<<description<<"_"<<iteration<<".txt";
```

```
50  current_filename=SScurrent.str();
51  cout << "Opening " << current_filename << endl;
52
53
54  //finds system time for the analysis, records in output file
55  time_t Time = chrono::system_clock::to_time_t(chrono::...
        system_clock::now());
56  char date [12] ;
57  struct tm * timeinfo = localtime(&Time);
58  strftime(date, 12, "%Y%m%d", timeinfo);
59
60
61
62
63  //Make output filenames
64  //checks for the directory in the analysis folder
65  stringstream SSoutput, SSplot0, SSplot1, SSplot2, SSplot3;
66  string output_filename, plot0_filename, plot1_filename, ...
        plot2_filename, plot3_filename;
67  string prefix1 = "/home/Monica/Data/analysis/";
68  prefix1.append(date);
69  string prefix0 = prefix1;
70    DIR* dir1 = opendir(prefix1.c_str());
71  if (dir1)
72    {
73    closedir(dir1);
74    cout << "data file exists\n";
75    }
76  else if (ENOENT == errno)
77    {
78      mkdir(prefix1.c_str(), mode0);
79      prefix1.append("/plots");
80      mkdir(prefix1.c_str(), mode0);
81      cout << "data file created\n";
82    }
83  else
```

```
84      {
85
86
87      cout << "\nError! File name not found \n"
88       << endl;
89      return;
90    }
91    // Print out analysis data location information for user
92    cout << "Location of data and plots will be " << prefix1 <<"\n";
93    SSoutput << prefix0 << "/" << target << "_rootanalysis_" << ...
          terminal <<"kV_"<< description << "_" << iteration;
94    output_filename=SSoutput.str();
95
96    SSplot0 << prefix0 << "/plots/" << target << "_plot_time_" << ...
          terminal <<"kV_"<< description << "_" << iteration << ".png";
97    plot0_filename=SSplot0.str();
98    SSplot1 << prefix0 << "/plots/" << target << "_plot_ratio_" << ...
          terminal <<"kV_"<< description << "_" << iteration << ".png";
99    plot1_filename=SSplot1.str();
100   SSplot2 << prefix0 << "/plots/" << target << "_plot_spectrum_" ...
          << terminal <<"kV_"<< description << "_" << iteration << "...
          .png";
101   plot2_filename=SSplot2.str();
102   SSplot3 << prefix0 << "/plots/" << target << "_plot_fit_" << ...
          terminal <<"kV_"<< description << "_" << iteration << ".png";
103   plot3_filename=SSplot3.str();
104
105   cout << "Output data files are:\n";
106   cout << output_filename << "\n";
107   cout << plot0_filename << "\n";
108   cout << plot1_filename << "\n";
109   cout << plot2_filename << "\n";
110   cout << plot3_filename << "\n";
111
112
113
```

```
114   /*
115
116 TIMELENGTH AND CALIBRATION HISTOGRAMS
117 First for-loop that reads the wavetree for the ADAQ file and then ...
       determines how long it is in time
118
119    */
120
121
122
123 //Read in ADAQ file and declare relevant variables
124
125   Double_t TimeStamp2Nanoseconds = 4e-9;
126   ULong64_t TimeStamp0;
127   Double_t PulseArea0, PulseEnergy0, PulseMax = 0;
128   Double_t Time0, Time1=0;
129   Double_t currentmax=0;
130   TFile *F0 = new TFile(ADAQ_Filename.c_str());
131
132
133 // Get the TTree containing data and assign the waveform data ...
       class
134   TTree *WT0 = (TTree *)F0->Get("WaveformTree");
135   Int_t NumEntries0 = WT0->GetEntries();
136   ADAQWaveformData *WD0 = new ADAQWaveformData;
137   WT0->SetBranchAddress("WaveformDataCh0", &WD0);
138
139
140   Int_t bins=400, binmin=0, adcmax=56000, binmax=10;
141
142
143   Int_t fitbins0=30, fitbins1=150, pFmin=19000, pFmax = 40000;
144
145   TH1F *Huncalibrated= new TH1F("uncalibrated", "uncalibrated data...
        ",bins,binmin, adcmax);
146   TH1F *Hcalibrated= new TH1F("calibrated", "Integrated spectrum",...
```

```
         bins, binmin, binmax);
147
148
149    TH1F *H0= new TH1F("peak0", "DE, SE, pF-19, E=5.105, 5.619, 6.13...
         MeV", fitbins1, pFmin, pFmax);
150
151   //Find the end time of the histogram
152    for(Int_t entry0=0; entry0<NumEntries0; entry0++)
153    { //Get entry from wavetree
154      WT0->GetEntry(entry0);
155      // Get the timestamp (64-bit integer, 8 ns resolution)
156      TimeStamp0 = WD0->GetTimeStamp();
157      // Convert to real time [s]
158
159      //converts time stamp to real time
160      Time0 = TimeStamp0 * TimeStamp2Nanoseconds;
161      //Get the Pulse Area
162      PulseArea0 = WD0->GetPulseArea();
163      Huncalibrated->Fill(PulseArea0);
164      H0->Fill(PulseArea0);
165
166
167      if (Time0>Time1){
168        Time1=Time0;
169        }
170    }
171
172
173    /*
174
175  CALIBRATION
176  Finds the known peaks in the spectrum, from full energy, single ...
       escape, and double escape peak of 6.13 MeV Gamma
177
178     */
179
```

```
180
181    //create vector to make a fit for two different vectors E = ...
           Energy, V = adc value
182    vector<Double_t> E, V;
183    //TSpectrum to use search function. Search for 1 peak??
184    TSpectrum *pk0=new TSpectrum(3);
185
186    //Search(histogram pointer, sigma (find in adaq), graph on or ...
           off, threshold)
187    Int_t nfound0 = pk0->Search(H0, 2,"gon",0.4);
188    cout << "Found " << nfound0 << " peaks.\n";
189    Double_t *peak0 = pk0->GetPositionX();
190    //V.push_back takes values of adc and adds end of array
191
192    V.push_back(peak0[0]);
193    V.push_back(peak0[1]);
194    V.push_back(peak0[2]);
195    std::sort(V.begin(), V.end());
196    //add energy at the end of the array E
197    E.push_back(5.105);
198    E.push_back(5.619);
199    E.push_back(6.13);
200
201
202    //make fit
203      //make TGraph to use the TF1 fit function
204    TGraph *G1 = new TGraph(E.size(), &V[0], &E[0]);
205
206  //root, fit, 1 dimension, pol1 = 1st order fit
207    TF1 *f1 = new TF1("f1","pol1");
208
209    Double_t slope0, const0;
210    G1->Fit("f1", "Q");
211    const0 = f1->GetParameter(0);
212    slope0 = f1->GetParameter(1);
213
```

```
214
215
216
217   /*
218
219 CREATE CALIBRATED HISTOGRAM
220
221    */
222
223
224
225
226 //Create the bin settings based on the t-length of the data.
227 //Time1 length of time data was collected earlier
228   Int_t bin_time = static_cast<int>(Time1);
229
230   //Define histograms.
231   //cout << "bin time is " << bin_time <<"\n";
232   TH1F *Htime= new TH1F("time", "Current/Gamma comparison", ...
          bin_time, 0, bin_time);
233
234   Double_t bintest0, bintest1, bintest2;
235   //upper and lower bound of energy
236   Double_t ubound=9;
237   Double_t lbound=4.5;
238
239   //Finally, fill histograms.
240   for(Int_t entry0=0; entry0<NumEntries0; entry0++)
241   {
242     WT0->GetEntry(entry0);
243     // Get the timestamp (64-bit integer, 8 ns resolution)
244     TimeStamp0 = WD0->GetTimeStamp();
245
246     // Convert to real time [s]
247     Time0 = TimeStamp0 * TimeStamp2Nanoseconds;
248
```

```
249      //Populate Time Histograms
250      //Get the Pulse Area
251
252      PulseArea0 = WD0->GetPulseArea();
253      PulseEnergy0 = const0+slope0*PulseArea0;
254
255      Hcalibrated->Fill(PulseEnergy0);
256
257      if(PulseEnergy0>lbound && PulseEnergy0<ubound){
258        Htime->Fill(Time0);
259      }
260    }
261
262
263
264    /*
265
266  ALIGNING GAMMA/CURRENT DATA
267
268     */
269
270
271
272    //Find start of gamma data
273    Double_t value;
274    Int_t t0,t1, entry1;
275    for(entry1=0; entry1<bin_time; entry1++){
276      bintest0 = Htime->GetBinContent(entry1);
277      //cout << "bintest0 is " << bintest0 << "  bintest2 is " << ...
             bintest2 << endl;
278      value = abs(bintest0-bintest2);
279      //cout << "value is " << value << endl;
280      if(value>20){
281             t0=entry1-1;
282             break;
283      }
```

```
284       bintest2=bintest0;
285     }
286
287     //Find maximum bin content.
288     for(entry1=0; entry1<bin_time; entry1++){
289       bintest1 = Htime->GetBinContent(entry1);
290       if (bintest1 > PulseMax){
291         PulseMax = bintest1;
292       }
293     }
294
295
296
297     /*
298
299   CURRENT DATA
300
301      */
302
303
304
305     vector<Double_t> T, I, Iplot, Y, R, Tr,Ex, Ey;
306     Double_t sumcurrent=0;
307       ifstream Input1(current_filename);
308     if(!Input1.is_open()){
309       cout << "\nError! File name not found\n"
310        << endl;
311       return;
312     }
313     else{
314       Double_t t1=1000000;
315       float time,i, b, current;
316       int a;
317       string line;
318       char c;
319       while(!Input1.eof()){
```

```
320       getline (Input1,line);
321
322       sscanf(line.c_str(), "%d/%d/%d %f:%f:%f %cM %f %f ",&a,&a,&a...
              , &b, &b, &b,&c, &time, &current);
323
324       if (current > currentmax){
325     currentmax = current;
326       }
327
328       if(Input1.eof())
329     break;
330       T.push_back(time);
331       I.push_back(current);
332       sumcurrent=sumcurrent+current;
333       //cout << "current " << current << "   time " << time << ...
              endl;
334       //cout << sumcurrent << endl;
335     }
336     Iplot=I;
337    }
338
339  Input1.close();
340  Int_t sizecurrent=I.size();
341  Double_t ratiomax =2*PulseMax/currentmax;
342  Double_t test;
343  Double_t plotmax =ratiomax*currentmax*1.1; //this is silly, but ...
       it works...
344
345
346
347
348   /*
349
350 ADJUST CURRENT DATA
351
352    */
```

```
353
354
355
356 if (T.at(1)<1)
357   {Double_t factor=10;
358     for(Int_t b=1; b<sizecurrent; b++){
359       T[b]=T[b]*factor;
360       // cout << "time " << T[b] << endl;
361     }}
362
363 Double_t i_1=I.at(0), i_2=I.at(1);
364 Int_t a=1;
365  for(a; a<sizecurrent; a++){
366    i_2=I.at(a);
367    test=abs(i_2/i_1);
368    //cout << "test is " << test << endl;
369   if(test>50){
370     t1=T.at(a);
371      break;
372      }
373   //cout << "i_1 is " << i_1 << " and i_2 is " << i_2 <<endl;
374      i_1=i_2;
375   }
376
377 Int_t time_diff=t0-t1;
378 //cout << "t0 is " << t0 << " and t1 is " << t1 << ", diff is "<<...
       time_diff << endl;
379
380 for(a=0; a<sizecurrent; a++){
381   T[a]=T[a]+time_diff;
382   if (T[a]<1E-2 && T[a]>-1E-2){
383     T[a]=0;
384   }
385   Iplot[a]=ratiomax*I[a];
386 }
387
```

```
388
389
390
391
392   Int_t sec_num = 120;
393   Int_t sections = bin_time/sec_num;
394   Int_t d, limit,f;
395   Double_t sum_y=0, sum_i=0, yield, error1;
396   for(Int_t c=0;c<sections+1;c++){
397     d=c*sec_num;
398     sum_y=0;
399     sum_i=0;
400     limit = (c+1)*sec_num;
401     if(limit>bin_time){
402       limit=bin_time;}
403     for(d;d<limit;d++){
404       yield = Htime->GetBinContent(d);
405
406       Y.push_back(yield);
407       sum_y = sum_y+Y[d];
408       sum_i = sum_i+I[d];
409       }
410
411     if(sum_i>0){
412       R.push_back(sum_y/sum_i);
413       error1=sqrt(sum_y)/sum_i;
414       Ey.push_back(error1);
415       Ex.push_back(0);
416       Tr.push_back(limit);
417       }
418   }
419
420    Double_t integ = Hcalibrated -> Integral(Hcalibrated->FindBin(...
          lbound), Hcalibrated->FindBin(ubound));
421    Double_t error = sqrt(integ);
422
```

```
423
424   //background subtraction
425   Int_t Iterations = 7;
426   TH1F *Clone_Hcalibrated = (TH1F *)Hcalibrated->Clone("...
          Clone_Hcalibrated");
427   TSpectrum *bkg=new TSpectrum();
428   TH1F *Background_Hcalibrated = (TH1F *)bkg->Background(...
          Clone_Hcalibrated, Iterations,"nosmoothing ...
          BackDecreasingWindow BackOrder2 BackSmoothing3 Q");
429
430   Clone_Hcalibrated->Add(Background_Hcalibrated,-1);
431
432
433
434   double lower [5] = {4.9, 5.48, 5.95, 6.44, 6.92};
435   double upper [5] = {5.3, 5.8, 6.3, 6.8, 7.3};
436   string peakName [] = {"_61DE.dat", "_61SE.dat", "_61FE.dat", "...
          _71SE.dat", "_71FE.dat"};
437
438
439   Double_t intPeak, Const, ConstErr, Mean, MeanErr, Sigma, ...
          SigmaErr, Res, ResErr, CovConstSigma, Err;
440
441    // Compute the spectrum bin width
442   Double_t Range = binmax - binmin;
443   Double_t BinWidth = Range / bins;
444
445   //integration and writing output file for each peak of interest
446   for(int i = 0; i < 5; i = i+1) {
447
448     TF1 *fit = new TF1("fit","gaus", lower[i],upper[i]);
449     TH1F *fit_hist = new TH1F("fit histogram", "fit", bins, binmin...
            , binmax);
450     TFitResultPtr fit_result = Clone_Hcalibrated->Fit("fit", "SRNQ...
            ");
451     fit->SetRange(binmin, binmax);
```

```
452     fit_hist->Eval(fit);
453
454     intPeak = fit_hist -> Integral(fit_hist->FindBin(binmin), ...
            fit_hist->FindBin(bins));
455   //add all the peaks to background_Hcalibrated
456     Background_Hcalibrated->Add(fit_hist,1);
457
458   // Get the fit correlation and covariance matrix
459     TMatrixDSym CorMatrix = fit_result->GetCorrelationMatrix();
460     TMatrixDSym CovMatrix = fit_result->GetCovarianceMatrix();
461
462   // Get the gaussian fit parameters
463
464     Const = fit_result->Parameter(0);
465     ConstErr = fit_result->ParError(0);
466
467     Mean = fit_result->Parameter(1);
468     MeanErr = fit_result->ParError(1);
469
470     Sigma = fit_result->Parameter(2);
471     SigmaErr = fit_result->ParError(2);
472
473     Res = 2.35 * Sigma / Mean * 100;
474     ResErr = Res * sqrt(pow(SigmaErr/Sigma,2) + pow(MeanErr/Mean...
            ,2));
475
476   // Compute the covariance of constant / sigma
477
478     CovConstSigma = CovMatrix(2,0);
479
480
481     Err = sqrt(TMath::TwoPi()) * sqrt(pow(Sigma*ConstErr,2)+pow(...
            Const*SigmaErr,2)+2*Sigma*Const*CovConstSigma) / BinWidth;
482
483   ofstream Out(output_filename.c_str()+peakName[i], ofstream::trunc...
        );
```

```
484
485   Out << setprecision(4);
486
487   Out << "# Directory : " << data_directory << "\n"
488       << "# Terminal potential [kV] : " << terminal << "\n"
489       << "# Target : " << target << "\n"
490       << "# Beam : " << beam << "\n"
491       << "# Iteration : " << iteration << "\n"
492       << "# Description of data : " << description << "\n"
493       << "# Date of analysis : " << date << "\n"
494       << "\n"
495       << "# Analysis results\n"
496       << "         Bins : " << bins << "\n"
497       << " Big Integral : " << integ << "\t\t"<< error << "\n"
498       << "       Bounds : " << lbound << "\t\t"<< ubound << "\n"
499       << "     Integral : " << intPeak << "\t\t" << Err << "\n"
500       << "       Bounds : " << lower[i] << "\t\t"<< upper[i] << "\...
          n"
501       << "Total current : " << sumcurrent << "\n"
502       << "\n"
503       << "# Calibration fit parameters\n"
504       << "        Slope : " << slope0 << "\n"
505       << "    Intercept : " << const0 << "\n"
506       << "\n"
507       << "# Gaussian fit parameters\n"
508       << "     Constant : " << Const << setw(15) << ConstErr << "\...
          n"
509       << "         Mean : " << Mean << setw(15) << MeanErr << "\n"
510       << "        Sigma : " << Sigma << setw(15) << SigmaErr << "\...
          n"
511       << "   Resolution : " << Res << setw(15) << ResErr << "\n"
512       << "CovConstSigma : " << CovConstSigma << "\n"
513       << "\n"
514       << "# Gaussian fit correlation matrix\n";
515
516   for(Int_t i=0; i<3; i++){
```

```
517       for(Int_t j=0; j<3; j++){
518         Out << setw(15) << CorMatrix(i,j);
519       }
520       Out << endl;
521     }
522
523     Out << "\n"
524         << "# Gaussian fit covariance matrix\n";
525
526     for(Int_t i=0; i<3; i++){
527       for(Int_t j=0; j<3; j++){
528         Out << setw(15) << CovMatrix(i,j);
529       }
530       Out << endl;
531     }
532
533
534     Out << "\n"
535         << "Time [s]| Ratio [\u0263/A]\t| Error\n";
536     for(Int_t z=0; z<sections+1; z++){
537       Out << Tr[z]<< "\t|\t" << R[z] << "\t|\t"<<Ey[z]<<"\n";
538           }
539     Out << "\n"
540         << "Time [s]|Yield [\u0263]\t|Time [s]\t|Current [A/s] \n";
541     for(Int_t y=0; y<Y.size(); y++){
542     Out << y << "\t|\t" << Y[y] << "\t|\t" << T[y] << "\t|\t" << I[y...
          ]
543     << endl;
544
545           }
546     for(Int_t x=Y.size(); x<I.size(); x++){
547     Out << "\t|\t\t|\t" << T[x] << "\t|\t" << I[x]
548     <<endl;
549           }
550
551
```

```
552     delete fit;
553     delete fit_hist;
554     CorMatrix.Clear();
555     CovMatrix.Clear();
556   }
557
558
559 //Make calibration file
560   ofstream Out0(calibration_filename.c_str(), ofstream::trunc);
561
562   Out0 << setprecision(8);
563
564   Out0 <<"\t"<< E[0]<<"\t" << V[0] << "\n"
565        <<"\t"<< E[1]<<"\t" << V[1] << "\n"
566        <<"\t"<< E[2]<<"\t" << V[2] << "\n"
567        <<endl;
568
569
570
571
572   //Plotting
573   int lakpurple=kViolet-6;
574   int lakred=2;
575   int lakgreen=kGreen-2;
576   int line=2;
577
578
579
580   // yield and curent vs. time
581   TCanvas *cl1 = new TCanvas("cl1","Time Histogram");
582   Htime->SetStats(false);
583   Htime->Draw("");
584   Htime->SetLineColor(lakgreen);
585   Htime->SetLineWidth(line);
586   Htime->SetAxisRange(-50, bin_time+50, "X");
587   Htime->SetAxisRange(0,plotmax ,"Y");
```

```
588    TGraph *G = new TGraph(T.size(), &T[0], &Iplot[0]);
589    G->Draw("same");
590    G->SetTitle("Hi Zach!");
591    G->SetMarkerColor(kRed);
592    G->SetMarkerSize(2.);
593    G->SetMarkerStyle(20);
594    TLegend *leg = new TLegend(0.7,0.1,0.89,0.3);
595      leg->SetHeader("Legend");
596          leg->AddEntry(Htime,"gamma yield");
597      leg->AddEntry("G","beam current");
598      leg->Draw();
599    TImage *img1 = TImage::Create();
600     img1->FromPad(cl1);
601     img1->WriteImage(plot0_filename.c_str());
602
603     //yield vs. current over time
604    TCanvas *cl2 = new TCanvas("cl2","Time Histogram");
605    TGraph *G0 = new TGraphErrors(R.size(), &Tr[0], &R[0], &Ex[0], &...
          Ey[0]);
606    TF1 *f2 = new TF1("f2", "pol1");
607    Double_t slope1, const1;
608    G0->Fit("f2", "Q");
609    const1 = f2->GetParameter(0);
610    slope1 = f2->GetParameter(1);
611    G0->Draw("ALP");
612    G0->SetLineWidth(0);
613    G0->SetTitle("Hi Zach!");
614    G0->SetMarkerColor(kRed);
615    G0->SetMarkerSize(2.);
616    G0->SetMarkerStyle(20);
617    TImage *img2 = TImage::Create();
618     img2->FromPad(cl2);
619     img2->WriteImage(plot1_filename.c_str());
620
621
622     //unaltered data spectrum
```

```
623    TCanvas *cl3 = new TCanvas("cl3", "Data Spectrum");
624    Hcalibrated->SetStats(false);
625    Hcalibrated->Draw("");
626    Hcalibrated->SetLineColor(lakgreen);
627    Hcalibrated->SetLineWidth(line);
628    cl3->SetLogy();
629    TImage *img3 = TImage::Create();
630     img3->FromPad(cl3);
631     img3->WriteImage(plot2_filename.c_str());
632
633    //data spectrum with fit
634    TCanvas *cl4 = new TCanvas("cl4", "Data Spectrum");
635    Hcalibrated->SetStats(false);
636    Hcalibrated->Draw("");
637    Hcalibrated->GetXaxis()->SetRangeUser(4.5, 9);
638    Hcalibrated->SetLineColor(lakgreen);
639    Hcalibrated->SetLineWidth(line);
640
641    Background_Hcalibrated->Draw("sameC");
642    Background_Hcalibrated->SetLineColor(lakred);
643    Background_Hcalibrated->SetLineWidth(line);
644    Background_Hcalibrated->SetLineStyle(2);
645    cl4->SetLogy();
646    TImage *img4 = TImage::Create();
647     img4->FromPad(cl4);
648     img4->WriteImage(plot3_filename.c_str());
649
650
651  }
```

allfiles.cc enables analyzing all files in a directory using analyzeF.cc

```
1  #include <TFile.h>
2  #include <TTree.h>
3  #include <TH1F.h>
4  #include <time.h>
5  #include <dirent.h>
6  #include <sys/stat.h>
7  #include "analyzeF.cc"
8
9  // Load ADAQ headers
10 #include "ADAQReadoutInformation.hh"
11
12 // Load the ADAQReadout library
13 R__LOAD_LIBRARY(/usr/local/ADAQ/lib/x86_64/libADAQReadout.so);
14
15 void allfiles(string name, const char* beg, string beam, string ...
       description)
16 {  string iteration0=description+"_0.adaq.root";
17    const char *ext0 = iteration0.c_str();
18    string iteration1=description+"_1.adaq.root";
19    const char *ext1 = iteration1.c_str();
20    string prefix="/home/Monica/Data/experiment/";
21    prefix.append(name);
22    const char *dirname = prefix.c_str();
23    cout << dirname << endl;
24    TSystemDirectory dir(dirname, dirname);
25    TList *files = dir.GetListOfFiles();
26    vector<string> V, V1;
27    if (files) {
28       TSystemFile *file;
29       TString fname;
30       TIter next(files);
31       while ((file=(TSystemFile*)next())) {
32          fname = file->GetName();
```

```
33          if (!file->IsDirectory() && fname.EndsWith(ext0) && ...
                fname.BeginsWith(beg)) {
34              cout << fname.Data() << endl;
35          Int_t kV = fname.Index("kV");
36          string voltage_str = fname.operator()(kV-3,3);
37              //Double_t voltage = voltage_str.Atof();
38          if (voltage_str!=0)
39            V.push_back(voltage_str);
40           }
41
42       if (!file->IsDirectory() && fname.EndsWith(ext1) && ...
            fname.BeginsWith(beg)) {
43              cout << fname.Data() << endl;
44          Int_t kV1 = fname.Index("kV");
45          string voltage1_str = fname.operator()(kV1-3,3);
46          if (voltage1_str!=0)
47            V1.push_back(voltage1_str);
48            }
49        }
50     }
51     for (int a=0; a<V.size(); a++){
52       analyzeF(name,beg,V[a],beam,description,"0");}
53
54     for (int b=0; b<V1.size(); b++){
55       analyzeF(name,beg,V1[b],beam,description,"1");}
56  }
```

readRoot.m reads output files from allFiles.cc or analyzeF.cc to extract date

```
function [big, big_error, gaus, gaus_error, charge]=readRoot(...
    directory,target, terminal, description, iteration, peak)

%File format: LiF_rootanalysis_600kV_data_0.dat where
% target = "LiF"
% terminal = 600
% description = "data"
% iteration = 0
% peak = 61FE, 61SE, 61DE, 71FE, 71SE

%directory is in C:/Users/monic/Documents/Thesis/analysis/ and ...
    describes the
%data/date it was analyzed.


%gamma in MeV, terminal in kV
filepath='C:/Users/monic/Documents/Thesis/analysis/';
name=num2str(terminal);

filename=strcat(filepath,directory,'/',target,'_rootanalysis_',...
    name,'kV_',description,'_',iteration,'_', peak,'.dat');
%%%%%%%%%%%%%%%%%%
N=6;
header=9;
string='%s %f %f';
fileID=fopen(filename);
M = textscan(fileID, string, N, 'HeaderLines', header, 'Delimiter'...
    , {'/t',':'});
%celldisp(M)
big=M{2}(2);
big_error=M{3}(2);
gaus=M{2}(4);
gaus_error=M{3}(4);
```

```
30 charge=M{2}(6);
31 fclose(fileID);
```

getIntegral.m calls readRoot.m to extract yield data, error from yield, and proton from multiple ROOT output files in one directory

```
function [E, gaus, gaus_error] = getIntegral(directory,peak, ...
    terminal0, terminal1)

    big0 =   zeros(length(terminal0), 1);
    big_error0 =   zeros(length(terminal0), 1);
    gaus0 =   zeros(length(terminal0), 1);
    gaus_error0 =   zeros(length(terminal0), 1);
    charge0 = zeros(length(terminal0), 1);
    big1 = zeros(length(terminal1), 1);
    big_error1 = zeros(length(terminal1), 1);
    gaus1 = zeros(length(terminal1), 1);
    gaus_error1 = zeros(length(terminal1), 1);
    charge1 = zeros(length(terminal1), 1);

    for j = 1:length(terminal0)
        [big0(j), big_error0(j), gaus0(j), gaus_error0(j), charge0...
            (j)] = readRoot(directory,'LiF', terminal0(j), 'LaBr', ...
            '0', peak);
    end



    for i = 1:length(terminal1)
        [big1(i), big_error1(i), gaus1(i), gaus_error1(i), charge1...
            (i)] = readRoot(directory,'LiF', terminal1(i), 'LaBr', ...
            '1', peak);
    end
    energy0 = terminal0.*2 + 20;
    energy1 = terminal1.*2 + 20;

    g = [gaus0; gaus1];
    gerr = [gaus_error0;gaus_error1];
```

```
28      [E, order] = sort([energy0; energy1]);
29      gaus = g(order,:);
30      gaus_error = gerr(order,:);
31  end
```

getOtherParameters.m calls readRoot.m to extract the bulk integral, charge, and energy from multiple ROOT output files in one directory

```
function [E, big, big_error, charge] = getOtherParameters(...
    directory, terminal0, terminal1)
%     terminal0 = xlsread('C:/Users/monic/Documents/Thesis/...
    analysis/terminal.xlsx', 'C1:C32');
%     terminal1 = xlsread('C:/Users/monic/Documents/Thesis/...
    analysis/terminal.xlsx', 'D1:D9');
     peak = '61FE';
     big0 =   zeros(length(terminal0), 1);
     big_error0 =   zeros(length(terminal0), 1);
     gaus0 =   zeros(length(terminal0), 1);
     gaus_error0 =   zeros(length(terminal0), 1);
     charge0 = zeros(length(terminal0), 1);
     big1 = zeros(length(terminal1), 1);
     big_error1 = zeros(length(terminal1), 1);
     gaus1 = zeros(length(terminal1), 1);
     gaus_error1 = zeros(length(terminal1), 1);
     charge1 = zeros(length(terminal1), 1);

     for j = 1:length(terminal0)
         [big0(j), big_error0(j), gaus0(j), gaus_error0(j), charge0...
             (j)] = readRoot(directory,'LiF', terminal0(j), 'LaBr', ...
             '0', peak);
     end

     for i = 1:length(terminal1)
         [big1(i), big_error1(i), gaus1(i), gaus_error1(i), charge1...
             (i)] = readRoot(directory,'LiF', terminal1(i), 'LaBr', ...
             '1', peak);
     end

     energy0 = terminal0.*2 + 20;
     energy1 = terminal1.*2 + 20;
```

```
26      b = [big0; big1];
27      be = [big_error0; big_error1];
28      g = [gaus0; gaus1];
29      gerr = [gaus_error0;gaus_error1];
30      c = [charge0;charge1];
31
32      [E, order] = sort([energy0; energy1]);
33      big = b(order,:);
34      big_error = be(order,:);
35      gaus = g(order,:);
36      gaus_error = gerr(order,:);
37      charge = c(order,:);
38  end
```

transmission.m calculates transmission using experimental data and the theoretical model

```
g = csvread('C:/Users/monic/Documents/Thesis/...
    GammaAttenuationGlass.csv');

energy = g(:,1);
muRho_g = g(:,2);
x_g = 0.8128; %cm
rho = 2.23 %g/cm^3


% I = I_0* exp(murho*x)
tau = exp(-muRho_g.*rho*x_g);
error=0.01*tau;



t_energy = [0.8348 1.173 1.332 4.4];
int_in = [19707.4 5724.52 4260.14 1625.06];
err_in = [161.398 103.979 71.5107 51.2946];
int_out = [22235.9 6190.4 4567.19 1717.54];
err_out = [167.041 103.709 73.9476 49.9521];
[t, t_error] = getTransmission(int_in, int_out, err_in, err_out);

stuff = figure('units', 'normalized', 'outerposition', [0 0 1 1]);
plot(energy, tau, 'b', 'LineWidth', 2)
xlim([0 20])
ylim([0 1.5])
hold on
errorbar(t_energy, t, t_error, 'ro', 'LineWidth', 2)
hold on
peaks = [5.11, 5.52, 6.13, 6.10, 7.12];
transmission_val = interp1(energy, tau, peaks);
err_tv = transmission_val.*0.01;
```

```
% scatter(peaks, transmission_val, 's', 'filled', 'LineWidth', 2)
xlim([0 8])
xlabel('Energy (Mev)');
ylabel('Transmission')
legend('Theoretical', 'Experimental')
set(gca, 'FontSize', 30)
```

LaBrEfficiency.m calculates efficiency and error from experimental data

```
function [efficiency, error]=LaBrEfficiency(integral, sigma_i, ...
    energy)
    %energy and corresponding sources for EACH energy
    energy = [0.662 0.8348 1.173 1.332]; %MeV
    sources = ['Cs-137','Mn-54','Co-60','Co-60'];

    %%%%%Calculate counts emitted from source%%%%%%%%%

    %source properties
    initial_activity = [4.66, 6.27, 0.872 0.872].*(3.7).*10^4; %...
        microcuries to decays/s
    t_elapsed = [123811200 56851200 123811200 123811200];
    half_life = [9.48603e+8 26974080 1.66226e+8 1.66226e+8]; %...
        seconds;
    time_const = log(2)./half_life;
    branching_ratio = [.8510 .999760 .9985 .999826];

    %detector setup
    %area of the detector
    A = pi*(.0431/2)^2; %m^2
    %distance from detector to source
    r = 0.15; %meters

    %Calculations independent of experiment setup and sources
    %solid angle = A/r^2, solid angle fraction = solid angle/...
        sphere
    solid_angle = A/(4*pi*r^2);
    activity = initial_activity.*exp(-1*time_const.*t_elapsed); %...
        counts/second
    collection = 10*60; %seconds
    counts = branching_ratio.*activity.*collection*solid_angle;

    %calculate efficiency and error
```

```
29      efficiency = integral./counts;
30
31      sigma_ia = 0.05.*initial_activity;
32      sigma_t = 12*3600; % error = 12 hours (something seconds);
33      sigma_ca = sqrt((exp(-time_const.*t_elapsed).*sigma_ia).^2 + (...
            initial_activity.*-1.*time_const.*exp(-time_const.*...
            t_elapsed).*sigma_t).^2);
34      sigma_r = 0.0005;
35      sigma_sa = A/(4*pi)*2*r^(-3)*(sigma_r);
36      sigma_c = counts.*sqrt((sigma_ca./activity).^2 + (sigma_sa./...
            solid_angle).^2);
37      error = efficiency.*sqrt(((sigma_i./integral).^2 + (sigma_c./...
            counts).^2));
38  end
```

LaBrmeasurements.m calls LaBrEfficiency and fits a log-log function to resulting efficiency measurements. This data is plotted.

```
peaks = [5.11, 5.62, 6.13, 6.61];
peak_name = ['61DE' '61SE' '61FE' '71SE'];
energy = [0.662 0.8348 1.173 1.332]; %MeV
sources = ['Cs-137','Mn-54','Co-60','Co-60'];


e_integral = [76047.8 23508.3 5411.99 4766.64];
e_integral_error = [283.104 163.671 92.5159 76.1349];


[efficiency, e_error] = LaBrEfficiency(e_integral, ...
    e_integral_error, energy);
xdx = efficiency + e_error;
[slope, intercept]=logfit(energy, efficiency, 'loglog');
[error_slope, error_int]=logfit(energy, xdx, 'loglog');
close all


en=linspace(.500,8.00);
efficiencyfit = 10^(intercept).*en.^slope;
errorfit = 10^(error_int).*en.^error_slope;

stuff = figure('units', 'normalized', 'outerposition', [0 0 1 1]);
redshade=[244, 197, 188]/255;
blueshade=[160, 180, 232]/255;
fill([en';flipud(en')],[(errorfit');flipud((2.*efficiencyfit-...
    errorfit)')],redshade,'linestyle','none');
hold on
plot(en,efficiencyfit,'r--', 'LineWidth', 2)

errorbar(energy,efficiency,e_error, 'bo', 'LineWidth', 2)

```

```
xlabel('Energy (MeV)')
ylabel('Efficiency')
xlim([.5 8])
set(gca, 'FontSize', 30)

e_peaks =10^(intercept)*peaks.^slope;
error1 =10^(error_int)*peaks.^error_slope;
err_peaks =error1-e_peaks;
```