University College London

Nuclear Forensics with Gamma Ray Spectroscopy

Group 14: Kanishk Ganga, John Wong, Pip Benjamin,
Ira Shokar, Lukas Kubin, Ke "Mark" Ma,
Sirui "Sarah" Gao, Yichen "Tiana" Tian

Board Member:
Dr. Ruben Saakyan

Executive Summary

This study aimed to demonstrate a proof of concept of forming a database able to automatically identify gamma-ray-emitting radioactive samples above a 3σ confidence level for given unknown samples. This was achieved by using a high-purity germanium (HPGe) detector to distinguish between unknown radioactive samples using features found in their gamma-ray spectra. The efficiency and resolution of the detector was also investigated with the aid of a Monte Carlo simulation, giving insight to the reliability of the detection of gamma photons at different energies.

The objectives were to familiarise the group with the HPGe detection mechanism, and use this knowledge to calibrate the detector in order to remove systematic error from the measurements taken. Following this, measurements of the background gamma-ray spectrum in the lab were taken before taking measurements of the samples given by the board member. This enabled a comparison between the gamma-ray spectra of the given samples and the lab background spectra. This allowed the creation of a database of known gamma peaks present in each sample.

The resolution with respect to photopeak energy was analysed, using a Gaussian approximation for each identified peak. The samples were then enclosed in separate numbered opaque containers, such that they were only visually identifiable by the number written on the enclosure. Measurements of the blind samples were then taken and compared to the original samples, and statistical analysis was able to quantify the confidence level of peaks identified in the energy spectra. The efficiency with respect to photopeak energy was constructed using a Monte Carlo simulation of the detector.

Photopeak analysis was performed using the 3σ rule. Prominent peaks were found at 662 ± 2 keV in all samples, which corresponded to one of the theoretical gamma emission energies of the isotope ^{137}Cs. Each sample had a different σ value in the ^{137}Cs peaks. Consequently, the confidence level of the ^{137}Cs measured inside the blind sample was compared to that of the samples in the database. The comparison resulted in correctly identifying the blind samples.

The proof of concept was shown to be successful with identifying the samples with a simple database. This concept could be developed further with longer measurements, where a more detailed database could be developed by determining more individual elements inside a given sample. Additionally, using detector shielding would improve the resolution of the peaks by increasing their corresponding confidence levels. The simulation successfully produced theoretical values for the absolute efficiency of the detector. This too can be improved with more time, as it would then be possible to produce a more detailed model of the detector.

Please note: The COVID-19 disruption resulted in project completion issues. Group member Ke Ma returned to China on 16/03/2020 due to the outbreak, resulting in limited and very infrequent connection to the group. This has impacted the completion of the *Data Analysis & Results* section, specifically the experimental absolute efficiency calculations (with corresponding values, uncertainties & graphs), and *Conclusions* section, as he has not been able to access the document or continue calculations.

Contents

Allocation of Work

The contributions of each member have been summarised in the table below to clearly show the collaborative nature of the group. Work was distributed according to people's desire to do the work, followed by consideration for fairly spreading the workload. Kanishk lead the long-term organisation of group using a Gantt chart he created and implemented primary internal deadlines. Pip was the team's communications officer; he organised meeting times and created initial agendas for most meetings, as well as leading the short-term organisation of group and ensuring internal deadlines were met on a weekly basis. Kanishk, Pip and John were responsible for the risk assessment.

Task	Kanishk	John	Pip	Ira	Lukas	Ke	Tian	Sirui
Gantt Chart	✓		✓				✓	
Background Research	✓	✓	✓	✓	✓	✓	✓	✓
Detector 1 Operation			✓			✓		
Detector 2 Operation							✓	✓
Lab Book Record			✓			✓	✓	✓
Data Analysis: Data Import Code	✓	✓		✓				
Data Analysis: Calibration Coefficients						✓	✓	✓
Data Analysis: Peak Finding	✓	✓	✓					
Data Analysis: Resolution & Efficiency		✓				✓		
Geant4 Simulation				✓	✓			
Meetings: Chairing	✓	✓	✓	✓				
Meetings: Minute-taking	✓			✓	✓		✓	✓
Final Report: Executive Summary	✓		✓					
Final Report: Introduction	✓			✓	✓			✓
Final Report: Experimental Procedure			✓		✓			
Final Report: Geant4 Simulation				✓	✓			
Final Report: Data Analysis	✓	✓						
Final Report: Conclusions		✓	✓	✓	✓	✓	✓	
Final Report: Editing/Formatting	✓		✓					
Poster Design/Content	✓	✓	✓		✓			

Chapter 1

Introduction

1.1 Background

Nuclear forensics is the analysis of nuclear materials to evaluate their characteristics, such as physical and chemical properties. By utilising gamma-ray spectroscopy (GRS) isotopic analysis can be performed. This is done by detecting the energies of the gamma rays emitted from radionuclides present in the material. This has powerful applications in the health and safety sector [1]; the illicit trafficking of nuclear materials may be monitored using similar techniques to those discussed in this study. Nuclear forensics is also useful in nuclear disarmament; GRS allows for radioactive analysis of nuclear weapons without disclosing confidential weapon design. Nuclear waste may also be analysed to give a quantitative description of their safety with respect to their containers, resulting in safer disposal.

In this experiment, three samples emitting gamma radiation were distinguished from each other by studying their photopeak energy spectrum, which is characteristic of the elements decaying within each sample. This project aimed to identify the samples based on their gamma-ray emission, measured by a gamma-sensitive detector. An investigation into the resolution and efficiency of the detector at different photopeak energies was also performed. It is seen later that there exists an explanation for the efficiency and resolution trend with respect to photopeak energy. Though this project only used three samples, it serves as a proof of concept for the creation of a much larger database that could computationally identify samples based on their raionuclide fingerprint for a given unknown sample.

GRS is a non-destructive method as only the gamma emissions are measured. Nuclei which decay via β-emission emit gamma rays by exciting to a state that, when relaxed to ground state, emit a gamma-ray photon with energy equal to the energy difference between the initial and final states. Additionally, if a parent nucleus undergoes β^+ decay, the emitted positron can annihilate with electrons, producing gamma radiation [2]. There exist many other sources of gamma rays, though not all are relevant processes to what may naturally occur in the samples given. Gamma emission is characteristic of the radionuclide and thus can be used to identify the radioactive element within a material.

It must be noted that to detect electromagnetic radiation such as gamma, a detector must be able to produce a charge pulse due to the gamma detection such that a connected interface can understand this as an electrical signal. Gamma-ray photons must interact with matter to transfer the energy of the photon to electrons within the conducting material of the detector. The probability of an interaction is dependent on the size of the interacting atoms, hence materials with greater atomic number will have a higher interaction probability, and therefore increase the sensitivity of the detector. Germanium is favoured for gamma-ray detection due to its high atomic number (relative to popular semiconductor materials like silicon) [3]. The attenuation (reduction in gamma-ray intensity due to an absorbing material) of germanium can be seen in Figure 1.1. At different energies, different interaction mechanisms dominate, with each mechanism resulting in detections.

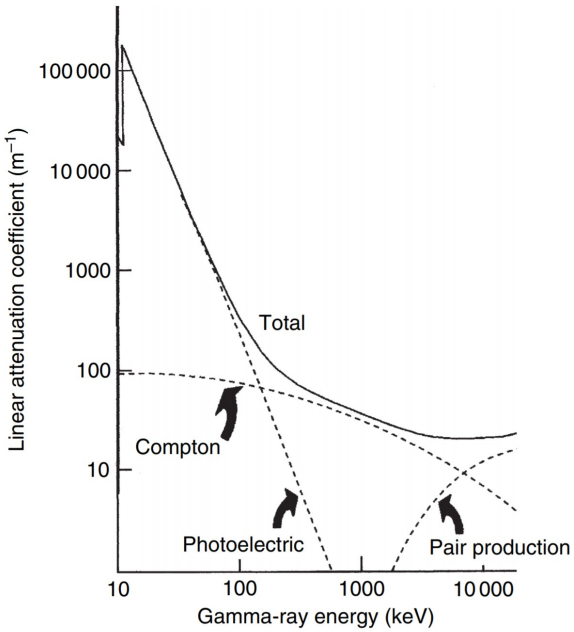


Figure 1.1: Linear attenuation of germanium with respect to energy (Gilmore, 2008, pp.25–26) [3]. At low energies, the photoelectric effect dominates. At high energies, pair production dominates. In between these two regions, Compton scattering dominates.

1.2 Semiconductor Detector Mechanics

For this study, a high-purity germanium (HPGe) detector was used to measure the number of photons emitted from given samples. A germanium detector is a type of semiconductor diode with p-i-n structure, operated in a reverse bias mode [4]. The "p-i-n" refers to the semiconductor layers in contact, i.e. a p-type semiconductor and an n-type semiconductor are in mutual contact with an intrinsic layer between them. The intrinsic/depletion layer is sensitive to ionising radiation. Photons that interact with the depletion region produce charge carriers (electrons and holes) that travel to the corresponding n- and p-type layers respectively due to an electric field that exists across the depletion region under reverse bias. A diagram of a p-type coaxial detector can be seen in Figure 1.2. Reverse bias increases the force acting on the charge carriers, maximising charge collection and produces a larger depletion region. For semiconductor detectors, the depletion layer thickness, d, is:

$$d = \left(\frac{2\varepsilon V}{eN}\right)^{\frac{1}{2}} \tag{1.1}$$

where $\varepsilon \equiv$ dielectric constant, $V \equiv$ reverse bias potential difference, $N \equiv$ net impurity concentration in the semiconductor material, and $e \equiv$ electron charge [5]. Under reverse bias, maximised depletion width is achieved where N is minimised. The charge produced by the photon is proportional to the energy of the incident gamma photon. A charge-sensitive pre-amplifier then converts this charge into an electric potential pulse that can be logged computationally. In semiconductors like germanium, there exists a small band gap separating the valence and conduction zones and therefore must be cooled to reduce the production of thermal charge carriers. Thermal charge carriers produce a leakage current that acts as noise, reducing the resolution of the detector. The detector is cooled in a vacuum chamber to avoid condensable contaminants affecting the surface of the detector. This investigation uses two detectors; one from Mirion Technologies and another from ORTEC.

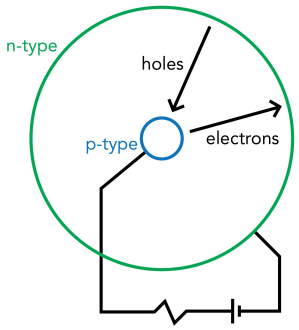


Figure 1.2: Cross section of a coaxial p-type detector perpendicular to the cylindrical axis. Under reverse bias, the electrons move from the p-type material to the n-type material, and the holes move from the n-type material to the p-type material.

1.3 Detector Resolution & Efficiency

The resolution of a detector characterises the accuracy of its response to a mono-energetic source. For a Poisson distribution, if N charge carriers are generated from a gamma interaction, a standard deviation of $\sqrt{N}$ is expected to describe the statistical error in this measurement [6]. Though each photon is quantised and therefore the charge carriers producing an electrical pulse is discrete, in the limit of large number of pulses, a Gaussian approximation can be made for the pulse peak [7];

$$G(E) = \frac{A}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(E-E_0)^2}{2\sigma^2}\right) \tag{1.2}$$

where $A \equiv$ area of Gaussian peak, $E_0 \equiv$ mean pulse height corresponding to the same peak, i.e. photopeak energy, $\sigma \equiv$ standard deviation from E_0. Resolution R is then given by:

$$R = \frac{\text{FWHM}}{E_0} \equiv \frac{\Delta E}{E} \tag{1.3}$$

where FWHM $\equiv$ full width at half maximum of the full-energy peak, related to the Gaussian approximation by FWHM $= 2\sqrt{2\ln 2}\sigma$. This definition assumes background has been considered. R is dimensionless and may instead be expressed as a percentage. It is then evident that a smaller FWHM and/or larger photopeak energy will result in better resolution.

For a given photon, the number of counts measured by a detector is dependent on the crystal's size, shape, and atomic number within the detector. Counts measured also depends on the sample's radioactivity, its distance to the detector, and the medium the photon travels through. Absolute efficiency of the detector, ε_{abs}, is given by the ratio of the number of photons detected by the HGPe crystal, N_{meas}, to the number of photons emitted by the source, N_{emit} [8]. That is to say;

$$\varepsilon_{\text{abs}} = \frac{N_{\text{meas}}}{N_{\text{emit}}} \tag{1.4}$$

The number of gamma photons emitted can be summarised using:

$$N_{\text{emit}} = AYt \tag{1.5}$$

where $A \equiv$ activity of the source, $Y \equiv$ gamma photon yield, $t \equiv$ exposure time. The activity of the source follows a typical exponential decay:

$$A = A_0 \exp\left(-\frac{\ln(2)}{\tau}t\right) \tag{1.6}$$

where $A_0 \equiv$ activity at the known calibration date, $\tau \equiv$ half-life of radionuclide.

1.4 Sample Origins

Radionuclides commonly present in sediment and soil samples include ^{238}U, ^{232}Th and ^{40}K [9]. These radionuclides occur naturally, though their observed levels can be affected by human processes such as mining and ore processing, as well as the discharge of waste from the nuclear industry. Many of these radionuclides are considered to be cancer-causing, which makes studying their concentration and distribution important in assessing their impact on human populations.

Three sediment samples were given; these were to be analysed for any characteristic signatures. The samples tested in this experiment are all from the Irish sea, which is considered to be one of the most radioactively contaminated seas in the world. [10]. The main radioactive pollutants are likely to have originated from Sellafield, which is a nuclear site close to Seascale. Since 1952, low-level radioactive waste has been discharged into the Irish sea. A large number of pollutants (^{137}Cs, ^{129}I, etc.) were predicted to be discharged from the nuclear site and distributed into the wider ocean through the pipeline. Instead, large amounts of radioactive material was dispersed onto coastal sands [11].

Each sample was contained within a Marinelli beaker, the shape of which is depicted in Figure 1.3 [12]. This design allows for almost complete coverage of the HPGe crystal. This maximises the number of incident photons on the detector. These same samples were later used as the blind test samples, by blindly shuffling and concealing them in opaque 3D-printed containers (further detail in Chapter 2.3). Two calibration samples were also used, containing ^{60}Co and ^{137}Cs respectively, to calibrate the detectors and to calculate safe operation distances (see Chapter 2.2 for dose calculation).

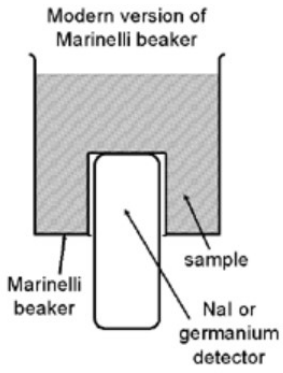


Figure 1.3: Cross-section of a Marinelli container parallel to the cylindrical axis (Oak Ridge Associated Universities, 2011) [12]).

1.5 Simulation in Geant4

Simulating the emission of gamma rays from our sample would be beneficial as a comparison with the experimental results would highlight any significant differences, leading to the discovery and/or explanation of any anomalous data. Due to the complexities involved in modelling the samples, the purpose of the simulation was narrowed to reproduce the detector and gamma photon interactions to calculate the absolute efficiency of the detector. Within the time-frame, it would not have been possible to produce a novel working simulation from scratch so a pre-built toolkit, Geant4 (GEometry ANd Tracking), was instead used [13]. Geant4 is an international collaboration and is widely used in nuclear experiments. A non-exhaustive list of experiments

using Geant4 includes ATLAS, CMS and LHCb at the Large Hadron Collider operated by CERN, Fermilab, and European Space Agency [14].

Geant4 is a toolkit that simulates the passage of particles through matter to simulate the response of detectors. It contains physics packages that cover a range of processes such as electromagnetic, hadronic and optical interactions. A simulation of the HPGe detector and the Marinelli container containing the sample were modelled before defining and simulating the type of interaction that would occur. Due to the random nature of decay mechanisms, and that it is unknown which particular particles interact or their exact positions, it is not possible to produce a deterministic simulation of the events. Geant4 provides Monte Carlo simulation tools, using stochastic processes to provide a representation of the interactions in experimental apparatus.

Chapter 2

Experimental Procedure

2.1 Apparatus

Two different HPGe detectors located in different areas of the laboratory were used to analyse the gamma-ray spectra of the sediment samples. The first detector was a *Mirion p-Type Coaxial Germanium Detector*. The second detector was an *ORTEC Solid-State Photon Detector* [15].

The set-up for the first detector (see Figure 2.1) consisted of a large liquid nitrogen dewar, a pre-amplifier, and a HPGe crystal connected to a Lynx DAQ data acquisition unit. The liquid nitrogen dewar was responsible for cooling the detector to 77K to minimise thermal charge carriers and form a large depletion region within the HPGe semiconductor detector. This allowed for high resolution spectrometry [16]. The Lynx DAQ unit converted the electrical pulses detected by the HPGe crystal into a multi-channel digital output. This output could be plotted with channel number on the horizontal axis, and number of counts on the vertical axis. The detector was calibrated to find the energy value associated with each channel using the known gamma ray emission lines of given radioactive sources.

The entire procedure was performed in parallel with the second HPGe detector (see Figure 2.2). The second detector was located on the opposite side of the laboratory, with a similar set-up to the first detector. When compared to the first detector, the second detector had minor hardware differences, with the major difference being the software used to convert the analogue detector signal to a useful digital output. The first detector used a Lynx DAQ connected to a computer running Apple's MacOS, whereas the second used an ORTEC digital signal processor connected to a computer running Windows OS.

Both detectors were set to collect data using their respective software. To prevent damage to the HPGe detector, the dewar was first filled with liquid nitrogen by the board member to cool the system. Operation of the system only took place after the detector was sufficiently cooled. To prepare each detector for data collection, the high voltage supply needed to be increased gradually to its maximum before taking any measurements. The software for the first detector had a pre-existing function to configure the voltage, and the second used a program called *MCB Configuration*.

Starting a measurement was initiated by clicking `START` in the software's user interface. To analyse the gamma-ray emission of the calibration samples and sediment samples, the samples were carefully placed on top of the HPGe detector. Samples not in use were placed a suitable distance away from the detector while spectra were being measured. This ensured that the characteristic spectrum of each sample was measured with minimal noise due to the other radioactive samples.

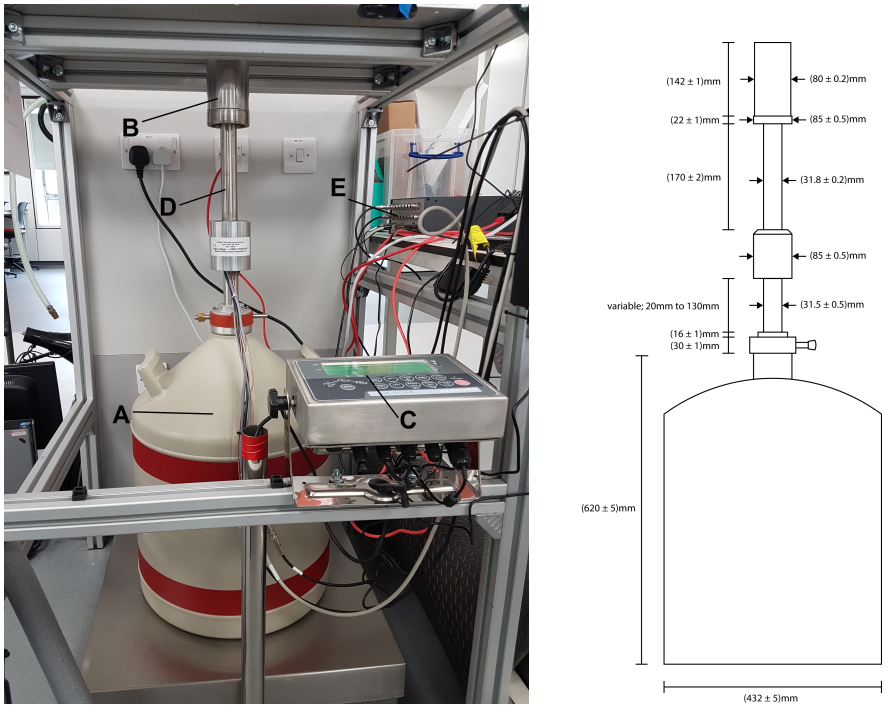


Figure 2.1: (Left) Photograph of the Mirion p-Type Coaxial Germanium Detector, showing A: liquid nitrogen dewar, B: HPGe crystal, C: detector temperature display, D: cooling finger, E: Lynx DAQ unit. (Right) Dimensions of the Mirion detector.

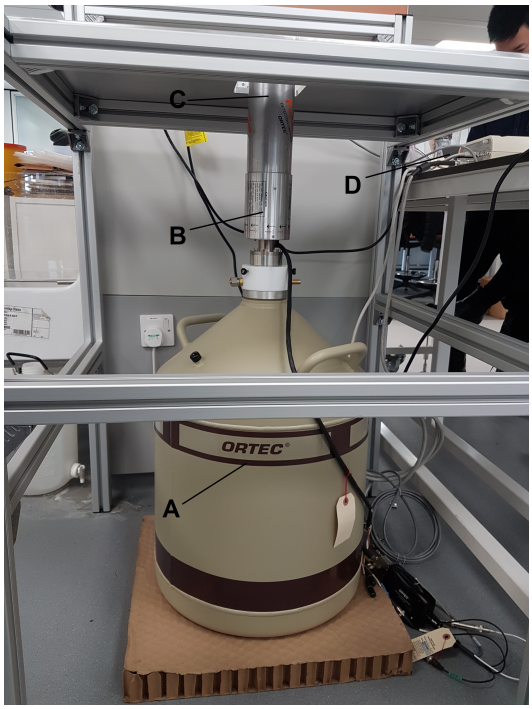


Figure 2.2: Photograph of ORTEC Solid-State Photon Detector, showing A: liquid nitrogen dewar, B: cooling finger, C: HPGe crystal, D: ORTEC digital signal processor.

2.2 Radiation Dose & Calibration

The theoretical dosage ensured present members were within safe radiation exposure limits. The radiation dose received by being in the proximity of the samples is found by current activity and known activity. Both of the ^{60}Co and ^{137}Cs calibration samples were produced in 1991, allowing calculation of the exposure time, t. The activity, A_0, for both samples on the calibration date was 0.37 MBq. The values for half-life and specific activity of both calibration sources were used in calculating the radiation dose [17]. From Equation 1.6, the activities for ^{60}Co and ^{137}Cs were found to be 8.16×10^{-3} MBq and 0.19 MBq respectively. External exposure for a point like source 30 cm away for an activity of 1 MBq is also known [17].

To determine the external exposure on the date of measurement, the ratio of current activity of the source to the activity of 1 MBq is multiplied by the external exposure values. The detector operator was measured to be sitting $\sim$ 150 cm away from the source. The calculated exposure values were therefore multiplied by the square of the ratio of the distances. For ^{60}Co, the external exposure to β-particles was 4.12×10^{-6} mSvh^{-1} and to gamma and X-rays 1.26×10^{-6} mSvh^{-1}. Similarly for ^{137}Cs, the exposure to β-particles was 1.62×10^{-3} mSvh^{-1} and to gamma and X-rays 8.14×10^{-6} mSvh^{-1}. The calibration sources were used for two laboratory sessions each lasting three hours. Therefore, the total external exposure to the detector operator from both sources was 9.8×10^{-3} mSv, which is approximately 100 times smallerthan the annual radiation dose limit of 1 mSv [18]. That is to say, the detector operator and others in the laboratory equal to or greater than 150 cm away from the detector will be well within the radiation limits imposed by the regulation.

The HPGe detector was calibrated using the aforementioned ^{60}Co and ^{137}Cs sources. High intensity peaks were expected at particular energies in the gamma ray spectrum for ^{60}Co and ^{137}Cs [17]. In the Lynx DAQ output (see Figure 2.1), each digital channel was assigned an arbitrary energy value, with the energy interval between channels determined by a set of linear calibration coefficients. The software output headers for information regarding each measurement. This included automatically set offset and slope coefficients, live time elapsed, and real time elapsed. The calibration coefficients used to present the detected spectra in terms of real energy values were calculated using the known energy peaks of the calibration sources. The full method for calculating these coefficients is described in Chapter 3.

Start Time	Mar.10	2020 at 11:52:36 AM	
Energy calibration	offset:0	Slope:0.75	Quadratic:0
Live Time(s)	605.33		
Real Time(s)	607.92		
Elapsed Computational	0		
Spectrum			
Channel	Energy (keV)	Counts	
1	0.75	7	
2	1.5	1308	
3	2.25	337	
⋮	⋮	⋮	

Table 2.1: Screenshot of .csv data output from the Mirion detector, with a header of important values for each measurement, including the offset and slope calibration coefficients, live time and real time.

The second detector did not assign energy values to the channels; the output was instead a list of values for the counts recorded in each channel without headers for each measurement, unlike the first detector. The detectors had different software for imaging the scanned spectra, and both were insufficient to effectively complete data analysis. A Python script for data analysis was therefore written to analyse the measured spectra. Details of the code can be found in Chapter 3 with full code available in Appendix D — Python Code. The calculated calibration coefficients were then used in the data analysis script to reproduce the gamma ray spectra, and compare peaks within the spectra to known emission lines from different radionuclides [17].

2.3 Sample Analysis

The three labelled sediment samples were contained in plastic Marinelli beakers. For identification purposes, each sample was given a label, these were *'11DRY'*, *'12MEDIUM'* and *'SAMPLEB'* respectively, and were sourced from West Kirby Beach in Merseyside, England, according to the hand-written notes on the lid of each sample. This information is given in full in Table 2.2. The gamma spectrum of a sample was produced by placing it over the HPGe crystal and allowing the detector to detect gamma rays for a set time. Figure 2.3 shows one of the samples placed on top of the Mirion HPGe detector.

Full Sample Title	Latitude	Longitude	Depth/cm	Sample Mass/kg
11 DRY WEST KIRBY	53°22'30"N	3°11'26"W	0-10	0.3696
12 MEDIUM WET	53°22'25.634"N	3°11'22.557"W	20-30	0.5497
SAMPLE B DRY	53°22'29"N	3°11'22"W	40-48	0.5104

Table 2.2: Information taken from the lids of the three West Kirby Beach sediment samples given to the group by the board member. The sample mass here was, in some cases, taken by subtracting the mass of the container from the gross mass indicated on the sample lid.

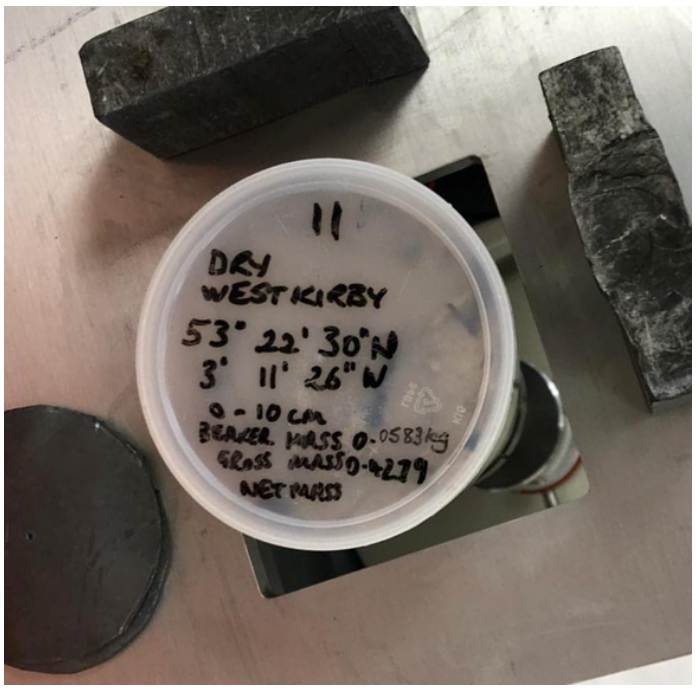


Figure 2.3: Photograph of the sample *11DRY* placed on top of the Mirion HPGe detector.

A background reading of 10 minutes was taken each session before collecting a 10 minute reading from each sample. This helped identify whether peaks were present in the sample spectrum due to radionuclides in the sample itself or due to gamma rays in the lab background radiation. The background and samples were measured for 600 ± 10 s. The variation in time (and therefore in counts) between the background and sample readings was fixed by scaling the background or sample time such that the live time of the background and sample were equal. This meant peaks in the sample could be assessed in comparison to the background, allowing statistical analysis to indicate whether each peak was significant, or a product of random statistical fluctuation.

After completing data collection from the samples, the blind samples were created by concealing each sediment

sample in a 3D-printed enclosure that covered their labels, as seen in Figure 2.4. After collecting data from each blind sample and an accompanying background reading for the session, the same previous procedure was used. The blind spectra were examined for any identifying features that may be present due to their characteristic gamma-ray emission. Analysis of these blind spectra alongside the previous non-blind spectra allowed for identification each of the samples.

Figure 2.4: Labelled containers for blind test sediment samples. Sediment colour and hand-written notes on the lids are obscured.

After initially performing 10 minute measurements of the background and each of the samples, the group performed further 30 minute measurements to assess the impact of longer measuring time on the statistical significance of the results. Due to the Poissonian nature of the data, the uncertainty for a given peak is expected to scale with the square root of the number of counts for that peak, thus reducing uncertainty by increasing the measuring time.

Due to time constraints on the availability of the lab and the necessity of having a trained demonstrator available whilst data acquisition was taking place, 30 minutes was deemed the maximum possible length of time to measure the data. This scaling meant the number of counts was expected to increase by a factor of 3, and the uncertainty in the counts would increase by a smaller factor of $\sqrt{3}$. Data sets for both the 10 and 30 minute measurements were then analysed to see whether the time difference caused any statistically significant differences.

Chapter 3

Data Analysis

3.1 Calibration Calculation

The Lynx DAQ unit (connected to the first detector) output arbitrary energy values for each digital channel. These values were calibrated using the known gamma radiation sources, ^{60}Co and ^{137}Cs. For ^{60}Co there exist gamma peaks at 1173 keV and 1333 keV, while ^{137}Cs has gamma peaks at 32 keV, 36 keV and 662 keV [17]. Due to large background. in the low-energy region, only the 662 keV gamma peak is considered.

Using the first detector, three peaks were found from the collected data from the ^{60}Co and ^{137}Cs sources. These peaks were at 489 ± 2 keV, 866 ± 2 keV and 984 ± 2 keV respectively. The known peaks were plotted against the measured peaks, and a line of best fit was fit to the data. Since the relationship between the known energy peaks and the measured energy peaks is linear, the arbitrary energy values E could be calibrated by finding the gradient and y-intercept of a straight line of general form $y = mx + c$. However, a 'slope' factor was also found in the data set, where the arbitrary values were scaled according to this slope factor. This contributed an extra scaling for the calibration, so the gradient of the general linear equation for calibration was modified, seen in Equation 3.1. The results of the Mirion detector are summarised in Figure 3.1.

$$E_{\text{calibrated}} = \left(\frac{m}{\text{slope}}\right) E + c \tag{3.1}$$

The second detector was calibrated using a similar method. However, the second detector was not connected to a conversion unit, so no arbitrary energy channels were assigned to the recorded data. In the data set, only the channel number and the counts were provided, so a scaling factor was also not found. ^{60}Co and ^{137}Cs were used once again in the calibration, where three peaks were found in three different channels. The calibration of channel number to energy was performed by plotting a graph of channel peaks against the 3 known peaks. Like the fist detector, a line of best fit was fit to the data, where the gradient and y-intercept of the straight line was found to calibrate the channel number to the energy of the peaks. This is also seen in Figure 3.1.

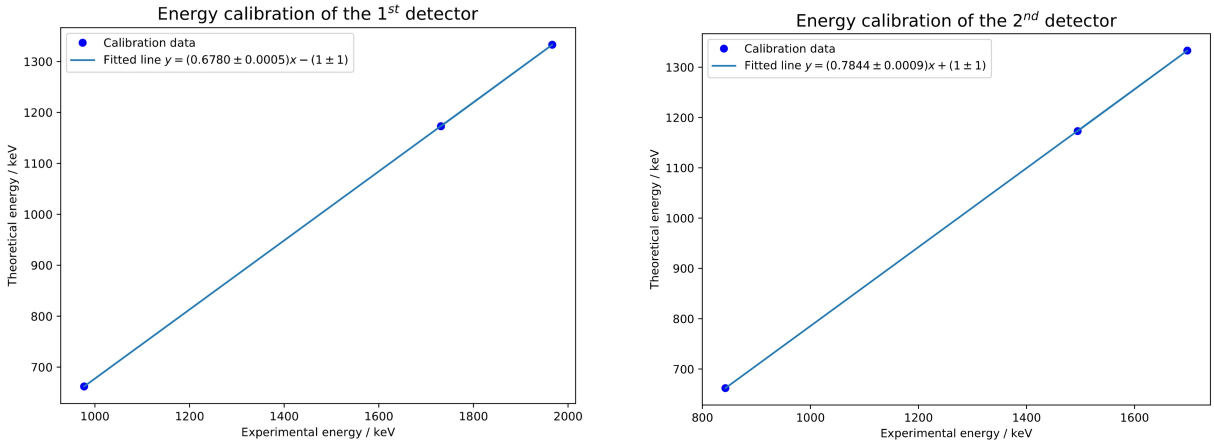


Figure 3.1: Energy calibration graphs for the Miron detector (left) and the ORTEC detector (right). The equation for the line of best fit contained parameters to calibrate the energy channels of the detectors.

3.2 Detector Resolution

Before analysing the peaks, the energy resolution of the peaks needed to be calculated. The energy resolution describes the the ability for the detector to distinguish between two energy peaks. The resolution R is given in Equation 1.3. The uncertainty of the energy values of the peaks could also be calculated; this is the error in the energies when identifying the characteristics lines.

The peaks from the *11DRY* sample were used to find R for both detectors. A peak finding function in Python was used to identify peaks in the sample data. After identifying the peaks, a Gaussian function was fit around two energy intervals of the peak using a curve fit function. The fit contained three parameters with their respective uncertainties, defined by the following equation:

$$f(x) = A \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.2}$$

The three fitted parameters were: $A \equiv$ normalisation constant defined by the height of the Gaussian curve, $\mu \equiv$ mean of the data distribution, $\sigma \equiv$ standard deviation of the data. The formula FWHM $= 2\sqrt{2\ln 2}\sigma$ was used to find the FWHM of the Gaussian fit. The uncertainty in the FWHM was calculated using propagation of uncertainty and the fitted σ uncertainty [6]. Then, R was calculated using Equation 1.3, where the uncertainty was also calculated using the aforementioned method.

A graph for R against the centre of the energy peak E was plotted for each detector. From the sets of data, a curve could be fitted to determine the energy resolution scale of each detector. The fitted curve used the following equation:

$$R = \frac{A}{\sqrt{E}} + B \tag{3.3}$$

The graphs for the energy resolution for detectors 1 and 2 are seen in Figures 3.2 and 3.3 respectively. The legends of the graphs contain the fitted parameters with their respective uncertainties and the reduced χ^2 of the fit. The reduced χ^2 value for the fit examined the goodness of the fit. From the figures, the reduced χ^2 values were 1.27 and 1.46 respectively. This indicates that the curve was a good fit for the data. As a result, each curve was used as an energy resolution model for the corresponding detector and was used to find the energy window of the peaks.

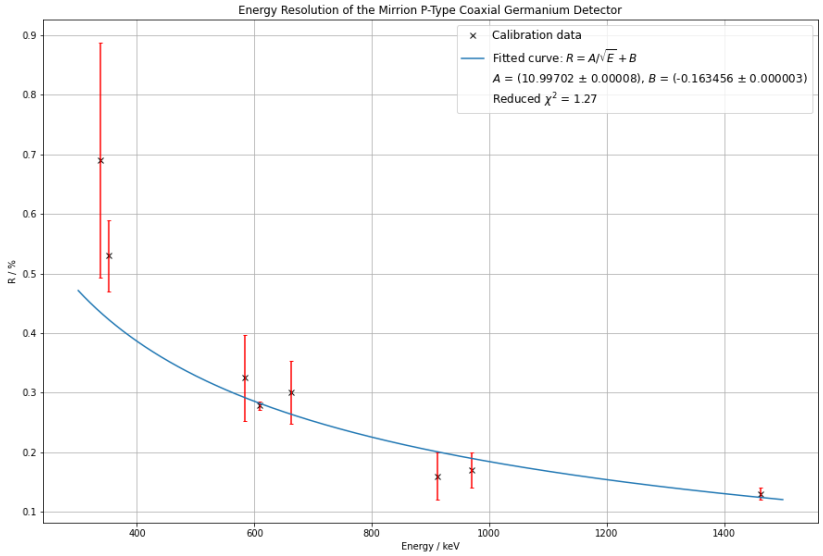


Figure 3.2: The fitted energy resolution of the first detector. The resolution showed the ratio of the uncertainty in the energy to the mean energy in percentage. The fitted curve was used to determined the resolution of a particular energy peak.

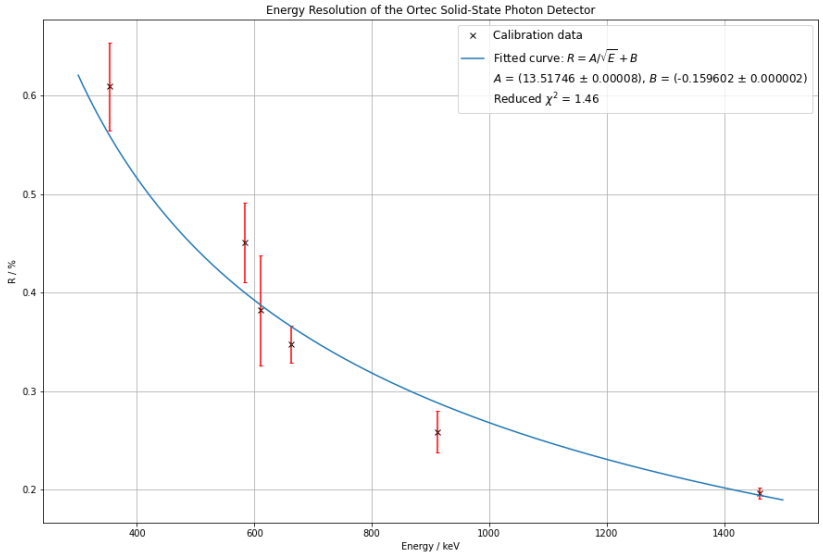


Figure 3.3: The fitted energy resolution of the second detector. The fitted curve was used to determined the energy resolution. The energy resolution of the second detector was lower than the first detector.

3.3 Identifying Characteristic Lines

Graphs of counts against energy were plotted. Each graph contained the scaled background counts and the data obtained from the samples. For each graph, peaks were identified in a local region by using a peak-finding function in python. A mathematical approach then quantified the discovered peaks; the 3σ rule was used to demonstrate that the data indicated the presence of a peak with at least 68% confidence.

This peak analysis was done because the background also contained peaks in the same region that were visibile in the graph. This resulted in some background peaks overlapping with data peaks, making the origin of the peaks unclear. In order to confidently identify the data peaks as evidence/discovery from the sample, the number of sigmas in the difference of the two peaks was calculated. With the calculated uncertainty, the sigma analysis could be performed using the formula below:

$$\sigma = \frac{N_{\text{data}} - N_{\text{background}}}{\sqrt{N_{\text{data}} + N_{\text{background}}}} \tag{3.4}$$

Using the 3σ rule, if a given data peak has at least 3σ in the difference of data counts and background counts, the peak could be presented as evidence of gamma-ray emission at this energy in the sample. In an even better scenario; if the sigma could be calculated to be 5σ or above, this could be present as discovery of gamma-ray emission at this energy in the sample data with 99.7% confidence.

The uncertainty in the photopeak energy was calculated by multiplying the energy resolution model value (Figures 3.2 and 3.3) by the energy value of the peak. The identified energy peaks, their uncertainty, and their sigma values were plotted on the corresponding spectrum measured from the sample. An example of the graph with the peak information is seen in Figure 3.4. Each sample was measured two times on each detector, so a total of 12 graphs were plotted; these can be seen in Appendix C. The collection of peaks on a single graph was used as an entry in a database for the samples. This would then suggest that a particular sample was present if some/all of the peaks on a single graph were detected with similar sigma significance values. This database was used to identify the blind samples. A table summarising the database is seen in Tables 3.1a and 3.1b.

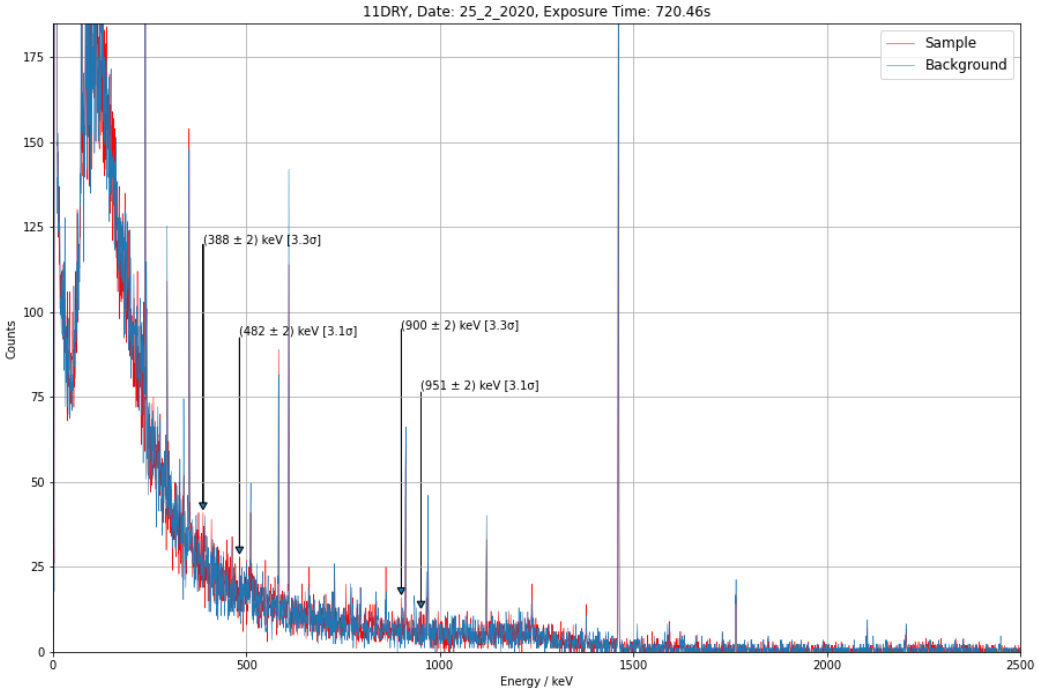


Figure 3.4: A graph of counts against energy of the *11DRY* sample for the first detector. Energy peaks with 3σ or above were indicated on the graph with an arrow. Above the arrow is the energy at which the peak is present, with uncertainty calculated from the resolution model. The sigma value of the peak is next to the photopeak energy value, in square brackets.

Peak (keV)	11DRY	12MEDIUM	SAMPLEB
1	388 ± 2 [3.3σ]	343 ± 2 [3.3σ]	420 ± 2 [3.1σ]
2	482 ± 2 [3.1σ]	469 ± 2 [3.1σ]	566 ± 2 [3.3σ]
3	900 ± 2 [3.3σ]	485 ± 2 [3.1σ]	662 ± 2 [4.7σ]
4	951 ± 2 [3.1σ]	662 ± 2 [5.3σ]	1180 ± 2 [3.1σ]
5	978 ± 2 [3.1σ]	662 ± 2 [11.0σ]	-
6	-	1140 ± 2 [3.2σ]	-
7	-	1180 ± 2 [3.4σ]	-

(a) Mirion Detector

Peak (keV)	11DRY	12MEDIUM	SAMPLEB
1	662 ± 2 [3.2σ]	395 ± 2 [3.1σ]	662 ± 2 [5.2σ]
2	-	662 ± 2 [12.8σ]	-
3	-	663 ± 2 [7.7σ]	-
4	-	690 ± 3 [3.2σ]	-

(b) ORTEC Detector

Table 3.1: A table summarising the database created from the samples. The sigma value is in the square bracket next to the energy values and the corresponding uncertainty.

3.4 Blind Sample Comparison

After the database was created, the blind samples were then measured. Peaks were identified using the same method as before, this time comparing the collection of peaks in a blind sample to one of the three entries in the database. The peaks found in the blind samples are summarised in Tables 3.2a and 3.2b.

Peak (keV)	Blind 1	Blind 2	Blind 3
1	432 ± 2 [3.3σ]	662 ± 2 [3.9σ]	400 ± 2 [3.2σ]
2	662 ± 2 [11.0σ]	700 ± 2 [3.1σ]	662 ± 2 [3.4σ]
3	890 ± 2 [3.8σ]	-	792 ± 2 [3.0σ]
4	926 ± 2 [3.5σ]	-	974 ± 2 [3.2σ]
5	1121 ± 2 [3.0σ]	-	-

(a) Mirion Detector

Peak (keV)	Blind 1	Blind 2	Blind 3
1	662 ± 2 [11.0σ]	662 ± 2 [3.6σ]	-
2	905 ± 3 [3.4σ]	977 ± 3 [3.1σ]	-
3	1462 ± 3 [4.0σ]	1074 ± 3 [4.1σ]	-

(b) ORTEC Detector

Table 3.2: Results of the peaks found in the blind samples. There were no peaks found with 3σ or above confidence in Blind 3 for the ORTEC detector.

In all samples, a peak of 662 ± 2 keV was present. This corresponds to ^{137}Cs. Some of the other peaks identified above a 3σ confidence level were considered too inconsistent to be used for identification as they only appeared once across all measurements. As a result, the σ confidence in the 662 keV peak was used as an identification tool due its reliable presence.

12MEDIUM had the highest confidence level for ^{137}Cs, at 11σ and 12.8σ (as measured by detectors 1 and 2 respectively). Blind 1 also had this characteristic 11σ confidence as measured by both detectors. It was therefore concluded that Blind 1 was *12MEDIUM*.

Similarly, Blind 2 showed a weak presence of ^{137}Cs, with confidence levels of 3.6σ and 3.9σ. Additionally, Blind 3 also had a weak presence of ^{137}Cs, with a confidence of 3.4σ as measured by detector 1, and no identifiable peaks measured by detector 2. This made it difficult to identify between Blind 2 and 3 as the confidence levels were all between 3.3σ and 4σ. The solution is clearer when considering the database samples. *SAMPLEB* has a ^{137}Cs peak with a confidence of 4.7σ and 5.2σ. *11DRY* has a weaker presence of ^{137}Cs, with a confidence level of 3.2σ measured by detector 2, and no peak identifiable by detector 1.

From this it is clear that Blind 2 is *SAMPLEB* due to the higher confidence levels in ^{137}Cs, and Blind 3 is *11DRY* due to only 1 of 2 detectors finding a ^{137}Cs peak with confidence higher than 3σ in each sample's set of measurements. With Blind 3/*11DRY*, the lack of ^{137}Cs above 3σ in one of the detectors suggests that the presence of ^{137}Cs is only just above the 3σ threshold, with statistical fluctuations potentially being the reason that the peak is not detected.

Chapter 4

Geant4 Simulation

4.1 Operating System & Boulby Experiment

Geant4 currently uses multi-threading (the ability to utilise more than one CPU thread to execute calculations in parallel). As there is currently no support for multi-threading within Windows, a Unix/Linux based operating system was required [19]; the High Energy Physics Group's Linux Cluster was used through remote command-line login and execution through a secure shell. Geant4 is pre-installed on these machines accesible via Scientific Linux 7 [20]. However, due to dependency issues with OpenGL, the visualisation API that draws graphics, a CentOS 7 [21] based machine was instead used. The choice in system was justified by CERN migating to CentOS from Scientific Linux in 2015.

Geant was originally produced in Fortran but was rewritten for object-oriented programming, written in C++. This language was unfamiliar to the collaborators in the project; a substantial proportion of the time allocated to building the simulation was used in debugging and understanding syntax. Geant4 provides a number of examples; a simple sampling calorimeter setup was modified to represent the detector system used to measure gamma-ray emission.

Unlike the detectors used, a similar detector experiment known as the *Boulby Underground Germanium Suite* (BUGS) is located underground and uses lead shielding to remove much of the background activity. It does however use a p-type coaxial detector similar to the one used in this investigation [22]. BUGS required a corresponding Geant4 programme, similar to the structure required to model the detector being used. Professor Ghag worked on this and was kind enough to provide the programme used at BUGS. Running the given code as it was returned a memory segmentation fault, an error that occurs when a program attempts to access a prohibited part of the system memory. With limited experience in Linux, C++ and the programme itself, debugging this error proved very challenging. The detector simulation was instead written by modifying one of the basic examples provided by Geant4.

4.2 Procedure

The simulations were built based on two of the B4 examples provided by Geant4, simulating a simple calorimeter. The B4a example implements a `SteppingAction` class, which step by step, collects energy deposits and tracks length at the absorber and gap layers of the detector, and then passes the data to the `EventAction` class which at the end of the event analyses the data and fills the histograms and ntuples. The first step was to change the geometry of the detector. All absorption and gap layers were deleted and a new set of geometry parameters for the germanium detector was defined. Using the `G4Tubs` class, a solid was described that had the shape of a cylinder with the dimensions of the first detector (see Figure 2.1). Since the detector used in the experiment was coaxial, a second smaller cylinder was subtracted from the solid to get a shape to more accurately resemble the detector.

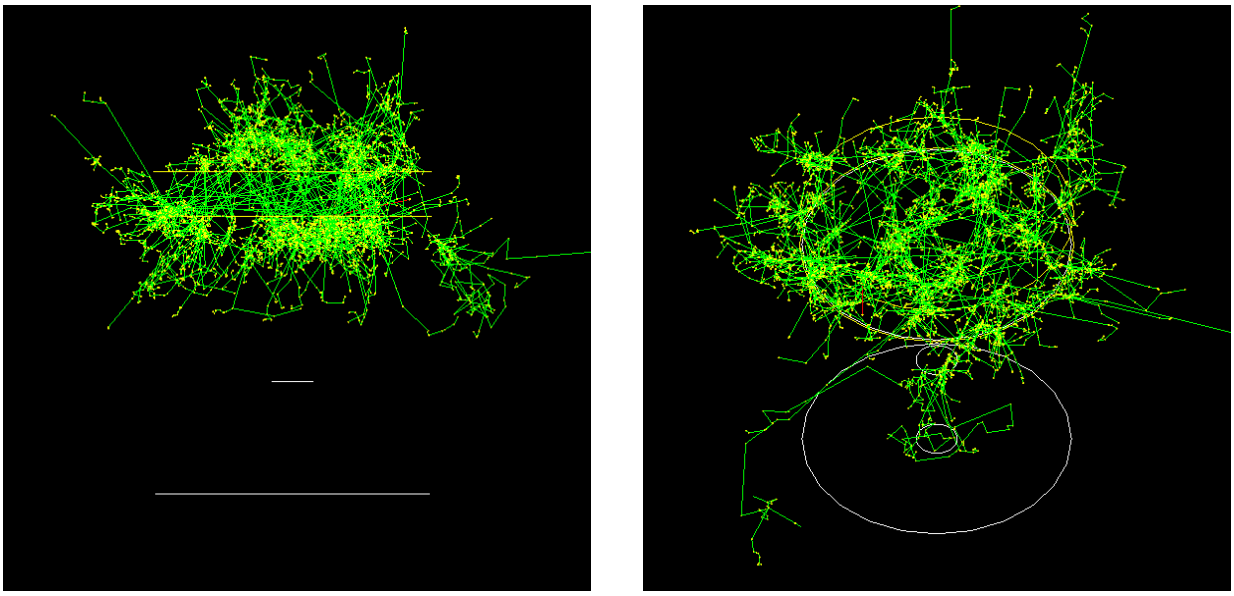


Figure 4.1: Simulation of a coaxial germanium detector with a Marinelli beaker emitting gamma rays in all directions, from all points within the beaker.

The B4 examples all use the `G4ParticleGun` class to define a particle gun that fires a single particle perpendicular to the input face of the calorimeter. Since all the particles (excluding the background) should be coming from the Marinelli beaker containing the sample, a `G4GeneralParicleSource` (GPS) class specified the energy spectrum of the particles and modelled the source position and structure more realistically. To make this change, any references to `G4ParticleGun` had to be replaced with `G4GeneralParicleSource`. The default kinematics of the particle were also changed; attributes that set the energy and direction of the particle momentum were deleted, as these were going to be specified inside a macro file. The simulation was now capable of firing gamma-ray particles at the detector.

The initial plan was to bind `GeneralParticleSource` to a volume that represents the Marinelli beaker to make the simulation as accurate as possible. The volume of the Marinelli beaker was initialised in the same way as the volume of the detector. The provided BUGS code was used as a reference when creating a new method inside `B4aDetectorConstruction`, which was to return the logical volume of the beaker. The method was to be invoked inside `B4aPrimaryGeneratorAction` to bind the position of the particle source to a point on the surface of the beaker volume. However, when attempting to compile the code a few errors were made apparent regarding the method, so a simpler approach was adopted, using macro commands for GPS a simple cylindrical volume with the dimensions of the Marinelli beaker. A particle source was bound to this volume and made to emit in random directions to simulate the gamma-ray emissions from the source. A visualisation of the simulation is seen in Figure 4.1. When running the simulation, particles that did not interact with the detector still registered as an interaction and gave an energy output. To fix this, a new version of B4a was imported and the `B4aDetectorConstruction` and `B4aSteppingAction` files were changed in multiple ways trying to fix the error, though this was unsuccessful.

A different version of the example was therefore used to more accurately simulate and output energy values. In the B4c example, the physics quantities are stored in `B4cCalorHit` via two `B4cCalorimeterSD` objects. One of the objects is related to the volume of the absorber and another to the gap. In `B4cDetectorConstruction`, a simplified version of the detector was modelled without the finger cut out. The calorimeter, as well as the absorber layer and gap layer, were changed from `G4Box` to `G4Tubs` to create a simple cylinder. Due to time constraints and limited C++ knowledge at the time of coding, the absorber was set as a number of layers

extending through out the cylinder, with the gap layers between being negligibly small. This ensured that the code would run and would output the expected energies. A graphical representation of the simulation can be seen in Figure 4.2. The point like emission source was placed 5cm above the top of the detector volume, which was half of the cylindrical part of the Marinelli beaker, excluding the part that surrounds the detector. The GPS was modelled as a point source firing gamma rays in random directions. Energy of the source was set to a Gaussian distribution, with the mean energy and standard deviation equal to those obtained from the experimental data, seen in Figure 4.3. After running the simulation for a certain number of events, a .csv file with all the detected energies was created. This was imported in Python to calculate the measured count rate of gamma rays, which was then used to calculate the absolute efficiency of the detector.

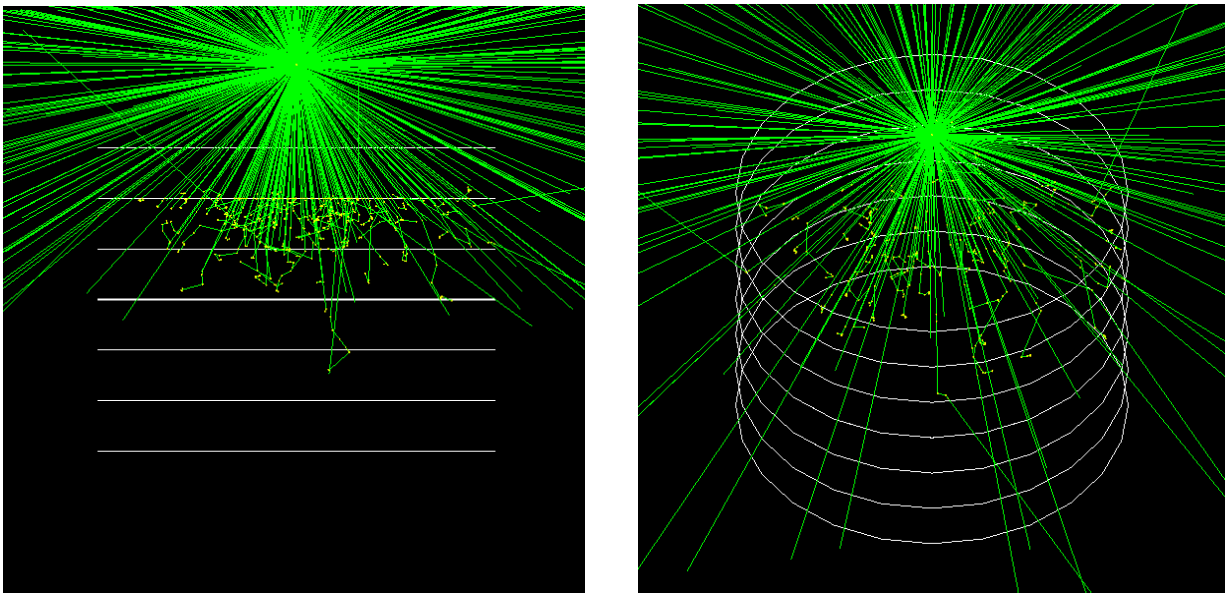


Figure 4.2: Simulation of a cylindrical germanium detector, made up of layers of germanium, with a point-like source emitting gamma rays.

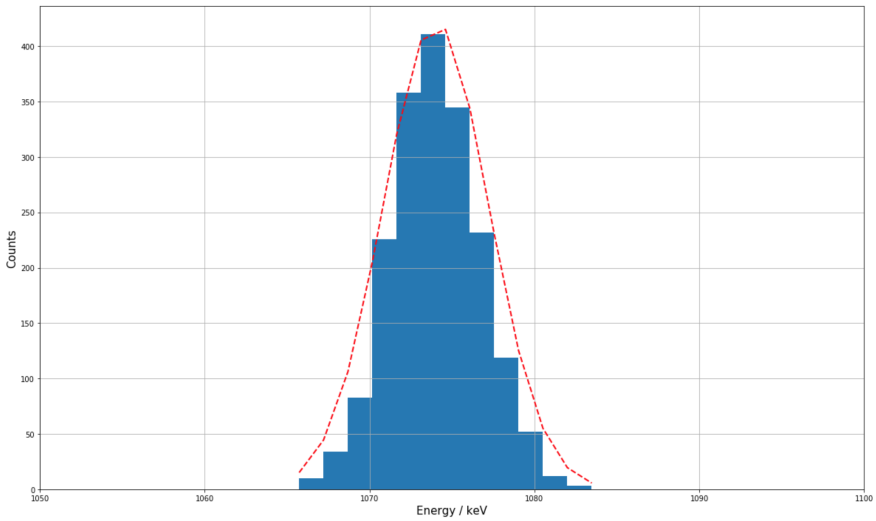


Figure 4.3: Figure showing an example distribution of input energies for the emission simulation, obtained from the experimental results, with a mean energy of 1074.0 keV and a standard deviation of 2.7 keV.

4.3 Simulation Results

The data corresponding to the energy deposited and the path lengths of the gamma rays that interacted with the detector volume were stored. Gamma rays were omnidirectionally emitted at random to simulate the stochastic momenta possibilities of the rays. This allowed determination of the absolute efficiency of the detector (introduced in Equation 1.4) given the detector geometry. The simulation results are graphed in Figure 4.4. At low energies, the absolute efficiency of the detector varies linearly with an increase in the energy of the gamma ray. For energies greater than 75 keV, the absolute efficiency falls rapidly. A least squares polynomial E^{-3} fit was used as the theoretical probability of an interaction between a gamma ray and an atomic electron is given by non-analytical expression in Equation 4.2, where the photoelectric mass attenuation coefficient τ is related to the atomic number Z and the energy E:

$$\tau \propto \frac{Z^n}{E^3} \tag{4.1}$$

where proportionality is approximate, since $n \in [4.0, 4.8]$. The photoelectric mass attenuation coefficient is also directly proportional to the reaction cross section σ:

$$\tau = \frac{N_A \sigma}{A} \tag{4.2}$$

where $N_A \equiv$ Avogadro's constant and $A \equiv$ atomic mass of the absorber atom [23]. The relationship between absolute efficiency and energy were similarly produced in the *SDRP Journal of Earth Sciences & Environmental Studies* [24], where a cylindrical germanium detector similar to that which was modelled for this project was investigated.

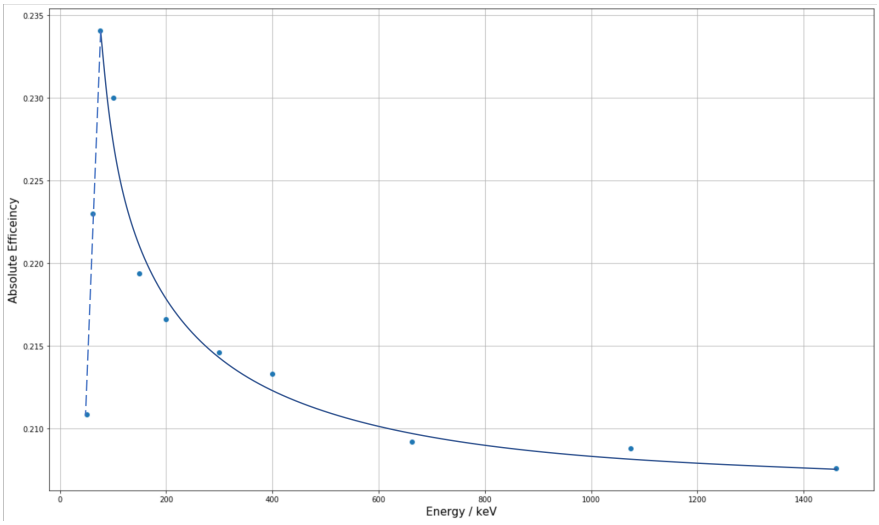


Figure 4.4: Plot showing the absolute efficiency calculations for the Germanium detector interacting with gamma rays produced by a point-like emission source, over various energies.

Chapter 5

Conclusions

5.1 Blind Sample Identification

A high intensity peak at 662 keV is expected in the gamma ray-spectra for ^{137}Cs [17]. Table 3.1 shows that all three blind samples have a peak at 662 ± 2 keV, with confidence level of 11.0σ, 3.9σ and 3.4σ in blind samples 1, 2 and 3 respectively. The confidence level exceeded 3σ for all three blind samples, thus there exists evidence of ^{137}Cs in all three samples, with very strong evidence that it is present in Blind Sample 1. Identification of the three blind samples was attempted by comparing the results of the blind samples to the database. In conclusion, there is strong evidence indicating that Blind 1 was *12MEDIUM*, Blind 2 was *SAMPLEB*, and Blind 3 was *11DRY*. The blind samples were shuffled by the project board member, and the conclusion above has been confirmed as correct by the board member. Therefore, the project was a success; the blind samples were correctly identified using the sample fingerprints stored in the database.

This project serves as a proof of concept for building a database to identify unknown samples which emit gamma peaks. This concept can be further developed, and a larger database could be built to identify the individual elements inside the samples. This could improve nuclear forensics to not just obtain fingerprints of the samples, but to further determine the composition of the sample.

The presence of clearly identifiable levels of ^{137}Cs in all three samples can likely be attributed to the aforementioned Sellafield nuclear site. Though finding a direct comparison to our data is difficult, *Radioactivity in Food and the Environment* [25] includes data which suggests that ^{137}Cs is the radionuclide with by far the highest concentration in sediment samples taken from various locations on the Lancashire coast. Monitoring of levels of ^{137}Cs related to the Sellafield nuclear site is of high importance in ensuring minimal contamination within the food supply and local area.

5.2 Simulation Conclusion

The simulation produced theoretical values of absolute efficiency of our detector. This enabled the calculation of the theoretical absolute efficiency at different photon emission energies. The efficiency fell rapidly as energy increased beyond 75 keV. An analytical approach would require careful analysis of the excitation and relaxation of the germanium electrons, but the Geant framework allows usage of inbuilt physics packages to tackle this.

With more time to develop the project, the Geant4 simulation could have been improved by creating a new `G4RadioactiveDecay` class to simulate the decay of radioactive nuclei rather than using a point like source emitting gamma rays in all directions. With the `G4RadioactiveDecay` class and a more geometrically accurate model of the Marinelli beaker, the simulation could have been used with the ^{60}Co and ^{137}Cs sources to produce a set of energy values to check the detector calibration. One major part of being able to accurately simulate the decay emissions from a sample in a Marinelli beaker is accurately reproducing the chemical

composition the sample. This is a difficult and time consuming process and one that could not be completed within the timeframe of the project.

Furthermore, the properties of the simulated detector could have been improved by including a dead layer. There existed many difficulties in creating the simulation; this was due to limited C++ knowledge, the level of understanding regarding how some classes interact with each other (for example, in more advanced examples that involved particle decay), and the programme written to simulate the detectors in the BUGS experiment. The unmodified examples often generated data that was unnecessary for the investigation, or the physical processes were modelled differently to what was needed for the experiment. These issues had to be adjusted, which often lead to errors caused by the changes themselves. Ideally, with time for Geant4 and C++ skill to be developed, the simulation would have been independently built instead of modifying examples. This would have allowed for the creation of specific classes and methods to model the physics of the detector more accurately.

5.3 Experimental Further Work

Although the exposure time of the data was 30 minutes, it still proved difficult to resolve some peaks from the background, and reliably confirm peaks that were only present in a single measurement. With additional time allocated to the project, it would have been possible to take longer measurements of the samples and the background, giving lower uncertainties in peaks. The new background would also be normalised with respect to time, so the scale of the background spectrum will be increased with the same ratio as the spectrum of the samples. This technique will not be very useful for the peaks that have already been found since the confidence level would not change. However, increasing the total scale would increase the difference between the peaks and the background, resulting in higher confidence levels for peaks with lower intensity. This means more isotopes can be found and therefore more information can be extracted from each energy spectrum.

Another potential development would be to analyse the effect shielding may have on the ability of the detector to identify characteristic gamma ray lines emitted by samples. Many HPGe detectors used in active research, such as that in the BUGS experiment, are located underground and accompanied by large amounts of lead shielding. This acts to vastly decrease levels of background noise, and makes the detection of smaller peaks, as well as peaks that are present in background radiation, significantly easier. Exploring this may allow for a more successful characterisation of the sediment sample emission spectra. A more complete analysis of the radionuclide contents in the sediment samples would allow for a greater understanding of the levels of contamination from human processes such as mining and nuclear energy production. This would enable a thorough assessment of the impact of the presence of higher levels of radionuclides on biodiversity and human health, which could help to identify and reduce the harm from potentially cancer causing radionuclides.

References

[1] Joint Working Group of the American Physical Society and the American Association for the Advancement of Science (2008). *Nuclear Forensics - Role, State of the Art, Program Needs*. [online] *American Physical Society*, AAAS Center for Science Technology and Security Policy 1200 New York Ave NW Washington DC 20005: American Association for the Advancement of Science, pp.12–21. Available at: https://www.aps.org/policy/reports/popa-reports/upload/nuclear-forensics.pdf [Accessed 18 Mar. 2020].

[2] Knoll, G.F. (2010). *Radiation Detection and Measurement*. 4th ed. New York: John Wiley & Sons, pp.10–15.

[3] Gilmore, G.R. (2008). *Practical Gamma-Ray Spectrometry*. 2nd ed. New York: John Wiley & Sons, Ltd., pp.25–26.

[4] Mirion Technologies (2016). *Germanium Detector Overview*. [online] Mirion, p.1. Available at: https://www.mirion.com/assets/germanium-detectors_XJzyVk2.pdf [Accessed 18 Mar. 2020].

[5] Knoll, G.F. (2010). *Radiation Detection and Measurement*. 4th ed. New York: John Wiley & Sons, pp.383–385.

[6] Hughes, I. G., Hase, T.P.A. (2010). *Measurements and their Uncertainties: A Practical Guide to Modern Error Analysis*. New York, NY: Oxford University Press.

[7] Knoll, G.F. (2010). *Radiation Detection and Measurement*. 4th ed. New York: John Wiley & Sons, pp.115–118.

[8] Mirion Technologies (n.d.). *Gamma-Ray Efficiency Calibration*. [online] Mirion. Available at: https://www.mirion.com/learning-center/lab-experiments/gamma-ray-efficiency-calibration-lab-experiment [Accessed 18 Mar. 2020].

[9] Onjefu, S.A., Taole, S.H., Kgabi, N.A., Grant, C. and Antoine, J. (2016). *Assessment of natural radionuclide distribution in shore sedimentsamples collected from the North Dune beach, Henties Bay, Namibia*

[10] Greenpeace. *Sellafield Nuclear Reprocessing Facility*. [Online] Available at: https://web.archive.org/web/20160303235706/http://www.greenpeace.org.uk/nuclear/sellafield-nuclear-reprocessing-facility [Accessed 23 Mar.2020]

[11] CORE, Cumbrians Opposed to a Radioactive Enviroment. *Irish Sea*. [Online] www.corecumbria.co.uk Available at: http://corecumbria.co.uk/alternative-tour-ofsellafield/irish-sea/?doing_wp_cron=1584976030.6103289127349853515625 [Accessed 23 Mar. 2020]

[12] Oak Ridge Associated Universities (2011). *Marinelli Beakers (ca. 1950)*. [online] www.orau.org. Available at: https://www.orau.org/ptp/collection/Miscellaneous/marinelli.htm [Accessed 19 Mar. 2020].

[13] Agostinelli, S. et al. (2003). Geant4 — A Simulation Toolkit. *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3), pp.250-303.

[14] Geant4 Collaboration (n.d.). *High Energy Physics.* [online] Geant4. Available at: https://geant4.web.cern.ch/applications/high_energy_physics [Accessed 19 Mar. 2020].

[15] ORTEC (n.d.). *GEM Series Coaxial HPGe Detector Product Configuration Guide.* [online] ORTEC. Available at: https://www.ortec-online.com/-/media/ametekortec/brochures/gem.pdf [Accessed 25 Mar. 2020]

[16] Upp, D.L., Keyser, R.M. and Twomey, T.R. (2005). New Cooling Methods for HPGe Detectors and Associated Electronics. *Journal of Radioanalytical and Nuclear Chemistry*, 264(1), pp.121–126.

[17] Delacroix, D., P. Guerre, J., Leblanc, P. and Hickman, C. (2002). Radionuclide and Radiation Protection Data Handbook 2002. *Radiation Protection Dosimetry*, 98(1).

[18] *The Ionising Radiations Regulations 2017.* (SCHEDULE 3). [online] London: HMSO. Available at: http://www.legislation.gov.uk/uksi/2017/1075/schedule/3 [Accessed 21 Mar. 2020].

[19] Allison, J. et al. (2016). Recent Developments in Geant4. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 835, pp.186-225.

[20] Fermilab, CERN, DESY and ETHZ. *Scientific Linux 7 Operating System.* [online] Available at: https://https://www.scientificlinux.org/ [Accessed 19 Mar. 2020].

[21] CERN. *CERN CentOS 7 Operating System.* [online] Available at: https://https://http://linux.web.cern.ch/linux/centos7/ [Accessed 19 Mar. 2020].

[22] Scovell, P.R. et al. (2018). Low-Background Gamma Spectroscopy at the Boulby Underground Laboratory. *Astroparticle Physics*, 97, pp.160-173.

[23] Nelson, G. and Reilly, D. (1991). Gamma-ray Interactions with Matter. Passive Nondestructive Analysis of Nuclear Materials, pp.27-42.

[24] Miller, M. (2018) Modelling a HPGe Detector's Absolute Efficiency as a Function of Gamma Energy and Soil Density in Uncontaminated Soil. *SDRP Journal of Earth Sciences & Environmental Studies 3(4)*, pp.485.

[25] Centre for Environment, Fisheries and Aquaculture Science (2012). *Radioactivity in Food and the Environment, 2011.* [online] GOV.UK, p.82. Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/750761/Radioactivity_in_food_and_the_environment_2011_RIFE_17.pdf [Accessed 27 Mar. 2020].

Appendices

Appendix A: Meeting Minutes

Meeting 1

Date: 17/01/20 at 11:30am
Location: Study Room D17
Chair: Kanishk Ganga
Minutes: Kanishk Ganga
Person(s) Absent: None

Agenda:

- Introduction to project
- Construct objectives to meet the aims presented by the project

Log:

- Project applications are related to monitoring of radioactivity in numerous areas, mainly in health, safety and military applications. Key applications are in nuclear trafficking control, nuclear waste, radioactive dating and nuclear disarmament.
- Advantages to gamma ray spectroscopy. Non-destructive method allows for broad application and relatively quick analysis once sample is obtained.
- Understand the mechanisms for germanium and scintillator detectors, their differences and advantages. There exists a potential opportunity to use both in the project, but time constraints must be considered.
- Stages to the project: Learn nuclear forensic theory. Learn gamma ray emission and detection mechanisms. Learn how to analyse the data from the detectors. Run the detectors to collect background and sample data. Calibrate the data using known radioactive sources. Find out what elements contribute to the background spectrum. Identify unknown samples from each other using blind analysis.
- Group organisation is key to success. Split group into subdivisions to tackle workload efficiently. Create Gantt chart to monitor progress and manage internal deadlines.
- Risk assessment: cryogenics, high voltage and radiation is involved. Finalise the document.
- Consider using CERN's GEANT4 (C++) software to create a simulation to help in understanding the efficiency of the detector.

Tasks for Next Meeting:

- Construct Gantt chart
- Divide team into groups
- Find and read on theory for detectors
- Prepare risk assessment (to be finalised by Ruben)
- Read up on GEANT4 and gauge possibility of simulation

Board Member Feedback:
No feedback was given for this meeting following a request for feedback.

Meeting 2

Date: 03/02/20 at 4:00pm
Location: Study Room D17
Chair: John Wong
Minutes: Ira Shokar
Person(s) Absent: None

Agenda:

- Finalising subgroups
- Setting project leaders to ensure internal deadlines are met and overall performance is satisfactory
- Simulation code and detector specifics
- Lab logistics
- Further questions on project/experiment

Log:

- Group is to be split into 2 subgroups; the "lab group" will focus on the experiment and data analysis, and the "theory group" will focus on the background, introduction, simulation and the poster.
- Project leaders were assigned to overlook a specific subtask. They are not to complete the work individually, but to oversee completion of the task and are responsible for the final product.
  - Introduction: Lucas
  - Method: Pip
  - Data analysis: John
  - Conclusion: Mark
  - Overall Report: Kanishk
  - Simulation: Ira
  - Poster: Tiana
  - Lab book: Sarah
- *Ruben:* GEANT4 provides the framework such that you do not need to start from scratch. HEP cluster has access to corresponding documentation. I see this as a stretch goal, with the aim being a comparison between results from experiment and theory. Calibration software is readily available, with a calibration source available to use for calibration. When looking at the calibration, consider the calibration efficiency, which is the ratio of number of gamma detected to number of gamma emitted.
- Lab space is available 10am-1pm every Tuesday, where Ruben can supervise. When Ruben is unavailable, a PhD student will supervise.
- *Ruben:* In principle, 3 hours per week should be sufficient. Ask Bernard if you need more time, as the desk should not be occupied otherwise.
- Discussion about lab availability after reading week; Ruben concludes by making note of this to email Neal Skipper.
- Discussion about the feasibility of studying the scintillator detector. Final decision will be made in next lab session when we better understand the workload.

- Ruben explains the basic of HPGe detection mechanism. There exist two shapes of detector, generally, these are the planar and the coaxial shape. The detector we use is a coaxial p-type detector. Due to an electric field, electrons drift and produce a current proportional to the energy, and this is what is detected. A p-type detector is thicker on one side for a larger surface area, giving greater energy resolution. Cooling the detector is necessary to minimise thermal interactions, which will degrade the energy resolution.
- *Ruben:* Samples will be given once we have demonstrated that we have effectively calibrated the detector. Samples will be labelled. The main task of the project is to see how accurately you can detect the samples given that we are overground and without shielding. Samples are sediments from the Irish Sea.
- All isotopes found in the samples are in a radiation dosage handbook. This can be used to calculate the dosage when working with a radioactive source. Handbook approximates samples as point sources that follow an inverse square law with distance. Values are given per hour.
- *Ruben:* Stretch goal: In research we use this technique for pre-screening for neutrino detection. It is being used in underground detectors to look at the interaction with anti-matter or detection of dark matter. For this, pure materials are need for very high calibration to detect these to do the screening. You need shielding for this, which may arrive during your experiment.
- To see what makes up the sources accurately, we would need to reference a large database to compare signatures, and we do not have access to this database. In essence, we are building a subset of this database with just 3 samples, so this project also serves as a proof-of-concept.

Tasks for Next Meeting:

- Begin coding preparation for the GEANT4 simulation
- Reach a decision on whether the scintillator detector will be included in our investigation
- Calibrate the detector in next lab session
- Complete mid-term review

Board Member Feedback:
"Good to see the group has organised itself and distributed tasks between individual members. I support the idea of having people responsible for individual tasks, which does not mean these will be the people necessarily writing this particular section of the report, preparing the poster etc.

The project offers a multitude of different developments (two HPGe detectors, scintillator detector, Geant-4 based simulations etc). You may not be able to do all of them. In fact part of the task is to identify realistic priorities and come up with criteria of task prioritisation and perhaps select those you can realistically deliver. Therefore on the project management side you should think about project deliverables, the decision points, the critical path etc. You also need to identify review points at which you can assess the progress and perhaps drop some tasks and pursue others. In summary, I would like to see the group identifying essential tasks, and the tasks in the "it would be good" category."

Meeting 3

Date: 24/02/20 at 4:00pm
Location: Study Room D17
Chair: Pip Benjamin
Minutes: Lukas Kubin
Person(s) Absent: None

Agenda:

- Progress on the simulation code
- Update on calibration code and data analysis
- Plan for upcoming lab sessions; scanning times, efficiency and second detector
- Update on report write-up

Log:

- Geometry of the detector should be available in the folders by the experiment in the lab; this can be used to model the detector.
- *Kanishk:* Gaussians have been fit to each peak, but we want to write something that will automatically find peaks in the spectrum. Python is being used for data analysis.
- *Ruben:* I suspect Python has libraries with efficient code to find peaks in a spectrum. Look into this.
- Lab sessions: calibration values found using the experiment's software, but we will be using python to ensure we can replicate the results. We are ready to scan the samples. Previous scans were 10 minute exposures. From these exposures, we believe a potassium peak exists, but Ruben strongly suggests we give additional statistical confidence to our argument.
- *Ruben:* I will bring the samples in [next lab session]. If you have a background spectrum you can estimate how much data you need. You can do some research and figure out how much data you need so you can save some time. With your calibration you can already identify potassium. It would be nice if you could identify other peaks coming from the environment. It would be nice for your report to have a background spectrum of UCL Lab 3.
- On the topic of detector efficiency; we do not know the activity of the sample to be able to determine efficiency. With data analysis of the spectra, we can determine the energy resolution of each peak.
- Second detector requires calibration; the PhD student will be assisting us. We have decided against using the scintillator detector, in favour of instead attempting the simulation and using the second detector.
- First draft of the report is to be done in 2 weeks from now. Kanishk has started introduction and background sections already. Pip has started the experimental method section.
- Details on the poster — It should be a visual, not a condensed version of the text; minimise words, and instead have bullet points and plenty of graphs and images. At a conference, the intention is to summarise the main results and attract interest to read the report.
- Kanishk will trade some data analysis work for time on the poster.

Tasks for Next Meeting:

- Mark: Recording data from the first detector and continue calibration calculations.
- Ira: Working with Lukas to get the simulation running and assist data analysis.
- Kanishk: Finish portion of data analysis and continue report writing for introduction. Start poster work.
- Tiana: Start work on the second detector, taking sample readings.
- Sarah: Working with Tiana on the detector and continuing lab book write-up.
- Pip: Data collection from first detector and reading up on the simulation.
- Lukas: Continue work on the simulation and look intro the introduction section on the report.
- John: Continue data analysis and look into the possibility of making a database to identify what elements correspond to the peaks.

Board Member Feedback:
"The project proceeds with a good, steady pace. The organisation and distribution of tasks seem to be working well. I see no issues with completing all measurements in the lab on time but I would suggest the group reviews whether the analysis of existing data and calibrations are lagging slightly behind. It should be noted that the members of the group can (and possibly should) get together outside of lab and project meeting hours. This may help with improving communication, as noted by several members in their mid-term reviews, and also help catching up with any analysis and calibration tasks. The group may want to consider using visual aids during project meetings. A projector is available in D17 for that purpose."

Meeting 4

Date: 09/03/20 at 4:00pm
Location: Study Room D17
Chair: Ira Shokar
Minutes: Sarah Gao
Person(s) Absent: Yichen Tian (illness)

Agenda:

- Update on data analysis; comparison of times, resolution, notable peaks
- Update on GEANT4 simulation
- Progress on report
- plan for upcoming lab sessions; scanning times and data analysis
- poster discussion; design ideas

Log:

- *John:* Comparing measurements on March 3rd and 20th February, we see that with the three samples, there exists a peak at 662keV, corresponding to Caesium (Cs). A Gaussian has been drawn on this.
- In the samples, only one of them shows a large peak. The other two shows signs of the peak, but the sigma confidence is unclear. We may have discovered something unexpected. John goes through the spectrum by inspection and selects possible peaks to analyse. Most of them are not statistically significant apart for the Cs peak found. We are unsure of a potassium peak in the sample due to the high potassium content of the background.
- We need to quantify the significance of the Cs peak. 3σ indicates evidence, and 5σ indicates a discovery. More Gaussian peaks will be plotted. The second detector will help identify if the Caesium peak is truly present in the sample.
- Right now we have a small number of peaks to determine the full curve for the resolution. Very generally we found that resolution decreases with increasing energy. Must consider error bars. We also need to consider fitting a distribution to try and represent the physical model of resolution.
- Simulation code: segmentation error is occurring. we discussed with Adam (PhD student) who was also unsure of its origin. We are close to simulating a decay.
- Report: Introduction is making progress. Executive summary is a detailed summary of the report, so we require that everything important is in there, taking up about a page. For the paper, we should consider using diagrams and captions to reduce lengthy explanations.
- *Kanishk:* I found a very good paper from the University of Washington that covers very similar ideas to what we are researching.
- *Ira:* I propose a deadline of 2 weeks from today for the report. From then onward, we can meet up and read through it to make any changes.
- Next lab session: We will take longer sample times of 30 minutes on both detectors. We have collected data from the second detector, but the data is in a different format so we need to to the first detector, so we need to adjust the data accordingly. This is due to the ADC and energy scale being different. outputting the data correctly should be straightforward using C++. Calibration for this detector must also be done.

Tasks for Next Meeting:

- Pip will take data from the first detector and assist with data analysis on the second detector, and will write down a procedure.
- Kanishk will work on the introduction primarily, and assist with getting the second detector data into the correct format.
- Mark will continue data analysis and calibration of the second detector, and will look into what we need for the conclusion.
- Sarah will continue work on the second detector and keep record in the lab book.
- Ira will continue development of the simulation and help write the corresponding section in the report.
- Lukas will continue development of the simulation and help write the corresponding section in the report.
- John will complete data analysis and ideally will be able to identify the blind samples in the next lab session.

Board Member Feedback:
"Good progress has been made in analysing data and producing an outline of the final report. The dynamics of the group is good and communications have improved. It was especially good to see that tasks are distributed in a fair, transparent and collaborative way. Using data projector for the progress report was very helpful and allowed the group to run the meeting more efficiently.

At this point I think the data analysis may become the critical path and should be watched by the group to see whether any additional manpower should be diverted there. In general, introducing elements of a more professional project management would help the group to deliver main goals on time: preparing a Gantt chart with tasks breakdown could have identified the critical path and ways to mitigate delays and ensure the status of tasks and manpower are in sync with the project schedule."

Meeting 5

Date: 17/03/20 at 10:00am
Location: Online meeting via *Zoom*
Chair: Kanishk Ganga
Minutes: Yichen Tian
Person(s) Absent: Ke Ma (Abroad, connection blocked in China)

Agenda:

- Update on GEANT4 simulation, problem of modelling a perfect detector
- Update on data analysis; energy resolution and overall results for 1st detector done; results for distinguishing blind samples done.
- Progress on second detector
- Report progress
- Poster discussion; overall structure

Log:

- *Ira:* Problem can be fixed by altering the voltage supply to about $100MeV$.
- *John:* I plotted the energy resolution graph spectrum for the first detector, using the formula scipy.optimize.minimize function.
- For a good fit, $\bar{\chi}^2$ should be 1, but ours is less than 0.5, meaning it's overestimating error. Our board member pointed out that we should come up with an analytical solution, which is $R = \Delta E/E = \sqrt{\alpha^2 + \beta^2/E}$. Meaning resolution should change as a result of poisson distributed counts ($\sqrt{\text{mean}}$). As energy linearly increases, $\Delta E/E$ should change as $\sqrt{E}$.
- σ is calculated by (background peak + data peak)/total uncertainty. The Poisson distributed uncertainty is given by $\sqrt{\text{background+data}}$.
- *John:* horizontal axis unit is keV; vertical axis unit is in percent, a sensible number for resolution is around 1 or 2 when energy is around $600keV$.
- *Tiana:* progress on second detector: calibration and all data taking done; needs to reformulate code for energy resolution and overall results.
- *John:* The preliminary conclusion we have for nuclear forensics is

blind sample 1	12 Medium
blind sample 2	Sample B
blind sample 3	11 Dry

- *Kanishk:* Progress on report; mostly finished before data analysis; plan for next week: finishing introduction and methodology; writing up analysis, simulation summary for last 2 weeks, conclusion and appendices.
- *John:* I changed the code to correct y-axis values. I will also add the physics behind each term in the resolution analytical formula.
- *Kanishk:* Update on poster: overall structure done, should add a few screenshots of final spectra and simulation; also a list of bullet points of report.

- *Board member:* Feedback for simulation: Concept is to reproduce spectra. We can possibly estimate efficiency of point like sources used for calibration (Co and Cs). To model the efficiency of our detector, we can mimic imperfections in detector by using data as input to simulations. We know energy resolution as function of energy, so every time a gamma deposit an energy inside detector, randomise it from a sample with a σ corresponding to that resolution.

Tasks for Next Week:

- *John:* Finish the graphs,trim code for later process, output graphs to poster.
- *Kanishk:* Continue with poster design, write up introduction in final report.
- *Tiana:* Reformulate code for second detector, responsible for bibliography in final report.
- *Pip:* Finish final report experiment part, help with results for nuclear forensics (identifying blind samples).
- *Mark:* Write up conclusion for final report and help with data analysis.
- *Lukas:* Complete simulation code with Ira, input data into simulation, help with simulation part of report.
- *Ira:* Write up simulation part of final report and help with any slacks in other parts (maybe conclusion).
- *Sarah:* Help Tiana with second detector and communicate with Mark.

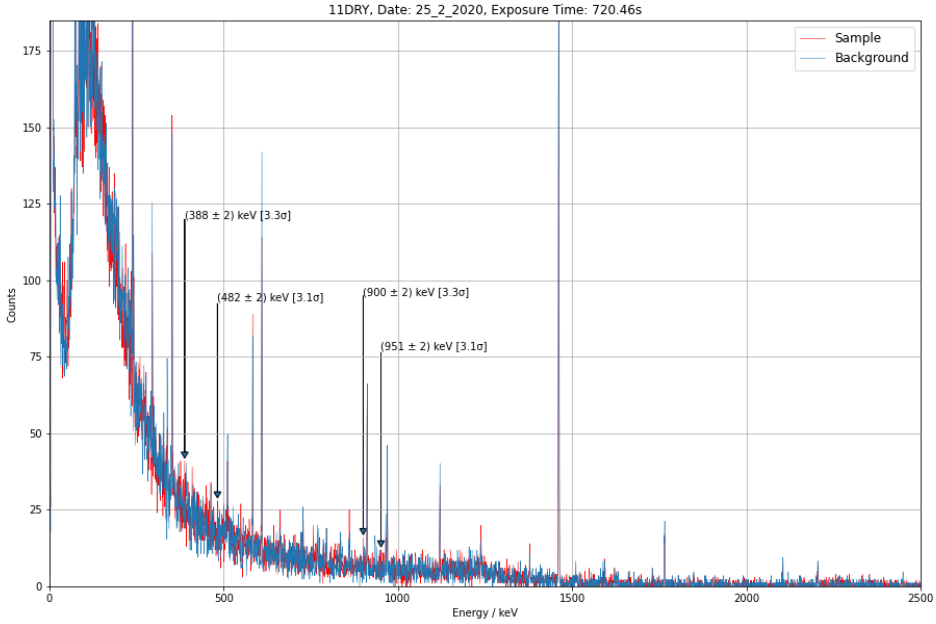
Board Member Feedback:
No feedback was given for this meeting following a request for feedcback.

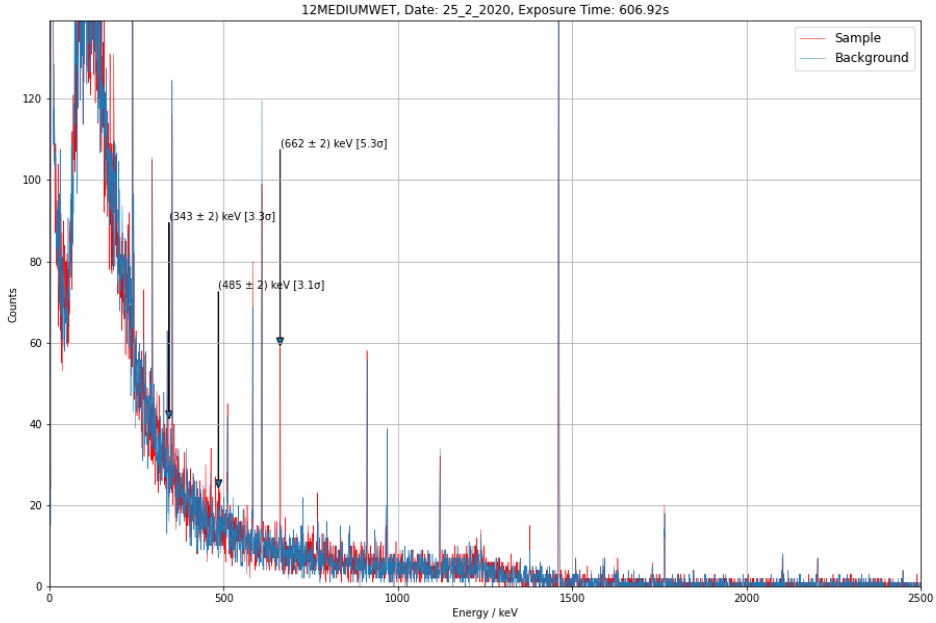
Appendix B: Finances

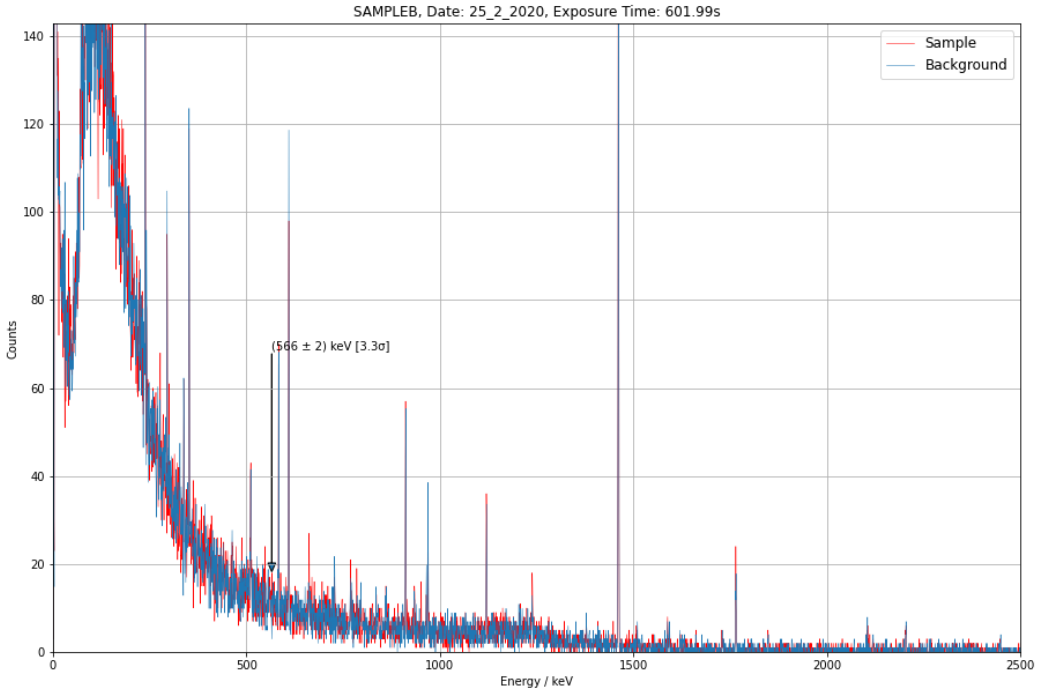
No additional finances were needed to complete the investigation.

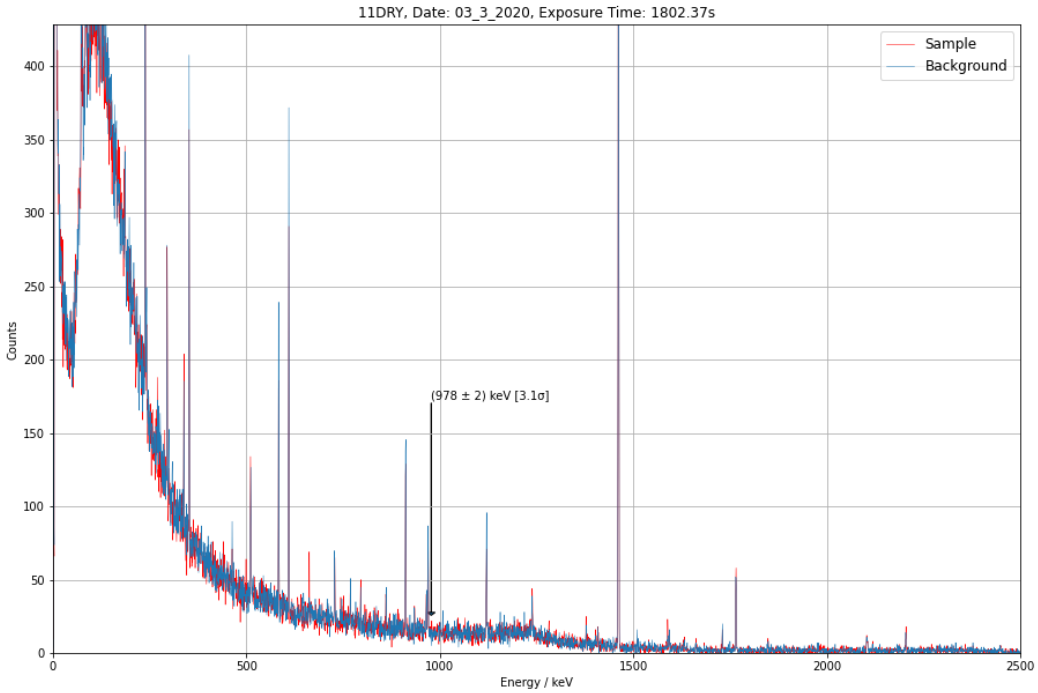
Appendix C: Detector Graphs

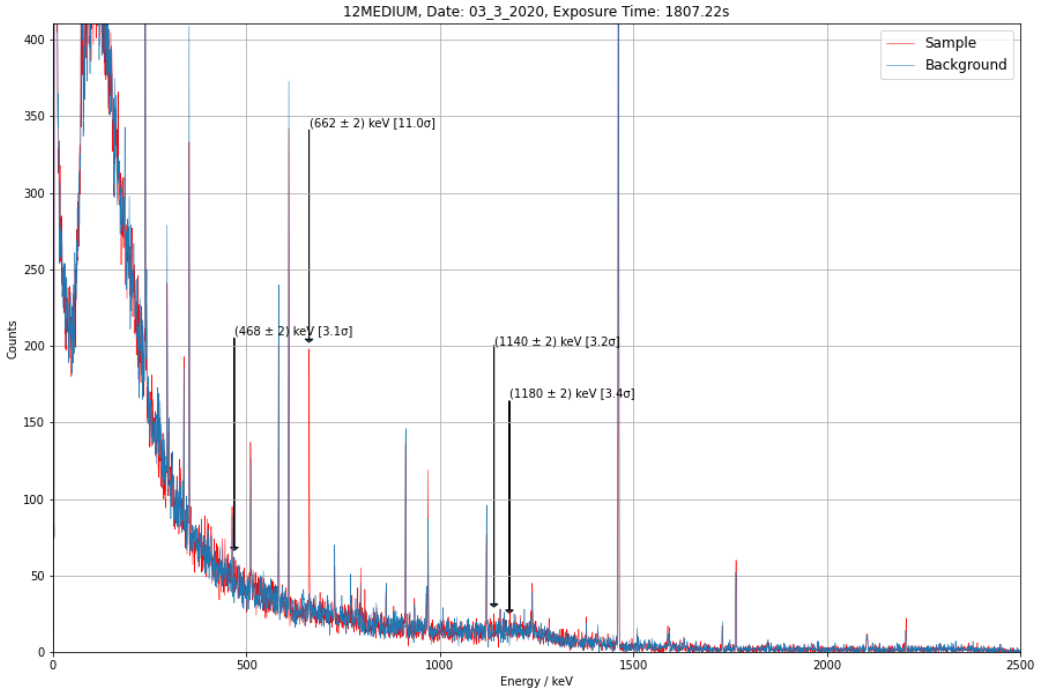
Mirion Detector Graphs

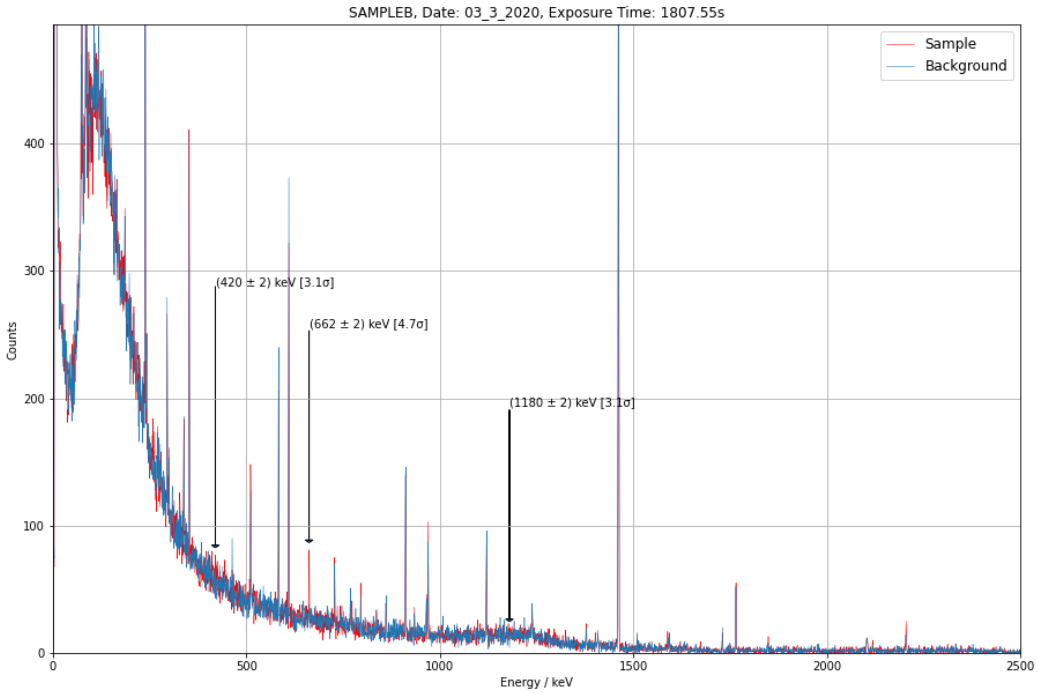


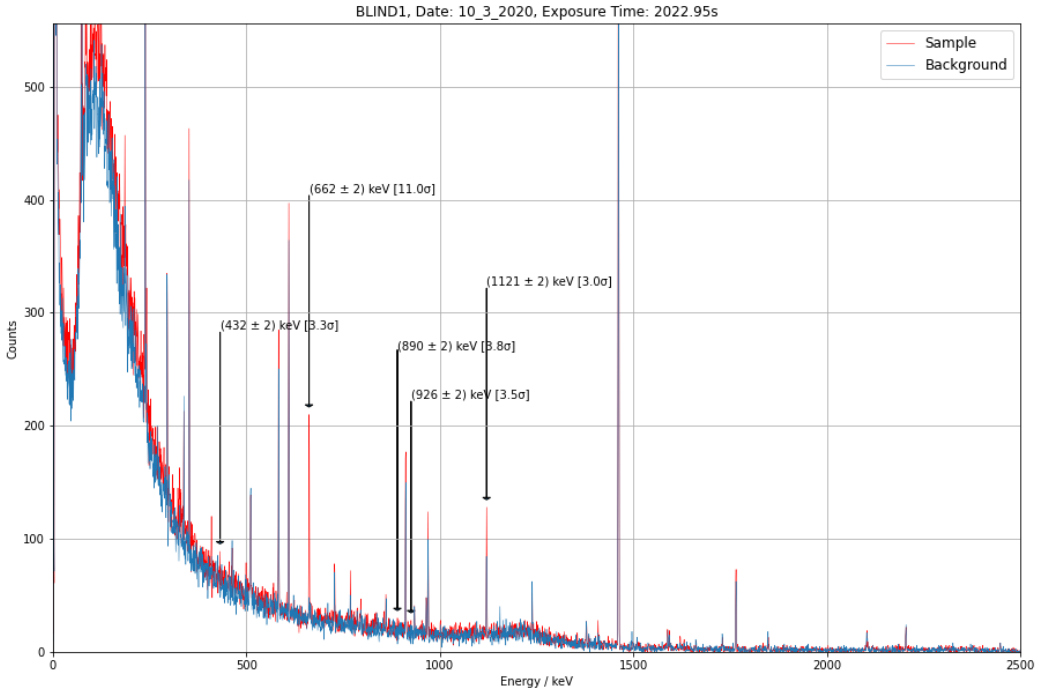


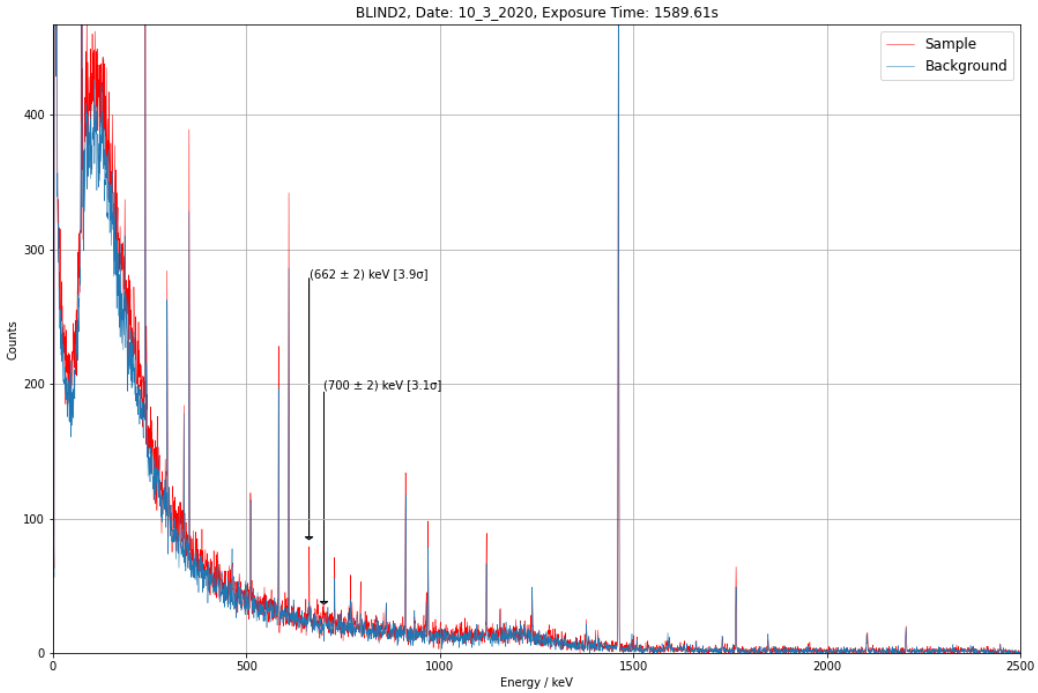

SAMPLEB, Date: 25_2_2020, Exposure Time: 601.99s
Sample
Background
(566 ± 2) keV [3.3σ]
Counts
Energy / keV

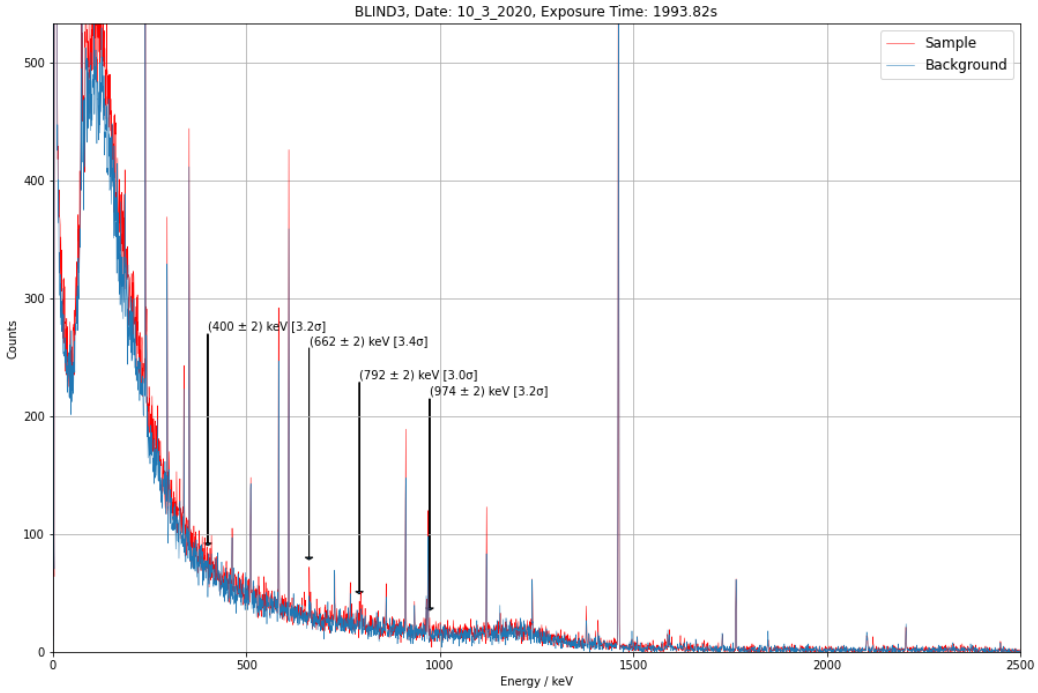

11DRY, Date: 03_3_2020, Exposure Time: 1802.37s
Sample
Background
(978 ± 2) keV [3.1σ]
Counts
Energy / keV


12MEDIUM, Date: 03_3_2020, Exposure Time: 1807.22s
Sample
Background
(662 ± 2) keV [11.0σ]
(468 ± 2) keV [3.1σ]
(1140 ± 2) keV [3.2σ]
(1180 ± 2) keV [3.4σ]
Counts
Energy / keV

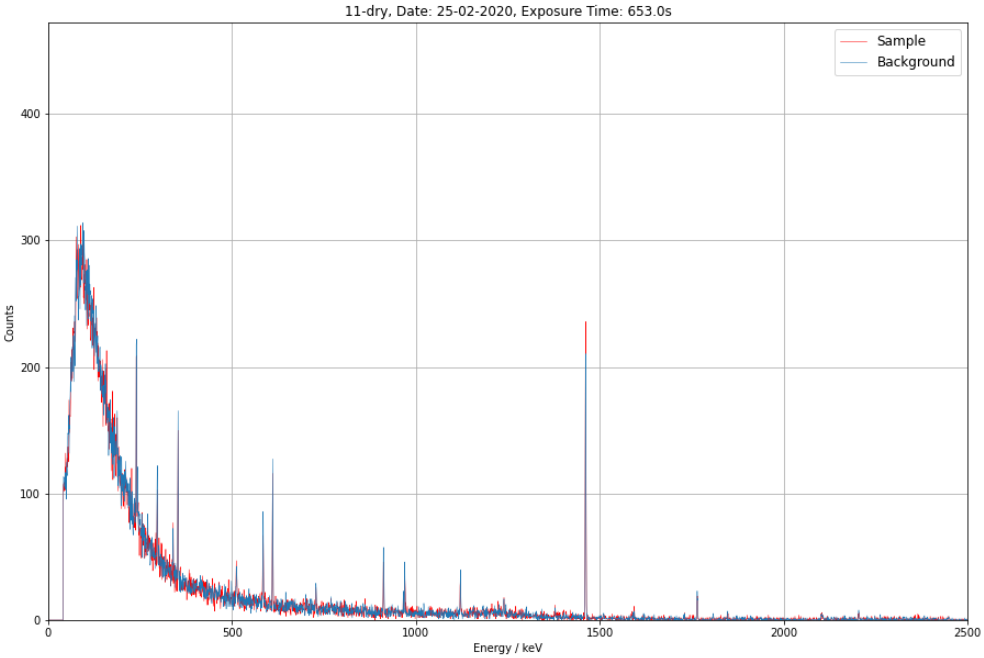

SAMPLEB, Date: 03_3_2020, Exposure Time: 1807.55s
Sample
Background
(420 ± 2) keV [3.1σ]
(662 ± 2) keV [4.7σ]
(1180 ± 2) keV [3.1σ]
Counts
Energy / keV

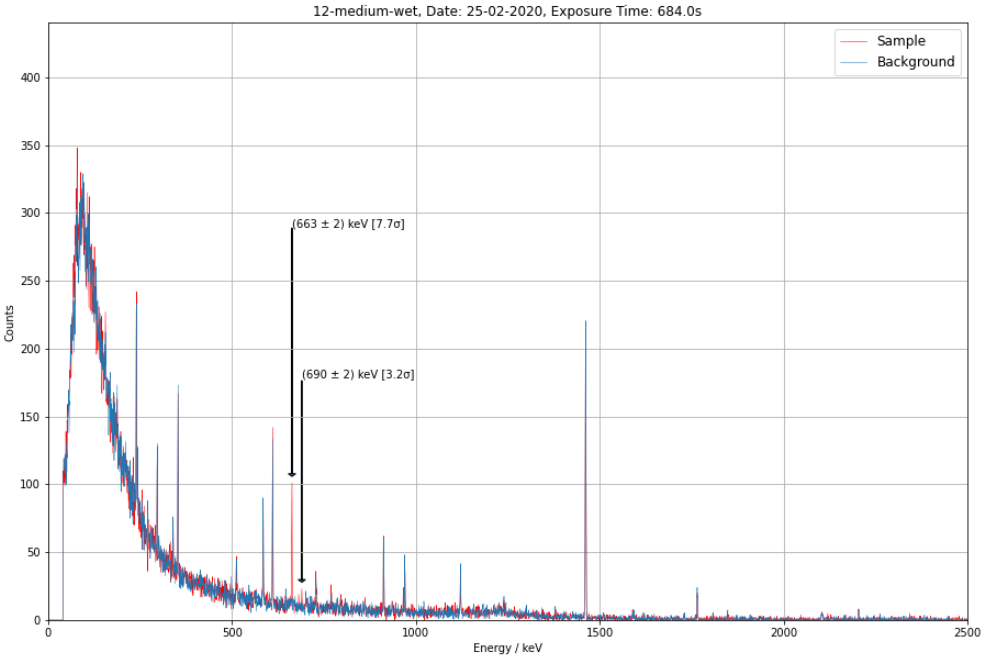

BLIND1, Date: 10_3_2020, Exposure Time: 2022.95s
Sample
Background
(662 ± 2) keV [11.0σ]
(1121 ± 2) keV [3.0σ]
(432 ± 2) keV [3.3σ]
(890 ± 2) keV [3.8σ]
(926 ± 2) keV [3.5σ]
Counts
Energy / keV

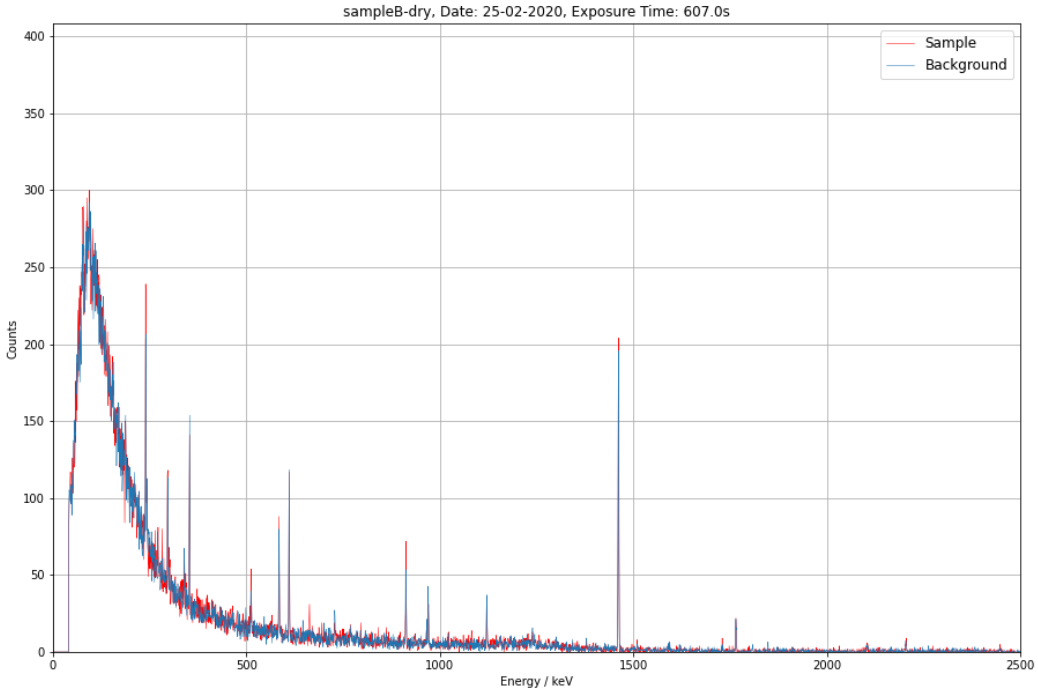

BLIND2, Date: 10_3_2020, Exposure Time: 1589.61s
Sample
Background
(662 ± 2) keV [3.9σ]
(700 ± 2) keV [3.1σ]
Counts
Energy / keV

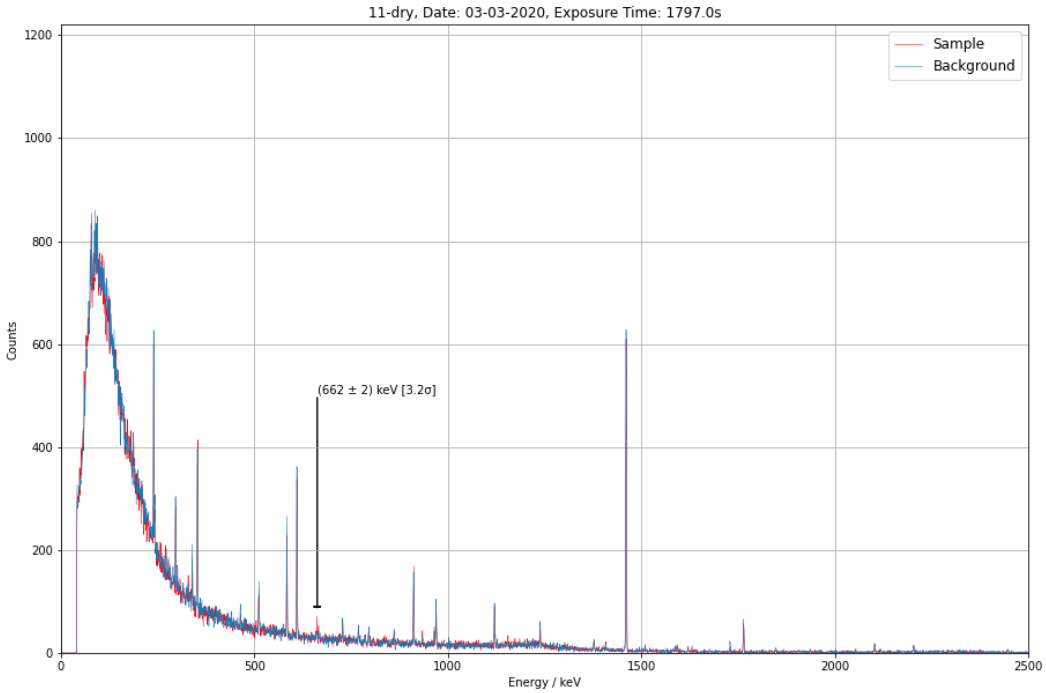

BLIND3, Date: 10_3_2020, Exposure Time: 1993.82s
Sample
Background
(400 ± 2) keV [3.2σ]
(662 ± 2) keV [3.4σ]
(792 ± 2) keV [3.0σ]
(974 ± 2) keV [3.2σ]
Counts
Energy / keV
0
100
200
300
400
500
0
500
1000
1500
2000
2500

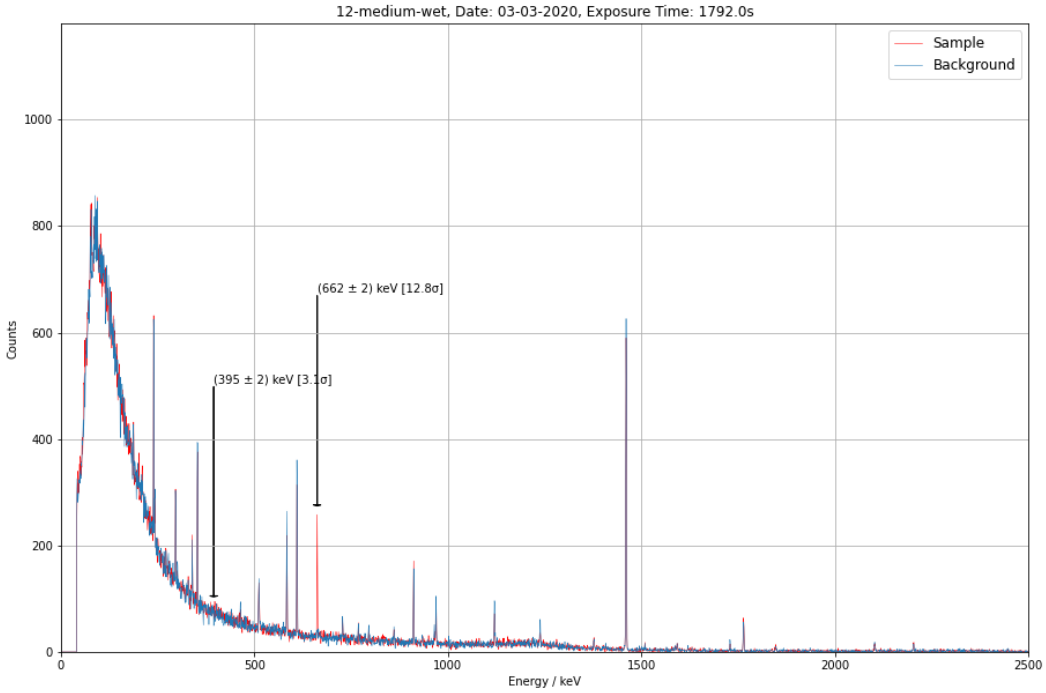
ORTEC Detector Graphs

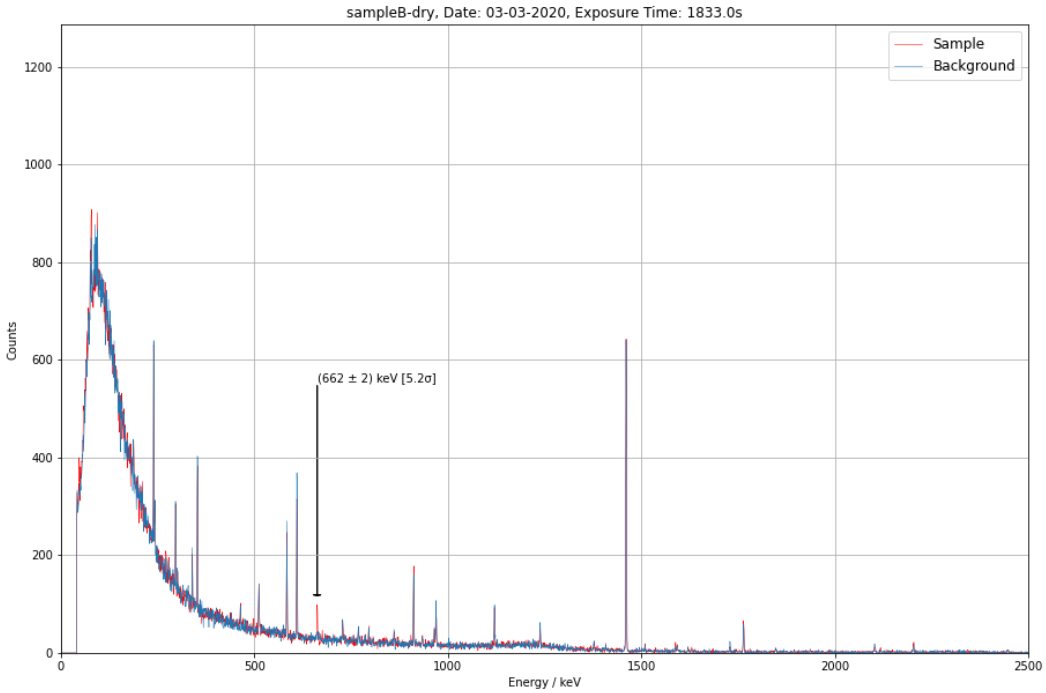


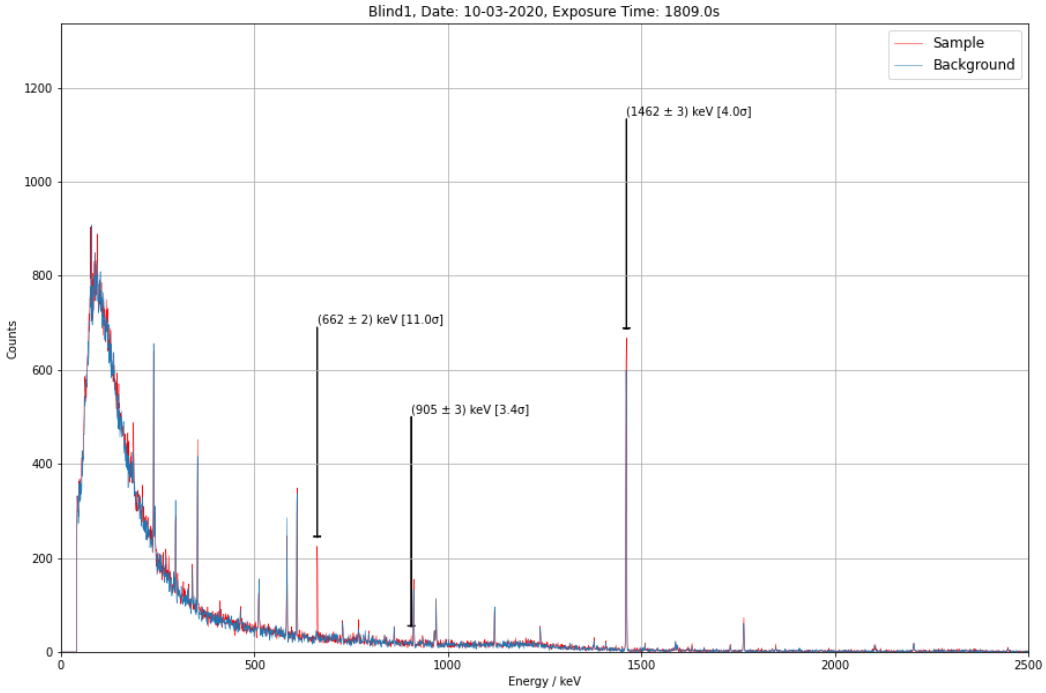


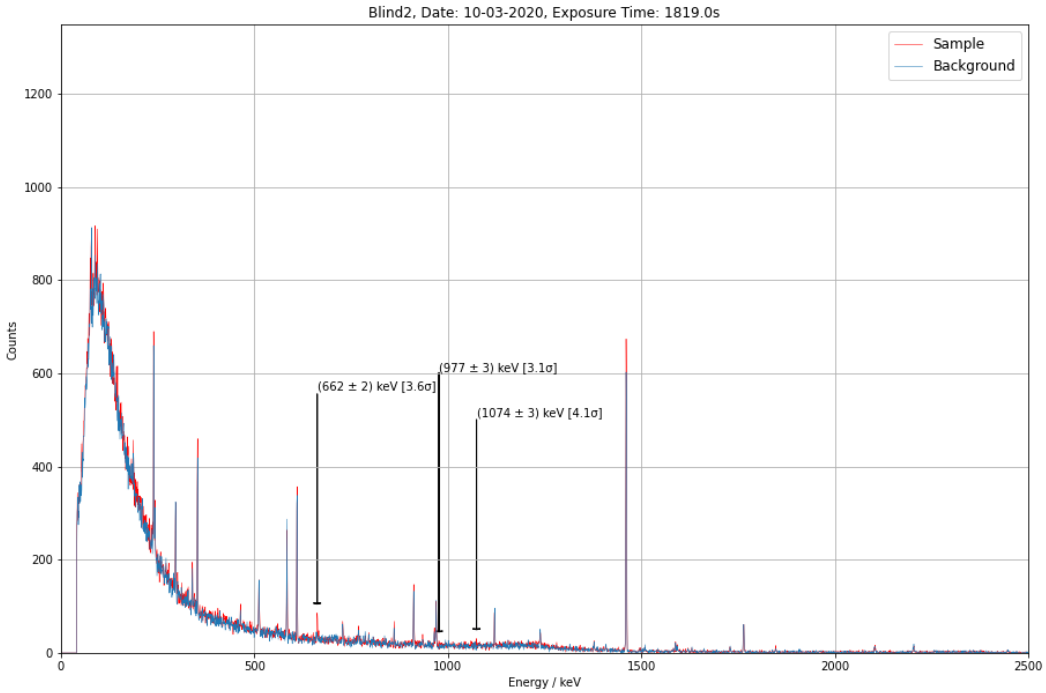

sampleB-dry, Date: 25-02-2020, Exposure Time: 607.0s
Sample
Background
Counts
Energy / keV
0
50
100
150
200
250
300
350
400
500
1000
1500
2000
2500

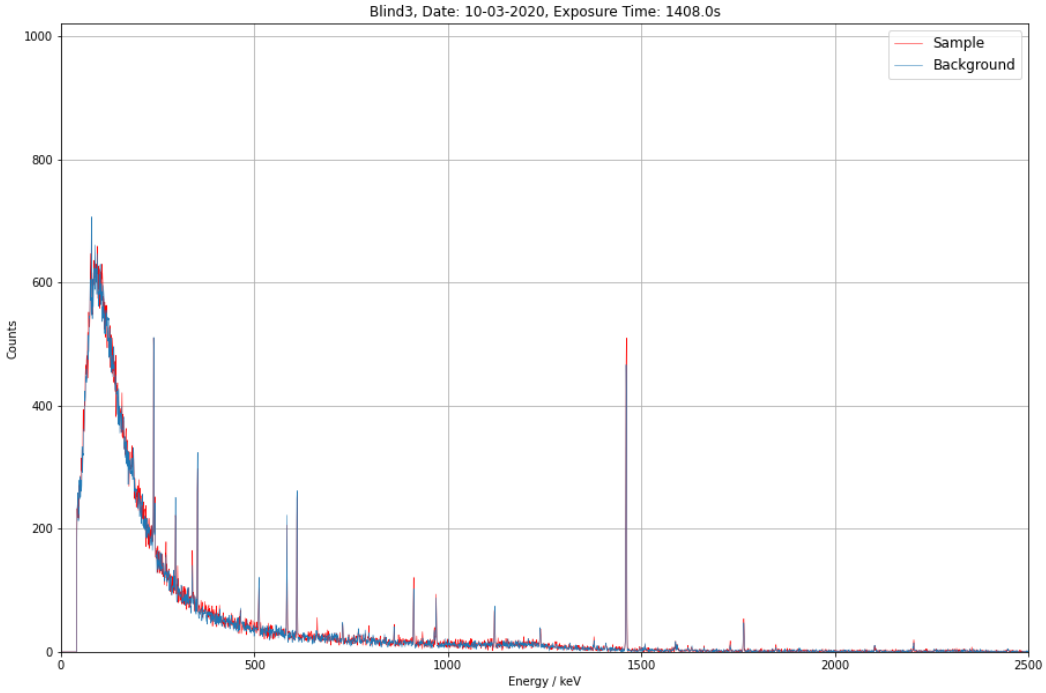

11-dry, Date: 03-03-2020, Exposure Time: 1797.0s
Sample
Background
(662 ± 2) keV [3.2σ]
Counts
Energy / keV
0
200
400
600
800
1000
1200
500
1000
1500
2000
2500


12-medium-wet, Date: 03-03-2020, Exposure Time: 1792.0s
Sample
Background
(662 ± 2) keV [12.8σ]
(395 ± 2) keV [3.1σ]
Counts
Energy / keV
0
200
400
600
800
1000
0
500
1000
1500
2000
2500


sampleB-dry, Date: 03-03-2020, Exposure Time: 1833.0s
Sample
Background
(662 ± 2) keV [5.2σ]
Counts
Energy / keV
0
200
400
600
800
1000
1200
0
500
1000
1500
2000
2500


Blind1, Date: 10-03-2020, Exposure Time: 1809.0s
Sample
Background
(1462 ± 3) keV [4.0σ]
(662 ± 2) keV [11.0σ]
(905 ± 3) keV [3.4σ]
Counts
Energy / keV
0
200
400
600
800
1000
1200
0
500
1000
1500
2000
2500


Blind2, Date: 10-03-2020, Exposure Time: 1819.0s
Sample
Background
(977 ± 3) keV [3.1σ]
(662 ± 2) keV [3.6σ]
(1074 ± 3) keV [4.1σ]
Counts
Energy / keV
0
200
400
600
800
1000
1200
0
500
1000
1500
2000
2500


Blind3, Date: 10-03-2020, Exposure Time: 1408.0s
Sample
Background
Counts
Energy / keV
0
200
400
600
800
1000
0
500
1000
1500
2000
2500

Appendix D: Python Code

Code for Detector Resolution

```
###################
# Python modules #
###################
%matplotlib inline

import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt

import scipy
from scipy.optimize import curve_fit
from scipy.special import factorial

from google.colab import files
from google.colab import drive
drive.mount('/content/drive/')

#########
# Files #
#########
# 25-2 #
#Background = '2nd detector/in csv. format/25-02-2020/BG-30min.csv'
#file = '2nd detector/in csv. format/25-02-2020/11-dry-30min.csv'
#file = '2nd detector/in csv. format/25-02-2020/12-medium-wet-30min.csv'
#file = '2nd detector/in csv. format/25-02-2020/sampleB-dry-30min.csv'

# 3-3 #
#Background = '2nd detector/in csv. format/03-03-2020/BG-30min.csv'
#file = '2nd detector/in csv. format/03-03-2020/11-dry-30min.csv'
#file = '2nd detector/in csv. format/03-03-2020/12-medium-wet-30min.csv'
#file = '2nd detector/in csv. format/03-03-2020/sampleB-dry-30min.csv'

# 10-3 #
Background = '2nd␣detector/in␣csv.␣format/10-03-2020/BG-30min.csv'
#file = '2nd detector/in csv. format/10-03-2020/Blind1-30min.csv'
#file = '2nd detector/in csv. format/10-03-2020/Blind2-30min.csv'
file = '2nd␣detector/in␣csv.␣format/10-03-2020/Blind3-30min.csv'

# Ensuring anyone can read the file from their local machine
directories = ['/content/drive/My␣Drive/Uni/PHAS52␣Group␣Project/CSV_Data/', # Ira path
              '/content/drive/My␣Drive/UCL/Year␣3/PHAS52␣Group␣Project/CSV_Data/', #
                  ↪ Kanishk path
              '/content/drive/My␣Drive/PHAS52␣Group␣Project/CSV_Data/' # Pip/John path
              ]
# Background #
for i in (directories):
  try:
    db = pd.read_csv(i+Background)
  except (FileNotFoundError):
```

```
    pass
# Data #
for i in (directories):
  try:
    df = pd.read_csv(i+file)
  except (FileNotFoundError):
    pass

################
# Sample data #
################
# Background #
back = db[11:-14].rename(columns=db.iloc[10])
# converting to floats
back_counts = [float(i) for i in back['0 16383']]
back_energy = np.arange(1,len(back_counts)+1,1)
# Data #
data = df[11:-14].rename(columns=df.iloc[10])
# converting to floats
data_counts = [float(i) for i in data['0 16383']]
data_energy = np.arange(1,len(data_counts)+1,1)

# Background reduction #
read_b_time = db[8:9].rename(columns=db.iloc[7])
b_time = np.array([float(i) for i in read_b_time['$MEAS_TIM:']])
read_d_time = df[8:9].rename(columns=df.iloc[7])
d_time = np.array([float(i) for i in read_d_time['$MEAS_TIM:']])
back_cal = d_time/b_time
# Array of the calibrated background counts
cal_back_counts = []
for x in back_counts:
    cal = x * back_cal
    cal_back_counts.append(float(cal))
# Array of the calibrated data counts
cal_back_counts = []
for x in back_counts:
    cal = x * back_cal
    cal_back_counts.append(float(cal))

# Calibration #
# Calibration parameter
scale = 1
m = 0.78438913
c = 0.98185618
# Array of calibrated energy
cal_energy = []
for x in data_energy:
   cal = m*(x*scale) + c
   cal_energy.append(cal)

# plotting data
plt.figure(figsize=(15,10))
plt.plot(cal_energy, data_counts,'r', linewidth = 0.5, label = 'Data')
plt.plot(cal_energy, cal_back_counts, linewidth = 0.5, label = 'Background')
```

```
plt.xlim(0,2500) # max = 2776keV
plt.ylim(0,450)

# Graph setting #
#plt.title('SAMPLEB (3rd of March)')
#plt.ylabel('Counts / %')
#plt.xlabel('Energy / keV')
plt.legend(loc='best',fontsize=12)
#plt.grid()

high_back = 381 # 381th position = 300keV
reduced_energy = data_energy[high_back:]
reduced_data_counts = data_counts[high_back:]
reduced_back_counts = back_counts[high_back:]

from scipy.signal import find_peaks
spikes, _ = find_peaks(data_counts, height=100,distance=10) # Data (clipped)

spikes = [1869]

# Finding peaks in calibrated energy
peaks = []
for i in spikes:
   fill = m*(data_energy[i]) + c
   int_fill = int(fill)
   peaks.append(fill)
   int_peaks.append(int_fill)

def gaussian(x, norm, mean, sd):
    """Equation for a Gaussian function
    x: variable
    norm: height of the Gaussian
    mean: position of the center of the peak
    sd: standard deviation
    """
    f = norm * np.exp(-((x - mean)**2/(2*sd**2)))

    return f

# fitting a Gaussian #
E = []
dE = []
Res = []
dRes = []

n_peak = 0
while len(spikes) > n_peak:

########
# Data #
########
    # filter out peak sections #
    res = 1
```

```
range_start, range_end = spikes[n_peak] - 2*res, spikes[n_peak] + 2*res
# slicing the data
energy_in_range = reduced_energy[range_start:range_end]
counts_in_range = reduced_data_counts[range_start:range_end]
back_counts_in_range = reduced_back_counts[range_start:range_end]
# converting the sliced data to array
x = np.array(energy_in_range)
y = np.array(counts_in_range)
b = np.array(back_counts_in_range)
# sum of the counts
N = sum(y)
b_N = sum(b)

# converting data to 1d array #
# Data #
counter = 0
xi = []
while len(y) > counter:
    fill = np.ones(int(y[counter])) * x[counter]
    xi.extend(fill)
    counter = counter + 1
# Background #
b_counter = 0
b_xi = []
while len(b) > b_counter:
    fill = np.ones(int(b[b_counter])) * x[b_counter]
    b_xi.extend(fill)
    b_counter = b_counter + 1

#############
# Data fit #
#############
# initial guesses
mean, sd = stats.norm.fit(xi)
norm = (1/(sd * np.sqrt(2*np.pi)))
guess = [norm, mean, sd]
#optimisation
popt, pcov = curve_fit(gaussian, x, y, p0=guess)
perr = np.sqrt(np.diag(pcov))
# generated values for Gaussian line
fit_x = np.linspace(min(x),max(x),10000)
fit_y = gaussian(fit_x,*popt)

# resolution analysis #
# Value #
FWHM = 2*np.sqrt(2*np.log(2)) * popt[2]
E0 = m*(popt[1]*scale) + c
R = FWHM / E0 * 100
# Uncertainty #
FWHM_err = perr[2]/popt[2] * FWHM
E0_err = perr[1]/popt[1] * E0
R_err = np.sqrt((FWHM_err/FWHM)**2 + (E0_err/E0)**2) * np.absolute(R)

# calibration #
```

```
    cal_x = [] # Data x
    for i in x:
      cal = m*(i*scale) + c
      cal_x.append(cal)

    cal_fit_x = [] # Fitted x
    for i in fit_x:
      cal = m*(i*scale) + c
      cal_fit_x.append(cal)

    E.append(E0)
    dE.append(E0_err)
    Res.append(R)
    dRes.append(R_err)
    n_peak = n_peak + 1

dframe = pd.DataFrame({"E0" : E, "dE0" : dE, "R" : Res, "dR" : dRes})
dframe.to_csv("Resolution.csv", index=False)
files.download('Resolution.csv')
```

Code to Identify Peaks

```
##################
# Python modules #
##################
%matplotlib inline

import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt

import scipy
from scipy.optimize import curve_fit
from scipy.special import factorial

from google.colab import files
from google.colab import drive
drive.mount('/content/drive/')

#########
# Files #
#########
# 25-2 #
#Background = '25_2_2020/BG10-59.csv'
#file = '25_2_2020/11DRY-11-42.csv'
#file = '25_2_2020/12MEDIUMWET-11-24.csv'
#file = '25_2_2020/SAMPLEB-11-59.csv'

# 3-3 #
Background = '03_3_2020/BG-11-07.csv'
#file = '03_3_2020/11DRY-11-40.csv'
#file = '03_3_2020/12MEDIUM-12-46.csv' # 976th = 662keV, 2156th = 1462keV, 1430th = 969.5
    ↪ keV, 3253th = 2205.5keV
file = '03_3_2020/SAMPLEB-12-12.csv'

# 10-3 #
#Background = '10_3_2020/BG11-16.csv'
#file = '10_3_2020/BLIND1_11-57.csv'
#file = '10_3_2020/BLIND2_12-59.csv'
#file = '10_3_2020/BLIND3_12-31.csv'

# Energy Resolution #
Resolution = 'Energy␣Resolution/Resolution.csv'
# Reading file names
sample = file[10:-10]
date = file[0:9]

# Ensuring anyone can read the file from their local machine
directories = ['/content/drive/My␣Drive/Uni/PHAS52␣Group␣Project/CSV_Data/', # Ira path
              '/content/drive/My␣Drive/UCL/Year␣3/PHAS52␣Group␣Project/CSV_Data/', #
                  ↪ Kanishk path
              '/content/drive/My␣Drive/PHAS52␣Group␣Project/CSV_Data/' # Pip/John path
              ]
```

```
# Background #
for i in (directories):
  try:
    db = pd.read_csv(i+Background)
  except (FileNotFoundError):
    pass
# Data #
for i in (directories):
  try:
    df = pd.read_csv(i+file)
  except (FileNotFoundError):
    pass
# Energy Resolution #
for i in (directories):
  try:
    dr = pd.read_csv(i+Resolution)
  except (FileNotFoundError):
    pass

######################
# Energy resolution #
######################
# Reading data #
resolution = dr[0:].rename(columns=db.iloc[5])
# converting to floats
E0 = [float(i) for i in resolution['E0']]
e_res = [float(i) for i in resolution['R']]
E0_err = [float(i) for i in resolution['dE0']]
e_res_err = [float(i) for i in resolution['dR']]
# coverting data to a list
E0 = np.array(E0)
e_res = np.array(e_res)
E0_err = np.array(E0_err)
e_res_err = np.array(e_res_err)

# curve fit #
def energy_resolution(A,B,x):
    """Energy resolution function
    Equation: R = FWHM / E0 * 100
    x : mean energy
    """
    y = A / np.sqrt(x) + B
    return y

def chi2(parameter, x, expData, y_error):
    """Calculates the chi squared for a theorectical fit and experimental data
    Needs the error as a numby array
    parameter: fitting parameters
    x: x values
    expData:
    y_error: error on the y values
    """

    A,B = parameter
```

```
    theoData = energy_resolution(A,B,x) # theoretical data
    chiSquared = np.sum(np.power((expData-theoData),2)/y_error**2)

    return chiSquared

# initial guesses
initial_guess = [1,1]
# optimisation
fit = scipy.optimize.minimize(chi2, initial_guess,args = (E0, e_res, e_res_err),method =
    ↪ 'L-BFGS-B')
fitted_params = fit.x
A,B = fitted_params
# error of fit
traceHessian = np.diag(fit.hess_inv.todense()) # trace of hessian matrix
ftol = 2.220446049250313e-09 # tolerance value at which opimization is stopped - DEFAULT
A_error = np.sqrt(max(1, abs(fit.fun)) * ftol * traceHessian[0])
B_error = np.sqrt(max(1, abs(fit.fun)) * ftol * traceHessian[1])
# fitted data
fit_E0 = np.linspace(300,1500,1201)
fit_e_res = energy_resolution(A,B,fit_E0)
# chi2 if the fit
x2 = chi2(fitted_params, E0, e_res, e_res_err)
dofs = len(E0) - len(initial_guess) # as we fit 3 parameters
chiSquaredRed = x2/dofs
p_value = scipy.stats.chi2.sf(x2,dofs)
#print("Reduced x2:",chiSquaredRed)
#print("P value:",p_value)

# Graph #
plt.figure(figsize=(15,10))
plt.plot(E0, e_res, 'kx',linewidth=0.5,label='Calibration data')
plt.errorbar(E0, e_res,yerr=e_res_err,xerr=E0_err,fmt='red', linestyle='None', capsize=2)
plt.plot(fit_E0,fit_e_res,label='Fitted curve:$R = A/\sqrt{E}+B$') # fit
plt.plot(np.NaN, np.NaN, '-', color='none', label= '$A$ = ({:.5f}  {:.5f}),$B$ = ({:.6f}
    ↪   {:.6f})'.format(A,A_error,B,B_error))
plt.plot(np.NaN, np.NaN, '-', color='none', label='Reduced $\chi^2$ = {:.2f}'.format(
    ↪ chiSquaredRed))
# graph setting
plt.title("Energy Resolution of the Mirrion P-Type Coaxial Germanium Detector")
plt.xlabel("Energy / keV")
plt.ylabel("R / %")
plt.legend(fontsize = 12)
plt.grid()

# save file #
#plt.savefig('Energy_Resolution.png')
#files.download('Energy_Resolution.png')

################
# Sample data #
################
# Background #
legend = db[:5]
back = db[6:].rename(columns=db.iloc[5])
```

```
# converting to floats
back_energy = [float(i) for i in back['Energy␣(keV)']]
back_counts = [float(i) for i in back['Counts']]

# Data #
header = df[:5]
data = df[6:].rename(columns=df.iloc[5])
# converting to floats
data_energy = [float(i) for i in data['Energy␣(keV)']]
data_counts = [float(i) for i in data['Counts']]
energy = data_energy

# Background reduction #
b_time = float(np.array(db.loc[1])[1]) # live time
d_time = float(np.array(df.loc[1])[1]) # live time
back_cal = d_time/b_time
# Array of the calibrated background counts
cal_back_counts = []
for x in back_counts:
    cal = x * back_cal
    cal_back_counts.append(cal)

# Calibration #
# Slope calibration
word = np.array(df.loc[0])[2]
slope = float(word.strip('Slope:␣'))
# Calibration parameter
scale = 1/ slope
m = 0.678
c = -0.642
# Array of calibrated energy
cal_energy = []
for x in energy:
   cal = m*(x*scale) + c
   cal_energy.append(cal)

   # filter out peak sections #
high_back = 443 # 443th position = 300keV
limit = 1769
reduced_energy = energy[high_back:limit]
reduced_data_counts = data_counts[high_back:limit]
reduced_back_counts = back_counts[high_back:limit]

##########################
# Peak finding function #
##########################
from scipy.signal import find_peaks
spikes, _ = find_peaks(reduced_data_counts, height=10,distance=1) # Data (clipped)

# Finding peaks in calibrated energy
peaks = []
int_peaks = []
for i in spikes:
   fill = m*(reduced_energy[i]*scale) + c
```

```
   int_fill = int(fill)
   peaks.append(fill)
   int_peaks.append(int_fill)

# Required data #
# converting energy resolution data to list
E0_list = list(fit_E0)
e_res_list = list(fit_e_res)
# calibration of data
cal_x = [] # Data x
for i in reduced_energy:
  cal = m*(i*scale) + c
  cal_x.append(cal)

##################
# Peak Analysis #
##################
# Determining the energy resolution
j = 0
find = []
while j < len(int_peaks):
    # finding the energy resolution of particular energy level
    find_x = E0_list.index(int_peaks[j])
    find_y = e_res_list[find_x]
    find.append(find_y)
    j = j + 1

k = 0
evidence = []
sigma = []
height = []
while k < len(peaks):
    # Defining he resolution range
    range_start, range_end = peaks[k] - find[k], peaks[k] + find[k]
    k = k + 1
    # finding the nth position of the start and end if the energy resolution
    for i in cal_x:
      if i < range_start:
        pos_start = cal_x.index(i) + 1
        pass
      if i < range_end:
        pos_end = cal_x.index(i) + 1
        pass

    # clipping data from energy resolution
    energy_in_range = cal_x[pos_start:pos_end]
    counts_in_range = reduced_data_counts[pos_start:pos_end]
    back_counts_in_range = reduced_back_counts[pos_start:pos_end]
    # converting the sliced data to array
    x = np.array(energy_in_range)
    y = np.array(counts_in_range)
    b = np.array(back_counts_in_range)
    # sum of the counts
    N = sum(y)
```

```
    b_N = sum(b)

    # converting data to 1d array #
    # Data
    counter = 0
    xi = []
    while len(y) > counter:
        fill = np.ones(int(y[counter])) * x[counter]
        xi.extend(fill)
        counter = counter + 1
    # Background
    b_counter = 0
    b_xi = []
    while len(b) > b_counter:
        fill = np.ones(int(b[b_counter])) * x[b_counter]
        b_xi.extend(fill)
        b_counter = b_counter + 1

    mean, sd = stats.norm.fit(xi)
    b_mean, b_sd = stats.norm.fit(b_xi)
    # calclating five sigma
    difference_counts = np.sum(y) - np.sum(b)
    uncertanty_counts = np.sqrt(np.sum(y) + np.sum(b))
    five_sigma = difference_counts / uncertanty_counts
    # collecting data
    evidence.append(mean)
    sigma.append(five_sigma)
    height.append(y)

Energy_peaks = []
dEnergy_peaks = []
five_sigma = []
arrow_base = []
for i in sigma:
    if i > 3:
      find_sigma = sigma.index(i)
      E0 = evidence[find_sigma]
      dE0 = find[find_sigma]*E0/100
       = sigma[find_sigma]
      a = height[find_sigma]
      # filling the array
      Energy_peaks.append(round(E0,1))
      dEnergy_peaks.append(round(dE0,1))
      five_sigma.append(round(,1))
      arrow_base.append(a)
      #print("E0:",round(E0,1),'',round(dE0,1),"keV","|:",round(,1))

# Graph #
plt.figure(figsize=(15,10))
plt.plot(cal_energy, data_counts, 'r',linewidth = 0.5, label = 'Sample')
plt.plot(cal_energy, cal_back_counts,linewidth = 0.5, label = 'Background')
# Peaks pointers
y_lim = max(reduced_data_counts) * 1.2
plot = 0
```

```
while plot < len(Energy_peaks):
    if plot + 1 == len(Energy_peaks):
      plt.arrow(Energy_peaks[plot],arrow_base[plot]+(y_lim-y_lim/1.5)+y_lim/8/8,0,-(y_lim-
          ↪ y_lim/1.5),head_width=20,head_length=2)
      plt.annotate('({:.0f}␣␣{:.0f})␣keV␣[{}]'.format(Energy_peaks[plot],dEnergy_peaks[
          ↪ plot],five_sigma[plot]),(Energy_peaks[plot],arrow_base[plot]+(y_lim-y_lim
          ↪ /1.55)))
      plot = plot + 1
    elif Energy_peaks[plot+1]-Energy_peaks[plot] < 300:
      plt.arrow(Energy_peaks[plot],arrow_base[plot]+(y_lim-y_lim/1.7)+y_lim/8/8,0,-(y_lim-
          ↪ y_lim/1.7),head_width=20,head_length=2)
      plt.annotate('({:.0f}␣␣{:.0f})␣keV␣[{}]'.format(Energy_peaks[plot],dEnergy_peaks[
          ↪ plot],five_sigma[plot]),(Energy_peaks[plot],arrow_base[plot]+(y_lim-y_lim
          ↪ /1.75)))
      plot = plot + 1
    else:
      plt.arrow(Energy_peaks[plot],arrow_base[plot]+(y_lim-y_lim/1.5)+y_lim/8/8,0,-(y_lim-
          ↪ y_lim/1.5),head_width=20,head_length=2)
      plt.annotate('({:.0f}␣␣{:.0f})␣keV␣[{}]'.format(Energy_peaks[plot],dEnergy_peaks[
          ↪ plot],five_sigma[plot]),(Energy_peaks[plot],arrow_base[plot]+(y_lim-y_lim
          ↪ /1.55)))
      plot = plot + 1
# Graph setting
plt.title('{},␣Date:␣{},␣Exposure␣Time:␣{}s'.format(sample,date,d_time))
plt.ylabel('Counts')
plt.ylim(0,y_lim)
plt.xlabel('Energy␣/␣keV')
plt.xlim(0,2500)
plt.legend(fontsize=12)
plt.grid()

plt.savefig('{}_{}.png'.format(date,sample))
files.download('{}_{}.png'.format(date,sample))
```

Appendix E: Geant4 Code

Modified from GEANT4 example B4c — `https://gitlab.cern.ch/geant4/geant4/-/tree/master/examples/basic/B4/B4c`

Make Files

CMakeLists.txt

```
#----------------------------------------------------------------------------
# Setup the project
#
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)
project(B4c)

#----------------------------------------------------------------------------
# Find Geant4 package, activating all available UI and Vis drivers by default
# You can set WITH_GEANT4_UIVIS to OFF via the command line or ccmake/cmake-gui
# to build a batch mode only executable
#
option(WITH_GEANT4_UIVIS "Build␣example␣with␣Geant4␣UI␣and␣Vis␣drivers" ON)
if(WITH_GEANT4_UIVIS)
  find_package(Geant4 REQUIRED ui_all vis_all)
else()
  find_package(Geant4 REQUIRED)
endif()

#----------------------------------------------------------------------------
# Setup Geant4 include directories and compile definitions
# Setup include directory for this project
#
include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/include)

#----------------------------------------------------------------------------
# Locate sources and headers for this project
# NB: headers are included so they will show up in IDEs
#
file(GLOB sources ${PROJECT_SOURCE_DIR}/src/*.cc)
file(GLOB headers ${PROJECT_SOURCE_DIR}/include/*.hh)

#----------------------------------------------------------------------------
# Add the executable, and link it to the Geant4 libraries
#
add_executable(exampleB4c exampleB4c.cc ${sources} ${headers})
target_link_libraries(exampleB4c ${Geant4_LIBRARIES})

#----------------------------------------------------------------------------
# Copy all scripts to the build directory, i.e. the directory in which we
# build B4c. This is so that we can run the executable directly because it
# relies on these scripts being in the current working directory.
#
set(EXAMPLEB4C_SCRIPTS
```

```
  exampleB4c.out
  exampleB4.in
  gui.mac
  init_vis.mac
  plotHisto.C
  plotNtuple.C
  run1.mac
  run2.mac
  vis.mac
  )

foreach(_script ${EXAMPLEB4C_SCRIPTS})
  configure_file(
    ${PROJECT_SOURCE_DIR}/${_script}
    ${PROJECT_BINARY_DIR}/${_script}
    COPYONLY
    )
endforeach()

#----------------------------------------------------------------------------
# Install the executable to 'bin' directory under CMAKE_INSTALL_PREFIX
#
install(TARGETS exampleB4c DESTINATION bin)
```

GNUmakefile

```
name := exampleB4c
G4TARGET := $(name)
G4EXLIB := true

ifndef G4INSTALL
  G4INSTALL = ../../..
endif

.PHONY: all
all: lib bin

include $(G4INSTALL)/config/binmake.gmk

visclean:
	rm -f g4*.prim g4*.eps g4*.wrl
	rm -f .DAWN_*
```

exampleB4c.cc

```
#include "B4cDetectorConstruction.hh"
#include "B4cActionInitialization.hh"

#ifdef G4MULTITHREADED
#include "G4MTRunManager.hh"
```

```
#else
#include "G4RunManager.hh"
#endif

#include "G4UImanager.hh"
#include "G4UIcommand.hh"
#include "FTFP_BERT.hh"

#include "Randomize.hh"

#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

namespace {
  void PrintUsage() {
    G4cerr << "␣Usage:␣" << G4endl;
    G4cerr << "␣exampleB4c␣[-m␣macro␣]␣[-u␣UIsession]␣[-t␣nThreads]" << G4endl;
    G4cerr << "␣␣␣note:␣-t␣option␣is␣available␣only␣for␣multi-threaded␣mode."
           << G4endl;
  }
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

int main(int argc,char** argv)
{
  // Evaluate arguments
  //
  if ( argc > 7 ) {
    PrintUsage();
    return 1;
  }

  G4String macro;
  G4String session;
#ifdef G4MULTITHREADED
  G4int nThreads = 0;
#endif
  for ( G4int i=1; i<argc; i=i+2 ) {
    if ( G4String(argv[i]) == "-m" ) macro = argv[i+1];
    else if ( G4String(argv[i]) == "-u" ) session = argv[i+1];
#ifdef G4MULTITHREADED
    else if ( G4String(argv[i]) == "-t" ) {
      nThreads = G4UIcommand::ConvertToInt(argv[i+1]);
    }
#endif
    else {
      PrintUsage();
      return 1;
    }
  }
```

```
  // Detect interactive mode (if no macro provided) and define UI session
  //
  G4UIExecutive* ui = 0;
  if ( ! macro.size() ) {
    ui = new G4UIExecutive(argc, argv, session);
  }

  // Optionally: choose a different Random engine...
  //
  // G4Random::setTheEngine(new CLHEP::MTwistEngine);

  // Construct the default run manager
  //
#ifdef G4MULTITHREADED
  G4MTRunManager * runManager = new G4MTRunManager;
  if ( nThreads > 0 ) {
    runManager->SetNumberOfThreads(nThreads);
  }
#else
  G4RunManager * runManager = new G4RunManager;
#endif

  // Set mandatory initialization classes
  //
  auto detConstruction = new B4cDetectorConstruction();
  runManager->SetUserInitialization(detConstruction);

  auto physicsList = new FTFP_BERT;
  runManager->SetUserInitialization(physicsList);

  auto actionInitialization = new B4cActionInitialization();
  runManager->SetUserInitialization(actionInitialization);

  // Initialize visualization
  auto visManager = new G4VisExecutive;
  // G4VisExecutive can take a verbosity argument - see /vis/verbose guidance.
  // G4VisManager* visManager = new G4VisExecutive("Quiet");
  visManager->Initialize();

  // Get the pointer to the User Interface manager
  auto UImanager = G4UImanager::GetUIpointer();

  // Process macro or start UI session
  //
  if ( macro.size() ) {
    // batch mode
    G4String command = "/control/execute␣";
    UImanager->ApplyCommand(command+macro);
  }
  else {
    // interactive mode : define UI session
    UImanager->ApplyCommand("/control/execute␣init_vis.mac");
    if (ui->IsGUI()) {
      UImanager->ApplyCommand("/control/execute␣gui.mac");
```

```
    }
    ui->SessionStart();
    delete ui;
  }

  // Job termination
  // Free the store: user actions, physics_list and detector_description are
  // owned and deleted by the run manager, so they should not be deleted
  // in the main() program !

  delete visManager;
  delete runManager;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo.....
```

Header Files

B4PrimaryGeneratorAction.hh

```
#ifndef B4PrimaryGeneratorAction_h
#define B4PrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"
#include "G4GeneralParticleSource.hh"
#include "globals.hh"

class G4ParticleGun;
class G4Event;

/// The primary generator action class with particle gum.
///
/// It defines a single particle which hits the calorimeter
/// perpendicular to the input face. The type of the particle
/// can be changed via the G4 build-in commands of G4ParticleGun class
/// (see the macros provided with this example).

class B4PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
public:
  B4PrimaryGeneratorAction();
  virtual ~B4PrimaryGeneratorAction();

  virtual void GeneratePrimaries(G4Event* event);

  // set methods
  void SetRandomFlag(G4bool value);

private:
  G4GeneralParticleSource* fParticleGun; // G4 particle gun
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B4RunAction.hh

```
#ifndef B4RunAction_h
#define B4RunAction_h 1

#include "G4UserRunAction.hh"
#include "globals.hh"

class G4Run;

/// Run action class
///
/// It accumulates statistic and computes dispersion of the energy deposit
/// and track lengths of charged particles with use of analysis tools:
/// H1D histograms are created in BeginOfRunAction() for the following
/// physics quantities:
/// - Edep in absorber
/// - Edep in gap
/// - Track length in absorber
/// - Track length in gap
/// The same values are also saved in the ntuple.
/// The histograms and ntuple are saved in the output file in a format
/// accoring to a selected technology in B4Analysis.hh.
///
/// In EndOfRunAction(), the accumulated statistic and computed
/// dispersion is printed.
///

class B4RunAction : public G4UserRunAction
{
  public:
    B4RunAction();
    virtual ~B4RunAction();

    virtual void BeginOfRunAction(const G4Run*);
    virtual void EndOfRunAction(const G4Run*);
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B4cActionInitialization.hh

```
#ifndef B4cActionInitialization_h
#define B4cActionInitialization_h 1

#include "G4VUserActionInitialization.hh"

/// Action initialization class.
///

class B4cActionInitialization : public G4VUserActionInitialization
{
  public:
    B4cActionInitialization();
    virtual ~B4cActionInitialization();

    virtual void BuildForMaster() const;
    virtual void Build() const;
};

#endif
```

B4cCalorHit.hh

```
#ifndef B4cCalorHit_h
#define B4cCalorHit_h 1

#include "G4VHit.hh"
#include "G4THitsCollection.hh"
#include "G4Allocator.hh"
#include "G4ThreeVector.hh"
#include "G4Threading.hh"

/// Calorimeter hit class
///
/// It defines data members to store the the energy deposit and track lengths
/// of charged particles in a selected volume:
/// - fEdep, fTrackLength

class B4cCalorHit : public G4VHit
{
  public:
    B4cCalorHit();
    B4cCalorHit(const B4cCalorHit&);
    virtual ~B4cCalorHit();

    // operators
    const B4cCalorHit& operator=(const B4cCalorHit&);
    G4bool operator==(const B4cCalorHit&) const;

    inline void* operator new(size_t);
    inline void operator delete(void*);

    // methods from base class
    virtual void Draw() {}
    virtual void Print();

    // methods to handle data
    void Add(G4double de, G4double dl);

    // get methods
    G4double GetEdep() const;
    G4double GetTrackLength() const;

  private:
    G4double fEdep; ///< Energy deposit in the sensitive volume
    G4double fTrackLength; ///< Track length in the sensitive volume
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

using B4cCalorHitsCollection = G4THitsCollection<B4cCalorHit>;

extern G4ThreadLocal G4Allocator<B4cCalorHit>* B4cCalorHitAllocator;
```

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

inline void* B4cCalorHit::operator new(size_t)
{
  if (!B4cCalorHitAllocator) {
    B4cCalorHitAllocator = new G4Allocator<B4cCalorHit>;
  }
  void *hit;
  hit = (void *) B4cCalorHitAllocator->MallocSingle();
  return hit;
}

inline void B4cCalorHit::operator delete(void *hit)
{
  if (!B4cCalorHitAllocator) {
    B4cCalorHitAllocator = new G4Allocator<B4cCalorHit>;
  }
  B4cCalorHitAllocator->FreeSingle((B4cCalorHit*) hit);
}

inline void B4cCalorHit::Add(G4double de, G4double dl) {
  fEdep += de;
  fTrackLength += dl;
}

inline G4double B4cCalorHit::GetEdep() const {
  return fEdep;
}

inline G4double B4cCalorHit::GetTrackLength() const {
  return fTrackLength;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B4cCalorimeterSD.hh

```
#ifndef B4cCalorimeterSD_h
#define B4cCalorimeterSD_h 1

#include "G4VSensitiveDetector.hh"

#include "B4cCalorHit.hh"

#include <vector>

class G4Step;
class G4HCofThisEvent;

/// Calorimeter sensitive detector class
///
/// In Initialize(), it creates one hit for each calorimeter layer and one more
/// hit for accounting the total quantities in all layers.
///
/// The values are accounted in hits in ProcessHits() function which is called
/// by Geant4 kernel at each step.

class B4cCalorimeterSD : public G4VSensitiveDetector
{
  public:
    B4cCalorimeterSD(const G4String& name,
                     const G4String& hitsCollectionName,
                     G4int nofCells);
    virtual ~B4cCalorimeterSD();

    // methods from base class
    virtual void Initialize(G4HCofThisEvent* hitCollection);
    virtual G4bool ProcessHits(G4Step* step, G4TouchableHistory* history);
    virtual void EndOfEvent(G4HCofThisEvent* hitCollection);

  private:
    B4cCalorHitsCollection* fHitsCollection;
    G4int fNofCells;
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

B4cDetectorConstruction.hh

```
#ifndef B4cDetectorConstruction_h
#define B4cDetectorConstruction_h 1

#include "G4VUserDetectorConstruction.hh"
#include "globals.hh"

class G4VPhysicalVolume;
class G4GlobalMagFieldMessenger;

/// Detector construction class to define materials and geometry.
/// The calorimeter is a box made of a given number of layers. A layer consists
/// of an absorber plate and of a detection gap. The layer is replicated.
///
/// Four parameters define the geometry of the calorimeter :
///
/// - the thickness of an absorber plate,
/// - the thickness of a gap,
/// - the number of layers,
/// - the transverse size of the calorimeter (the input face is a square).
///
/// In ConstructSDandField() sensitive detectors of B4cCalorimeterSD type
/// are created and associated with the Absorber and Gap volumes.
/// In addition a transverse uniform magnetic field is defined
/// via G4GlobalMagFieldMessenger class.

class B4cDetectorConstruction : public G4VUserDetectorConstruction
{
  public:
    B4cDetectorConstruction();
    virtual ~B4cDetectorConstruction();

  public:
    virtual G4VPhysicalVolume* Construct();
    virtual void ConstructSDandField();

  private:
    // methods
    //
    void DefineMaterials();
    G4VPhysicalVolume* DefineVolumes();

    // data members
    //
    static G4ThreadLocal G4GlobalMagFieldMessenger* fMagFieldMessenger;
                                  // magnetic field messenger

    G4bool fCheckOverlaps; // option to activate checking of volumes overlaps
    G4int fNofLayers; // number of layers
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

```
#endif
```

B4cEventAction.hh

```
#ifndef B4cEventAction_h
#define B4cEventAction_h 1

#include "G4UserEventAction.hh"

#include "B4cCalorHit.hh"

#include "globals.hh"

/// Event action class
///
/// In EndOfEventAction(), it prints the accumulated quantities of the energy
/// deposit and track lengths of charged particles in Absober and Gap layers
/// stored in the hits collections.

class B4cEventAction : public G4UserEventAction
{
public:
  B4cEventAction();
  virtual ~B4cEventAction();

  virtual void BeginOfEventAction(const G4Event* event);
  virtual void EndOfEventAction(const G4Event* event);

private:
  // methods
  B4cCalorHitsCollection* GetHitsCollection(G4int hcID,
                                      const G4Event* event) const;
  void PrintEventStatistics(G4double absoEdep, G4double absoTrackLength,
                            G4double gapEdep, G4double gapTrackLength) const;

  // data members
  G4int fAbsHCID;
  G4int fGapHCID;
};

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#endif
```

Source Files

B4PrimaryGeneratorAction.cc

```
#include "B4PrimaryGeneratorAction.hh"

#include "G4RunManager.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4LogicalVolume.hh"
#include "G4Box.hh"
#include "G4Event.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "G4SystemOfUnits.hh"
#include "Randomize.hh"
// #include "G4GeneralParticleSource.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4PrimaryGeneratorAction::B4PrimaryGeneratorAction()
 : G4VUserPrimaryGeneratorAction(),
   fParticleGun(nullptr)
{
  fParticleGun = new G4GeneralParticleSource();

  // default particle kinematic
  //
  auto particleDefinition
    = G4ParticleTable::GetParticleTable()->FindParticle("gamma");
  fParticleGun->SetParticleDefinition(particleDefinition);
}


//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4PrimaryGeneratorAction::~B4PrimaryGeneratorAction()
{
  delete fParticleGun;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  // This function is called at the begining of event

  // In order to avoid dependence of PrimaryGeneratorAction
  // on DetectorConstruction class we get world volume
  // from G4LogicalVolumeStore
  //
  G4double worldZHalfLength = 0.;
  auto worldLV = G4LogicalVolumeStore::GetInstance()->GetVolume("World");
```

```
  // Check that the world volume has box shape
  G4Box* worldBox = nullptr;
  if ( worldLV ) {
    worldBox = dynamic_cast<G4Box*>(worldLV->GetSolid());
  }

  if ( worldBox ) {
    worldZHalfLength = worldBox->GetZHalfLength();
  }
  else {
    G4ExceptionDescription msg;
    msg << "World␣volume␣of␣box␣shape␣not␣found." << G4endl;
    msg << "Perhaps␣you␣have␣changed␣geometry." << G4endl;
    msg << "The␣gun␣will␣be␣place␣in␣the␣center.";
    G4Exception("B4PrimaryGeneratorAction::GeneratePrimaries()",
      "MyCode0002", JustWarning, msg);
  }

  // Set gun position
  fParticleGun
    ->SetParticlePosition(G4ThreeVector(0., 0., -worldZHalfLength));

  fParticleGun->GeneratePrimaryVertex(anEvent);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B4RunAction.cc

```
#include "B4RunAction.hh"
#include "B4Analysis.hh"

#include "G4Run.hh"
#include "G4RunManager.hh"
#include "G4UnitsTable.hh"
#include "G4SystemOfUnits.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4RunAction::B4RunAction()
 : G4UserRunAction()
{
  // set printing event number per each event
  G4RunManager::GetRunManager()->SetPrintProgress(1);

  // Create analysis manager
  // The choice of analysis technology is done via selectin of a namespace
  // in B4Analysis.hh
  auto analysisManager = G4AnalysisManager::Instance();
  G4cout << "Using␣" << analysisManager->GetType() << G4endl;

  // Create directories
  analysisManager->SetHistoDirectoryName("histograms");
  analysisManager->SetNtupleDirectoryName("ntuple");
  analysisManager->SetVerboseLevel(1);
  //analysisManager->SetNtupleMerging(true);
    // Note: merging ntuples is available only with Root output

  // Book histograms, ntuple
  //

  // Creating histograms
  //analysisManager->CreateH1("Eabs","Edep in absorber", 100, 0., 800*MeV);
  //analysisManager->CreateH1("Egap","Edep in gap", 100, 0., 100*MeV);
  //analysisManager->CreateH1("Labs","trackL in absorber", 100, 0., 1*m);
  //analysisManager->CreateH1("Lgap","trackL in gap", 100, 0., 50*cm);

  // Creating ntuple
  //
  analysisManager->CreateNtuple("B4", "Edep␣and␣TrackL");
  analysisManager->CreateNtupleDColumn("Eabs");
  analysisManager->CreateNtupleDColumn("Egap");
  analysisManager->CreateNtupleDColumn("Labs");
  analysisManager->CreateNtupleDColumn("Lgap");
  analysisManager->FinishNtuple();
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4RunAction::~B4RunAction()
```

```
{
  delete G4AnalysisManager::Instance();
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4RunAction::BeginOfRunAction(const G4Run* /*run*/)
{
  //inform the runManager to save random number seed
  //G4RunManager::GetRunManager()->SetRandomNumberStore(true);

  // Get analysis manager
  auto analysisManager = G4AnalysisManager::Instance();

  // Open an output file
  //
  G4String fileName = "B4";
  analysisManager->OpenFile(fileName);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4RunAction::EndOfRunAction(const G4Run* /*run*/)
{
  // print histogram statistics
  //
  auto analysisManager = G4AnalysisManager::Instance();
  //if ( analysisManager->GetH1(1) ) {
  // G4cout << G4endl << " ----> print histograms statistic ";
  // if(isMaster) {
  // G4cout << "for the entire run " << G4endl << G4endl;
  // }
  // else {
  // G4cout << "for the local thread " << G4endl << G4endl;
  // }


  // G4cout << " EAbs : mean = "
  // << G4BestUnit(analysisManager->GetH1(0)->mean(), "Energy")
  // << " rms = "
  // << G4BestUnit(analysisManager->GetH1(0)->rms(), "Energy") << G4endl;
  //
  // G4cout << " EGap : mean = "
  // << G4BestUnit(analysisManager->GetH1(1)->mean(), "Energy")
  // << " rms = "
  // << G4BestUnit(analysisManager->GetH1(1)->rms(), "Energy") << G4endl;
  //
  // G4cout << " LAbs : mean = "
  // << G4BestUnit(analysisManager->GetH1(2)->mean(), "Length")
  // << " rms = "
  // << G4BestUnit(analysisManager->GetH1(2)->rms(), "Length") << G4endl;

  // G4cout << " LGap : mean = "
  // << G4BestUnit(analysisManager->GetH1(3)->mean(), "Length")
```

```
  // << " rms = "
  // << G4BestUnit(analysisManager->GetH1(3)->rms(), "Length") << G4endl;
  //}

  // save histograms & ntuple
  //
  analysisManager->Write();
  analysisManager->CloseFile();
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B4cActionInitialization.cc

```
#include "B4cActionInitialization.hh"
#include "B4PrimaryGeneratorAction.hh"
#include "B4RunAction.hh"
#include "B4cEventAction.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cActionInitialization::B4cActionInitialization()
 : G4VUserActionInitialization()
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cActionInitialization::~B4cActionInitialization()
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cActionInitialization::BuildForMaster() const
{
  SetUserAction(new B4RunAction);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cActionInitialization::Build() const
{
  SetUserAction(new B4PrimaryGeneratorAction);
  SetUserAction(new B4RunAction);
  SetUserAction(new B4cEventAction);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B4cActionInitialization.cc

```
#include "B4cActionInitialization.hh"
#include "B4PrimaryGeneratorAction.hh"
#include "B4RunAction.hh"
#include "B4cEventAction.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cActionInitialization::B4cActionInitialization()
 : G4VUserActionInitialization()
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cActionInitialization::~B4cActionInitialization()
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cActionInitialization::BuildForMaster() const
{
  SetUserAction(new B4RunAction);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cActionInitialization::Build() const
{
  SetUserAction(new B4PrimaryGeneratorAction);
  SetUserAction(new B4RunAction);
  SetUserAction(new B4cEventAction);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B4cCalorHit.cc

```
#include "B4cCalorHit.hh"
#include "G4UnitsTable.hh"
#include "G4VVisManager.hh"
#include "G4Circle.hh"
#include "G4Colour.hh"
#include "G4VisAttributes.hh"

#include <iomanip>

G4ThreadLocal G4Allocator<B4cCalorHit>* B4cCalorHitAllocator = 0;

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cCalorHit::B4cCalorHit()
 : G4VHit(),
   fEdep(0.),
   fTrackLength(0.)
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cCalorHit::~B4cCalorHit() {}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cCalorHit::B4cCalorHit(const B4cCalorHit& right)
  : G4VHit()
{
  fEdep = right.fEdep;
  fTrackLength = right.fTrackLength;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

const B4cCalorHit& B4cCalorHit::operator=(const B4cCalorHit& right)
{
  fEdep = right.fEdep;
  fTrackLength = right.fTrackLength;

  return *this;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

G4bool B4cCalorHit::operator==(const B4cCalorHit& right) const
{
  return ( this == &right ) ? true : false;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

```
void B4cCalorHit::Print()
{
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B4cCalorimeterSD.cc

```
#include "B4cCalorimeterSD.hh"
#include "G4HCofThisEvent.hh"
#include "G4Step.hh"
#include "G4ThreeVector.hh"
#include "G4SDManager.hh"
#include "G4ios.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cCalorimeterSD::B4cCalorimeterSD(
                         const G4String& name,
                         const G4String& hitsCollectionName,
                         G4int nofCells)
 : G4VSensitiveDetector(name),
   fHitsCollection(nullptr),
   fNofCells(nofCells)
{
  collectionName.insert(hitsCollectionName);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cCalorimeterSD::~B4cCalorimeterSD()
{
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cCalorimeterSD::Initialize(G4HCofThisEvent* hce)
{
  // Create hits collection
  fHitsCollection
    = new B4cCalorHitsCollection(SensitiveDetectorName, collectionName[0]);

  // Add this collection in hce
  auto hcID
    = G4SDManager::GetSDMpointer()->GetCollectionID(collectionName[0]);
  hce->AddHitsCollection( hcID, fHitsCollection );

  // Create hits
  // fNofCells for cells + one more for total sums
  for (G4int i=0; i<fNofCells+1; i++ ) {
    fHitsCollection->insert(new B4cCalorHit());
  }
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

G4bool B4cCalorimeterSD::ProcessHits(G4Step* step,
                                     G4TouchableHistory*)
{
```

```
  // energy deposit
  auto edep = step->GetTotalEnergyDeposit();

  // step length
  G4double stepLength = 0.;
  if ( step->GetTrack()->GetDefinition()->GetPDGCharge() != 0. ) {
    stepLength = step->GetStepLength();
  }

  if ( edep==0. && stepLength == 0. ) return false;

  auto touchable = (step->GetPreStepPoint()->GetTouchable());

  // Get calorimeter cell id
  auto layerNumber = touchable->GetReplicaNumber(1);

  // Get hit accounting data for this cell
  auto hit = (*fHitsCollection)[layerNumber];
  if ( ! hit ) {
    G4ExceptionDescription msg;
    msg << "Cannot␣access␣hit␣" << layerNumber;
    G4Exception("B4cCalorimeterSD::ProcessHits()",
      "MyCode0004", FatalException, msg);
  }

  // Get hit for total accounting
  auto hitTotal
    = (*fHitsCollection)[fHitsCollection->entries()-1];

  // Add values
  hit->Add(edep, stepLength);
  hitTotal->Add(edep, stepLength);

  return true;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cCalorimeterSD::EndOfEvent(G4HCofThisEvent*)
{
   auto nofHits = fHitsCollection->entries();
   G4cout
     << G4endl
     << "-------->Hits␣Collection:␣in␣this␣event␣they␣are␣" << nofHits
     << "␣hits␣in␣the␣tracker␣chambers:␣" << G4endl;
   for ( std::size_t i=0; i<nofHits; ++i ) (*fHitsCollection)[i]->Print();
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B4cDetectorConstruction.cc

```
#include "B4cDetectorConstruction.hh"
#include "B4cCalorimeterSD.hh"
#include "G4Material.hh"
#include "G4NistManager.hh"

#include "G4Box.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4PVReplica.hh"
#include "G4GlobalMagFieldMessenger.hh"
#include "G4AutoDelete.hh"

#include "G4SDManager.hh"

#include "G4VisAttributes.hh"
#include "G4Colour.hh"

#include "G4PhysicalConstants.hh"
#include "G4SystemOfUnits.hh"
#include "G4Tubs.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

G4ThreadLocal
G4GlobalMagFieldMessenger* B4cDetectorConstruction::fMagFieldMessenger = 0;

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cDetectorConstruction::B4cDetectorConstruction()
 : G4VUserDetectorConstruction(),
   fCheckOverlaps(true),
   fNofLayers(-1)
{
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cDetectorConstruction::~B4cDetectorConstruction()
{
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

G4VPhysicalVolume* B4cDetectorConstruction::Construct()
{
  // Define materials
  DefineMaterials();

  // Define volumes
  return DefineVolumes();
}
```

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cDetectorConstruction::DefineMaterials()
{
  // Lead material defined using NIST Manager
  auto nistManager = G4NistManager::Instance();
  nistManager->FindOrBuildMaterial("G4_Ge");
  G4double a; // mass of a mole;
  G4double z; // z=mean number of protons;
  G4double density;

  // Vacuum
  new G4Material("Galactic", z=1., a=1.01*g/mole,density= universe_mean_density,
                 kStateGas, 2.73*kelvin, 3.e-18*pascal);

  // Print materials
  G4cout << *(G4Material::GetMaterialTable()) << G4endl;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

G4VPhysicalVolume* B4cDetectorConstruction::DefineVolumes()
{
  // Geometry parameters
  fNofLayers = 2;
  G4double innerRadius = 0.*cm;
  G4double outerRadius = 60.9/2*cm;
  G4double hz = 62/4.*cm;
  G4double gap_hz = 0.01*mm;
  G4double startAngle = 0.*deg;
  G4double spanningAngle = 360.*deg;

  auto abs_hz = hz - gap_hz;
  auto worldSizeXY = 10. * outerRadius;
  auto worldSizeZ = 10.* hz *2.;


  // Get materials
  auto defaultMaterial = G4Material::GetMaterial("Galactic");
  auto absorberMaterial = G4Material::GetMaterial("G4_Ge");
  auto gapMaterial = G4Material::GetMaterial("G4_Ge");


  //
  // World
  //
  auto worldS
    = new G4Box("World", // its name
                worldSizeXY/2, worldSizeXY/2, worldSizeZ/2); // its size

  auto worldLV
    = new G4LogicalVolume(
                worldS, // its solid
                defaultMaterial, // its material
```

```
              "World"); // its name

auto worldPV
  = new G4PVPlacement(
              0, // no rotation
              G4ThreeVector(), // at (0,0,0)
              worldLV, // its logical volume
              "World", // its name
              0, // its mother volume
              false, // no boolean operation
              0, // copy number
              fCheckOverlaps); // checking overlaps

//
// Calorimeter
//
auto calorimeterS
  = new G4Tubs("Calorimeter",
              innerRadius,
              outerRadius,
              hz,
              startAngle,
              spanningAngle);
auto calorLV
  = new G4LogicalVolume(
              calorimeterS, // its solid
              defaultMaterial, // its material
              "Calorimeter"); // its name

new G4PVPlacement(
              0, // no rotation
              G4ThreeVector(), // at (0,0,0)
              calorLV, // its logical volume
              "Calorimeter", // its name
              worldLV, // its mother volume
              false, // no boolean operation
              0, // copy number
              fCheckOverlaps); // checking overlaps

//
// Layer
//
auto layerS
  = new G4Tubs("Layer",
              innerRadius,
              outerRadius,
              hz,
              startAngle,
              spanningAngle);
auto layerLV
  = new G4LogicalVolume(
              layerS, // its solid
              defaultMaterial, // its material
              "Layer"); // its name
```

```
new G4PVReplica(
             "Layer", // its name
              layerLV, // its logical volume
              calorLV, // its mother
              kZAxis, // axis of replication
              fNofLayers, // number of replica
              hz); // witdth of replica

//
// Absorber
//
auto absorberS
  = new G4Tubs("Abso",
              innerRadius,
              outerRadius,
              abs_hz,
              startAngle,
              spanningAngle);
auto absorberLV
  = new G4LogicalVolume(
              absorberS, // its solid
              absorberMaterial, // its material
              "AbsoLV"); // its name

 new G4PVPlacement(
              0, // no rotation
              G4ThreeVector(0., 0., -gap_hz/2), // its position
              absorberLV, // its logical volume
              "Abso", // its name
              layerLV, // its mother volume
              false, // no boolean operation
              0, // copy number
              fCheckOverlaps); // checking overlaps

//
// Gap
//
auto gapS
  = new G4Tubs("Gap",
              innerRadius,
              outerRadius,
              gap_hz,
              startAngle,
              spanningAngle);
auto gapLV
  = new G4LogicalVolume(
              gapS, // its solid
              gapMaterial, // its material
              "GapLV"); // its name

new G4PVPlacement(
              0, // no rotation
              G4ThreeVector(0., 0., abs_hz/2), // its position
```

```
                gapLV, // its logical volume
                "Gap", // its name
                layerLV, // its mother volume
                false, // no boolean operation
                0, // copy number
                fCheckOverlaps); // checking overlaps

  //
  // print parameters
  //
  // Visualization attributes
  //
  worldLV->SetVisAttributes (G4VisAttributes::GetInvisible());

  auto simpleBoxVisAtt= new G4VisAttributes(G4Colour(1.0,1.0,1.0));
  simpleBoxVisAtt->SetVisibility(true);
  calorLV->SetVisAttributes(simpleBoxVisAtt);

  //
  // Always return the physical World
  //
  return worldPV;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cDetectorConstruction::ConstructSDandField()
{
  // G4SDManager::GetSDMpointer()->SetVerboseLevel(1);

  //
  // Sensitive detectors
  //
  auto absoSD
    = new B4cCalorimeterSD("AbsorberSD", "AbsorberHitsCollection", fNofLayers);
  G4SDManager::GetSDMpointer()->AddNewDetector(absoSD);
  SetSensitiveDetector("AbsoLV",absoSD);

  auto gapSD
    = new B4cCalorimeterSD("GapSD", "GapHitsCollection", fNofLayers);
  G4SDManager::GetSDMpointer()->AddNewDetector(gapSD);
  SetSensitiveDetector("GapLV",gapSD);

  //
  // Magnetic field
  //
  // Create global magnetic field messenger.
  // Uniform magnetic field is then created automatically if
  // the field value is not zero.
  G4ThreeVector fieldValue;
  fMagFieldMessenger = new G4GlobalMagFieldMessenger(fieldValue);
  fMagFieldMessenger->SetVerboseLevel(1);

  // Register the field messenger for deleting
```

```
  G4AutoDelete::Register(fMagFieldMessenger);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

B4cEventAction.cc

```
#include "B4cEventAction.hh"
#include "B4cCalorimeterSD.hh"
#include "B4cCalorHit.hh"
#include "B4Analysis.hh"

#include "G4RunManager.hh"
#include "G4Event.hh"
#include "G4SDManager.hh"
#include "G4HCofThisEvent.hh"
#include "G4UnitsTable.hh"

#include "Randomize.hh"
#include <iomanip>

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cEventAction::B4cEventAction()
 : G4UserEventAction(),
   fAbsHCID(-1),
   fGapHCID(-1)
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cEventAction::~B4cEventAction()
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

B4cCalorHitsCollection*
B4cEventAction::GetHitsCollection(G4int hcID,
                                  const G4Event* event) const
{
  auto hitsCollection
    = static_cast<B4cCalorHitsCollection*>(
        event->GetHCofThisEvent()->GetHC(hcID));

  if ( ! hitsCollection ) {
    G4ExceptionDescription msg;
    msg << "Cannot␣access␣hitsCollection␣ID␣" << hcID;
    G4Exception("B4cEventAction::GetHitsCollection()",
      "MyCode0003", FatalException, msg);
  }

  return hitsCollection;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cEventAction::PrintEventStatistics(
                             G4double absoEdep, G4double absoTrackLength,
```

```
                          G4double gapEdep, G4double gapTrackLength) const
{
  // print event statistics
  G4cout
     << "␣␣␣Absorber:␣total␣energy:␣"
     << std::setw(7) << G4BestUnit(absoEdep, "Energy")
     << "␣␣␣␣␣␣␣total␣track␣length:␣"
     << std::setw(7) << G4BestUnit(absoTrackLength, "Length")
     << G4endl
     << "␣␣␣␣␣␣␣␣Gap:␣total␣energy:␣"
     << std::setw(7) << G4BestUnit(gapEdep, "Energy")
     << "␣␣␣␣␣␣␣total␣track␣length:␣"
     << std::setw(7) << G4BestUnit(gapTrackLength, "Length")
     << G4endl;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cEventAction::BeginOfEventAction(const G4Event* /*event*/)
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void B4cEventAction::EndOfEventAction(const G4Event* event)
{
  // Get hits collections IDs (only once)
  if ( fAbsHCID == -1 ) {
    fAbsHCID
      = G4SDManager::GetSDMpointer()->GetCollectionID("AbsorberHitsCollection");
    fGapHCID
      = G4SDManager::GetSDMpointer()->GetCollectionID("GapHitsCollection");
  }

  // Get hits collections
  auto absoHC = GetHitsCollection(fAbsHCID, event);
  auto gapHC = GetHitsCollection(fGapHCID, event);

  // Get hit with total values
  auto absoHit = (*absoHC)[absoHC->entries()-1];
  auto gapHit = (*gapHC)[gapHC->entries()-1];

  // Print per event (modulo n)
  //
  auto eventID = event->GetEventID();
  auto printModulo = G4RunManager::GetRunManager()->GetPrintProgress();
  if ( ( printModulo > 0 ) && ( eventID % printModulo == 0 ) ) {
    G4cout << "--->␣End␣of␣event:␣" << eventID << G4endl;

    PrintEventStatistics(
      absoHit->GetEdep(), absoHit->GetTrackLength(),
      gapHit->GetEdep(), gapHit->GetTrackLength());
  }

  // Fill histograms, ntuple
```

```
  //

  // get analysis manager
  auto analysisManager = G4AnalysisManager::Instance();

  // fill histograms
  analysisManager->FillH1(0, absoHit->GetEdep());
  analysisManager->FillH1(1, gapHit->GetEdep());
  analysisManager->FillH1(2, absoHit->GetTrackLength());
  analysisManager->FillH1(3, gapHit->GetTrackLength());

  // fill ntuple
  analysisManager->FillNtupleDColumn(0, absoHit->GetEdep());
  analysisManager->FillNtupleDColumn(1, gapHit->GetEdep());
  analysisManager->FillNtupleDColumn(2, absoHit->GetTrackLength());
  analysisManager->FillNtupleDColumn(3, gapHit->GetTrackLength());
  analysisManager->AddNtupleRow();
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

Macro File

***run1.mac* - Macro for point-like emission**

```
/vis/viewer/set/viewpointThetaPhi 45 270
/gps/position 0 0 36 cm
/gps/particle gamma
/gps/ang/type iso
/gps/ene/type Gauss
/gps/ene/mono 662.2 keV
/gps/ene/sigma 2.4 keV
/run/beamOn 5000
```

***run2.mac* - Macro for cylindrical Marinelli emission (requires alterations to the source and header files above).**

```
/vis/viewer/set/viewpointThetaPhi 45 270
/gps/pos/type Volume
/gps/pos/shape Cylinder
/gps/pos/centre 0 0 36 cm
/gps/pos/radius 30 cm
/gps/pos/halfz 5 cm
/gps/ang/type iso
gps/ene/type Gauss
/gps/ene/mono 662.2 keV
/gps/ene/sigma 2.4 keV
/run/beamOn 5000
```

Plot Energy Histograms

plotNtuple.C

```
{
  gROOT->Reset();
  gROOT->SetStyle("Plain");

  // Open file filled by Geant4 simulation
  TFile f("B4.root");

  // Create a canvas and divide it into 2x2 pads
  TCanvas* c1 = new TCanvas("c1", "", 20, 20, 1000, 1000);
  c1->Divide(2,2);

  // Get ntuple
  TNtuple* ntuple = (TNtuple*)f.Get("B4");

  // Draw Eabs histogram in the pad 1
  c1->cd(1);
  ntuple->Draw("Eabs");

  // Draw Labs histogram in the pad 2
  c1->cd(2);
```

```
  ntuple->Draw("Labs");

}
```

Appendix F: Risk Assessment

Department: Physics and Astronomy — Risk Assessment Form

WORK/PROJECT TITLE: Nuclear Forensics with Gamma-Ray Spectroscopy

LOCATION(S): Physics Department Lab 3

DESCRIPTION OF WORK: Measurement of the gamma rays emitted by a range of samples to identify radionuclides.

PERSONS INVOLVED: Pip Benjamin, Kanishk Ganga, Ira Sokar, John Hei Wong, Sarah Gao, Ke Ma, Lukas Kubin, Yichen Tian

HAZARD IDENTIFICATION *(state the hazards involved in the work)* Consider **Chemicals** (an additional assessment will also be needed), the **environment**, **equipment**, **manual handling**, **electrical equipment**, **fire** and **explosion**, **disposal of waste**.

Use of cryogenic liquids - very low temperatures can lead to frostbite and damages of human skin

Use of gamma-ray sources (closed) - ionizing radiation damage to human body

Cobalt-60 external exposure for Betas, electrons ($4.12 * 10^{-6}\ mSV\,h^{-1}$) and Gammas, X-rays ($1.26 * 10^{-6}\ mSV\,h^{-1}$)

Ceaium-137 external exposure for Betas, electrons ($1.62 * 10^{-3}\ mSV\,h^{-1}$) and Gammas, X-rays ($8.14 * 10^{-6}\ mSV\,h^{-1}$)

High voltages - improper handling of cables may lead to electrocution; or starting a fire

RISK ASSESSMENT *(assess the risks involved in the work and state high, medium or low risk)*

Low Risk - Keeping liquid nitrogen in a container to maintain the detector in a critical temperature

Low Risk - Use of sealed radioactive source

Low Risk - Use of laboratory standard high voltage power supply

CONTROL MEASURES *(say how you will reduce the risk to an acceptable level.)*

Liquid nitrogen will be handled by lab demonstrators and will not be directly accessed during the experiment.

Only very weak radioactive sources will be used, with doses below thresholds for public exposure. Only trained staff wearing dosimeters will move sources to/from the experiment.

Ends of cables will not be handled. If needed, lab demonstrators will set up this aspect of the equipment and ensure temperatures and voltages are within safe limits.

Departmental Safety procedures will be followed.

DECLARATION

I the undersigned have assessed the work, titled above, and declare that there is no significant risk / the risks will be controlled by the methods stated on this form *(delete as applicable)* and that the work will be carried out in accordance with Departmental codes of practice.

Name……...

Signed…… Date…….

Supervisor……

Signed…… Date…….

Appendix G: Group Completion Certificates

UCL Certificate of Achievement

Ira Shokar

has satisfactorily completed the course

Principles of Laboratory Safety

UCL Safety Services — January 20, 2020

Date

Generated by UCL Moodle

UCL Certificate of Achievement

John Wong

has satisfactorily completed the course

Principles of Laboratory Safety

UCL Safety Services — January 17, 2020

Date

Generated by UCL Moodle

UCL Certificate of Achievement

Kanishk Ganga

has satisfactorily completed the course

Principles of Laboratory Safety

UCL Safety Services

Date

Generated by UCL Moodle

UCL Certificate of Achievement

Mark Ma

has satisfactorily completed the course

Principles of Laboratory Safety

UCL Safety Services

Date

Generated by UCL Moodle

UCL Certificate of Achievement

Lukas Kubin

has satisfactorily completed the course

Principles of Laboratory Safety

UCL Safety Services

Date

Generated by UCL Moodle

UCL Certificate of Achievement

Pip Benjamin

has satisfactorily completed the course

Principles of Laboratory Safety

UCL Safety Services

January 23, 2020

Date

Generated by UCL Moodle

UCL Certificate of Achievement

Sirui Gao

has satisfactorily completed the course

Principles of Laboratory Safety

UCL Safety Services

January 17, 2020

Date

Generated by UCL Moodle

UCL Certificate of Achievement

Yichen Tian

has satisfactorily completed the course

Principles of Laboratory Safety

UCL Safety Services

January 16, 2020

Date

Generated by UCL Moodle

Appendix H: Gantt Chart