

02_AbsoluteDeterminationOfSourceActivity_final

October 9, 2025

```

<p style="text-align: center;"><strong>Welcome to PHYS5036. </strong>
```

Detection and Analysis of Ionising Radiation

```
[ ]: # Importing the required modules
```

```
import matplotlib.pyplot as plt          # Common python package for plotting
import numpy as np                       # For many numerical operation (e.g.
    ↳ arrays, sine functions, maths constants)
import pandas as pd                     # Common python dataframe package used
    ↳ in industry and academia
from scipy.optimize import curve_fit     # Commonly used fitting package - has
    ↳ some tricky syntax but useful to learn to use.
import os                               # For file path operations
```

```
[ ]: %pwd
```

1 Attenuation of Gamma-Radiation in Matter

Data collection steps are typically indicated with a blue title.

Computational work is indicated with a red title.

The purpose of this experiment is to familiarise the student with the calibration of a scintillation counter by observing gamma-ray spectra of several radioisotopes. This data will then be analysed and a Monte Carlo simulation utilised to determine the factors describing gamma-ray attenuation in aluminium and lead.

Absolute Determination Objectives: - Produce γ -ray spectra using a scintillation counter coupled to a multichannel analyser (MCA). - Produce a calibration for a scintillation counter, taking into account any experimental uncertainties. - Measure and identify unknown γ -emitting isotopes, taking the performance characteristics and any uncertainties evaluated into account. - Calculate the relationship between energy resolution and the γ -ray energy for a scintillation counter. - Use a Monte Carlo simulation to better understand detector systems and particle interactions. - Combine simulation with experimental results to determinate the source activity.

Hints: 1. Save your data from K-Spect as .Spe file format. 2. All the data-taking for this experiment can be performed in two lab days with proper planning. 3. The code given in this notebook may be incomplete. Make sure you understand the limits and what you are expecting.

1.1 Safety in the lab

The experiment makes use of a scintillation counter, various radioactive sources and unvarnished lead bricks used for shielding.

Handling radioactive sources

- Any source removed from the safe **must** be done by one of the demonstrators. You are required to sign out any source and **you** are responsible for it until it is returned to the safe.
- Do not “lend” or give your source to a colleague when you are finished with it. Notify a demonstrator who can sign the source back in so it is available for anyone to use.
- Keep exposure to a minimum by handling sources quickly, thinking before handling the source and keeping sources as far away from you and anyone else as much as possible.
 - If the source is not being used for a short time, the source should be put behind lead shielding provided.
 - If the source is not being used for an extended time, it should be returned to the safe. This should be done by a demonstrator who can confirm the source is correctly returned.
- Sources are wrapped, placed inside containers or have a protective shielding around them. **Do not tamper with the containers and shielding.**
- Replace the pen sources into the shielding container when not in use.

Operating electronics Under no circumstances should the HV cables be removed without the lab head present. Make sure you are happy with the setting before turning on the crate and the HV. Ensure you switch the equipment off properly at the end of the day.

Handling Lead Bricks They are heavy and can cause significant damage if dropped. Do not handle the lead bricks with bare hands. Nitrile (blue rubber) gloves are provided for handling lead bricks. Wash hands thoroughly after contact with lead.

General Rules Eating and drinking is strictly prohibited in the radiation laboratory. Go into the corridor or stairwell if you need a snack or a drink.

Further safety guidance can be found in the Radioactive Sources in Teaching Labs located on top of the source safe. Please also adhere to the general and specific lab safety guidelines found elsewhere.

1.2 Section 1 - Background Theory

A Scintillation Detector operates by absorbing energy from fast-moving charged particles and generating a large number of visible and ultraviolet photons. The number of photons generated is a measure of the energy absorbed in the detector. The scintillation material is optically coupled to a photomultiplier tube. Low-energy photons generated in the scintillation material strike a photocathode at the entrance to the photomultiplier tube where they eject electrons by the photoelectric effect. These electrons are accelerated in a high potential towards a series of dynodes. At each dynode, the incident electrons release more electrons (about 3 to 5 each) and so the electron flux

multiplies. After 8 to 12 dynodes (depending on photomultiplier type) the avalanche of electrons is detectable as a small negative electrical pulse. This is then amplified to give a typical pulse profile as shown below. The main pulse (negative) is followed by a positive overshoot. It is important to analyse the first portion of the pulse.

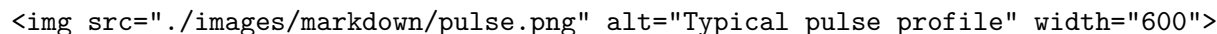
Typical pulse profile for a Sodium Iodide detector.

Figure 1: Typical pulse profile for a Sodium Iodide detector.

1.2.1 1.1. Gamma-ray Interactions:

In this experiment the Scintillation Detector is used to detect gamma-rays emitted by a radioactive source. There are three primary processes by which the gamma-rays can interact with the scintillation material:

1. Photoelectric Effect
2. Compton Scattering
3. Pair Production (for gamma rays of energy >1.02 MeV)

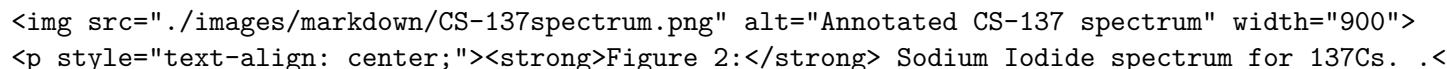
Compton Scattering: Compton Scattering is the purely kinematic collision of a gamma-ray and a loosely bound electron. The energies of the gamma-ray ($E_{\gamma'}$) and the electron after the collision depend only on the initial gamma-ray energy (E_{γ}) and the angle of deflection (θ) of the gamma ray:

$$E_{\gamma'} = \frac{E_{\gamma}}{1 + \frac{E_{\gamma}}{m_e c^2} (1 - \cos \theta)}$$

where m_e is the electron mass and c is the velocity of light. Thus, the maximum energy that can be transferred to the electron occurs when the photon is deflected through 180° .

Energy Deposited in the Scintillator: The amount of energy deposited in the scintillator depends on the interactions of the gamma-ray with the scintillator. The three most distinct features, observed in typical energy spectra, are: 1. All the energy of the incident gamma-ray is absorbed by the detector. This produces the Total Energy Peak (also called the Photo Peak) in a spectrum.

2. The gamma-ray Compton Scatters in the scintillator, and the scattered gamma-ray escapes from the crystal without any further interaction. In this case only the energy imparted to the Compton electron is absorbed. This gives rise to a distribution of energies up to the maximum allowed electron energy. The corresponding cut-off is called the Compton Edge because it is an edge, it is less easy to define its position than the Total Energy Peak.
3. The gamma-ray may Compton Scatter outside the scintillator and the scattered gamma-ray may then enter the scintillator and interact. This gives rise to a small Backscatter Peak where the energy of the photon corresponds to a deflection of 130° to 180° . It generally has very low intensity and sits on top of the Compton distribution. Its position is even harder to define experimentally than the Compton Edge.

Sodium Iodide spectrum for ^{137}Cs .

1.3 Section 2: Digitisation of the Detector Output Signal

The raw output of the PMT or Germanium detector is an electrical pulse, the height of which is dependent on the total energy deposited in the detector, while its shape is related to the time profile of the detector material. In order to build up a spectrum of the energy deposited in the detector, the PMT output is usually amplified and then the pulse height digitised in an Analogue-to-Digital Converter (ADC).

Before beginning, check that your station is setup as follows: - The signal from the scintillation detector to the input of the amplifier and the output of the amplifier to the oscilloscope. - The amplifier should be set to Negative-input used on Unipolar mode. - The operating voltage of each scintillation detector is different, but will be in the range 700V to 800V. - Turn on the crate, HV and launch the KSpect Software.

1.3.1 2.1. Single-Channel Analyser (SCA)

The Output from the scope should now be split: one cable should be connected to the oscilloscope to display the pulse profile, while the other should be connected to the Input of the SCA.

Place the mixed (90Sr, 137Cs, 241Am) source in front of the detector, and observe the pulse profile on the oscilloscope.

Alter the amplifier gain until the maximum pulse amplitude observed is less than 10V (8V recommended), the largest voltage the single channel analyser can discriminate. If there is insufficient range in the amplifier controls to obtain the correct pulse amplitude you may need to make some fine adjustments to the high voltage supply. **ASK A DEMONSTRATOR BEFORE ADJUSTING THE VOLTAGE.**

Set the Single Channel Analyser to **window** mode and set the window size knob to 10; this corresponds to an interval setting of 1V. Make sure the knob on the counter labelled discriminator is on its minimum setting of 0.1V. Obtain an energy spectrum of Cs-137 by stepping the lower level of the voltage window from 0V to 9V in steps of 1V. Record the number of pulses in the selected range for a given measurement time using the electronic counter.

Repeat the measurements, this time reducing the window size to 0.5V (5 on the knob) and stepping the lower voltage by 0.5V for each reading from 0V to 9.5V (this will require double the measurements).

Produce histograms of both sets of results and compare the two spectra.

Discuss with peer(s) or staff your thoughts on the following: >- How does changing the amplifier gain affect the pulse-height distribution you observe? >- Why is it important to keep the maximum pulse amplitude below 10 V? >- What differences do you observe between the 1V-window spectrum and the 0.5V-window spectrum? Which one gives finer energy resolution, and why? >- What features of the Cs-137 energy spectrum can you identify (e.g., photopeak, Compton edge)? >- How might electronic noise influence the lowest voltage bins, and how could you distinguish this from genuine low-energy events? >- What physical processes in the scintillation crystal produce the different parts of the spectrum (photoelectric absorption, Compton scattering, backscattering)? >- If you increased the measurement time, how would that affect the shape and statistical quality of the spectrum?

1.3.2 2.2. Multi-Channel Analyser

The hardware set-up contains a Kromek multi-channel analyser (MCA) with an analogue to digital converter (ADC) that sorts input pulses from the scintillation counter by their amplitudes into 4096 channels (or bins). The MCA used in this experiment is capable of sorting pulses up to 2.4V.

The K-Spect software is used to control the hardware and process this channel data. Ensure in the detector settings that the pulse is set to positive.

Start by inserting the Sr-90, Am-241, Cs-137 mixed source into the source holder in front of the detector.

After a few seconds, you should see tall amplified noise counts in the 0-100 channel range. Remove this by increasing the Lower Level Discriminator (LLD) to just above that feature (typically 50–120 channels, detector-dependent). Ask for help if you need to cut off more than this.

Clear your measurement and a sharp peak corresponding to the 662 keV photopeak of Cs-137 should now be visible. To keep all sources for this experiment within the available channel range, align this peak to roughly channel 1700 by increasing or decreasing the gain from the amplifier in the crate. Read the exact channel by clicking the spectrum and checking the Marker value.

ASK A DEMONSTRATOR BEFORE ADJUSTING THE VOLTAGE

Clear and restart the acquisition to see the effect of any changes. Once you are satisfied, do not change the high voltage or amplifier gain further for the remainder of the experiment.

Now obtain a spectrum for Cs-137 and save the file in ASCII SPE format. If you are using this notebook, upload to JupyterHub to an appropriately named folder.

All measurements of this type are affected by background signals due to factors such as electronic noise and cosmic radiation. To obtain a true energy spectrum arising from just the source radiation, you will also need a background measurement taken the same day as your source data. The background spectrum will be subtracted from each source spectrum.

Acquisition time and Poisson statistics. Radiation counts follow Poisson statistics. The fractional (relative) statistical uncertainty of ε on a measured peak area, can be described as

$$\varepsilon \approx \frac{1}{\sqrt{N}}$$

where N is the net counts in the peak after background subtraction. So in order to get a 10% measurement, for example, we need $N = 100$ counts.

General tips: - Pick a target fractional uncertainty for key measurements - If background is non-negligible, aim for more counts than this - Ensure background measurement will be at least as long as each source run, longer will help reduce background uncertainties. - You can estimate counts using the peak, or use the region-of-interest (ROI) feature in the acquisition software to check the integrated peak counts in that region.

1.3.3 2.3. Analysis introduction

Below you will find a function made to read in the .spe files and return a data frame for use in analysis. Please read through it in order to understand its function, but you are not expected to edit this.

```
[ ]: ## The code below extracts the data from your .spe files and returns it as a
      ↳pandas dataframe.
## you are not expected to edit this - but feel free to make changes if you are
      ↳confident.

def extract_data(filename):
    """Extracts channel range, counts, and measurement times from .spe file"""
    start = end = None
    counts = None
    livetime = realtime = None

    with open(filename, "r") as f:
        lines = f.readlines()
    i = 0
    while i < len(lines):
        line = lines[i].strip()
        if line == "$MEAS_TIM:":
            # Extract measurement times (from header. Syntax: livetime,
            ↳realtime)
            livetime, realtime = map(float, lines[i+1].split())
        elif line == "$DATA:":
            # Extract channel start and end (from header)
            start, end = map(int, lines[i+1].split())
            nochannels = end - start + 1
            # Read channel values
            block = lines[i+2:i+2+nochannels]
            counts = np.array([int(x.strip()) for x in block], dtype=int)
            i += 1
    return start, end, counts, livetime, realtime

def read_spe_files(data_file, background_file, bin_num=4096):
    # Extract data and background
    data_start, data_end, yield1, data_livetime, data_realtime =
    ↳extract_data(data_file)

    if background_file is None:
        # No background file provided - return data only
        print("No background file provided - returning data only. Please ensure
        ↳you have accounted for background in your analysis!")
        channel = np.arange(data_start, data_end + 1, dtype=int)
        error1 = np.sqrt(yield1.astype(float))
        df = pd.DataFrame({
            "channel": channel,
            "data": yield1,
            "err_data": error1,
        })
```

```

    return df

    else:
        bg_start, bg_end, yield2, bg_livetime, bg_realtime = extract_data(background_file)

        # check that they are the same length
        if (data_start, data_end) != (bg_start, bg_end):
            raise ValueError("data and background files have different channel_
#ranges!")

        # Background-subtracted counts (scaled by uptime)
        yield3 = yield1 - yield2*(data_livetime/bg_livetime)

        # Channels
        channel = np.arange(data_start, data_end + 1, dtype=int)

        # Poisson errors
        error1 = np.sqrt(yield1.astype(float))
        error2 = np.sqrt(yield2.astype(float))
        error3 = np.sqrt(yield1 + yield2*(data_realtime/bg_realtime))

        # Assemble into a DataFrame
        df = pd.DataFrame({
            "channel": channel,
            "data": yield1,
            "err_data": error1,
            "background": yield2,
            "err_background": error2,
            "data_bgsbtracted": yield3,
            "err_data_bgsbtracted": error3,
        })

        # Rebin if requested. I.e. if bin_number = 512, rebin to 512 channels
        if bin_num is not None:
            if bin_num not in [512, 1024, 2048, 4096, 8192, 16384]:
                raise ValueError("bin_num must be one of [512, 1024, 2048, #
#4096, 8192, 16384]")
            if (data_end - data_start + 1) % bin_num != 0:
                raise ValueError(f"Cannot rebin to {bin_num} channels - #
#original number of channels is {data_end - data_start + 1}, which is not a #
#multiple of {bin_num}")
            rebin_factor = (data_end - data_start + 1) // bin_num
            df_rebinned = df.groupby(df.index // rebin_factor).sum().
#reset_index(drop=True)
            df_rebinned["channel"] = np.arange(bin_num)
            df = df_rebinned

```

```
return df
```

Upload your saved Caesium-137 and background measurements to an appropriate folder and edit the code below.

The code below shows how to read in the files with the above function (`read_spe_files`) and some simple analysis methods to get your started.

Edit the code below to plot the Cs-137 data you just recorded.

```
[ ]: # example of using the read in function

# print current folder from os
print(f"Current working folder: {os.getcwd()}, make sure to set path relative_
↳to this")

# set file paths here to YOUR OWN files
data_file = "cs137mix.spe"
background_file = "background.spe"

# define our pandas data frame by reading in the spe files
df_source = read_spe_files(data_file, background_file)

# show a few lines of the dataframe to check it looks OK and view the column_
↳names
df_source[40:50]
```

```
[ ]: # Simple plot of the data and how to select a subset
plt.plot(df_source["channel"],df_source["data_bgsubtracted"])
plt.plot(df_source["channel"][1100:1500],df_source["data_bgsubtracted"][1100:
↳1500]) # only plot channels 1200:1999
plt.show()

# Alternate way using variables to define the range with some extra features
# Plotting just chosen range, but with error bars
min, max = 1450, 2000
plt.errorbar(df_source["channel"][min:max], df_source["data_bgsubtracted"][min:
↳max], yerr=df_source["err_data_bgsubtracted"][min:max], fmt='o',_
↳label='Subset with error bars')
plt.xlabel('Channel')
plt.ylim(0, 500)
plt.legend()
plt.show()
```

Discuss with peer(s) or staff your thoughts on the following: >- Can you name the features shown in the spectrum? >- Is the data taking up the full channel range, if not why?

Now you need to obtain the location of the photopeak energy. At this point, consider the shape

of the photopeak and how you can use this knowledge to fit an appropriate model and extract its parameters. Note that the uncertainty on the peak position is required. Ask the demonstrators if you are unsure and need some pointers.

2.3.1 Energy Calibration data In order to calibrate the detectors, we will need to find the relationship between the known energy of the different gamma-ray photopeaks and the corresponding measured ADC channel.

The aim of this section is to performing fits to the main photopeaks and fill in the table below. Some of the values in this table will be needed later in the analysis.

1.3.4 Photopeak data

Source	Peak channel	Error (ch)	Peak (ch)	Error	Integral (counts)	Error
Am-241						
Co-60						
Co-60						
Cs-137						
Cs-137						
Na-22						
Na-22						

To do this as accurately as possible, the photopeaks must be fitted with a function that describes both the background under the photo peak and the peak itself. Consider the shape of the photopeak and how you can use this knowledge to fit an appropriate model and extract its parameters. Note that the uncertainty on the peak position is required.

Ask the demonstrators if you are unsure and need some pointers.

```
[ ]: # Write a function to model a Gaussian peak + polynomial background. An example
      ↪ of writing a fit function is shown below for a polynomial background.
```

```
def polybackground(x,const,lin,quad):
    background = const+lin*x+quad*x*x
    return background

def gauss():
    return peak

def signalFit():
    return peak+background

def doubleGaussignalFit():
    return peak1 + peak2 + background
```

```
[ ]: # Plot the photopeak region of Cs-137 and perform a fit to the data here
```

At this stage, check with a demonstrator that your code is correct and that you have obtained a reasonable value for Cs-137.

Now repeat this procedure. Note the shape of the photopeaks of Co-60 and how you should adapt your model to reflect this.

Discuss with peer(s) or staff your thoughts on the following: >- When fitting the photo peak, which fitting function is recommended—Gaussian, Lorentzian, or something else? >- How do you identify multiple peaks in the spectrum if they overlap?

1.3.5 2.3.2 Energy Calibration Fitting

To accurately determine the energy of detected events, the detector readouts (recorded in channel numbers) must be calibrated against known photon energies. This converts spectra from an arbitrary channel scale into a meaningful energy scale.

Using the known photopeak energies in the table below, we will:

1. Plot the relationship between the measured photopeak channel number and the known photopeak energy.
2. Fit several models to this relationship using orthogonal distance regression (ODR), since the uncertainties are primarily in the x-values.
3. Evaluate the quality of each fit by examining residuals and computing the reduced chi-squared (χ^2) value.

Reference Photopeak Energies

Isotope	Photopeak (keV)
Am-241	59.5
Co-60	1172
Co-60	1333
Cs-137	662
Na-22	511
Na-22	1275

Procedure

1. Plot the Data

- Create a scatter plot of energy (keV) versus channel number for your measured photopeaks (include horizontal error bars)
- This visual inspection will help you decide what kind of function might best represent the calibration.
 - How do we get the uncertainty in Energy based on the uncertainty in channel number?

```
[ ]: # Analysis code here
```

2. Fit Several Candidate Models Test multiple functional forms for the calibration curve using ODR.

1. Linear:

$$E(x) = a_0 x + a_1$$

2. Quadratic:

$$E(x) = a_0 x^2 + a_1 x + a_2$$

3. Cubic:

$$E(x) = a_0 x^3 + a_1 x^2 + a_2 x + a_3$$

4. Rational (2/1):

$$E(x) = \frac{a_0 x^2 + a_1 x + a_2}{a_3 x + 1}$$

- x is the channel number measured by the detector.
- $E(x)$ is the predicted energy in keV.
- The parameters a_0, a_1, a_2, a_3 are determined from the fitting procedure.

```
[ ]: # Define your fit functions here
```

1.3.6 Fitting a Straight Line Using ODR

Since the uncertainties are in x , we will use `scipy.odr`, a fitting package designed for errors in both variables.

Below is an example of how to fit a straight line to your data:

```
“python from scipy.odr import ODR, Model, RealData
```

2 Define straight line function for ODR

```
def straightLine(B, x): m, c = B  
return m*x + c
```

3 Prepare the data, including x-errors, you can add y-errors with `sy`

```
data = RealData(Channel, Energy, sx=Channel_Error)
```

4 Create the model

```
model = Model(straightLine)
```

5 Set up and run ODR

```
odr = ODR(data, model, beta0=[1, 1]) # beta0 is your initial guess for slope and intercept output
= odr.run()
```

6 Extract optimal parameters and their uncertainties

```
popt, perr = output.beta, output.sd_beta
print("Optimal parameters:", popt) print("Parameter uncertainties:", perr)
```

3. Assessing Fit Quality: Residuals and Chi-Squared Residuals

To evaluate the quality of your fit, it is helpful to start with a visual inspection of the residuals before moving on to statistical measures.

The residuals are the differences between the measured energies and the energies predicted by the fitting function:

$$\text{Residual}_i = E_i - E_{\text{fit},i}$$

Plotting residuals as a function of channel number can reveal: - Systematic trends suggesting the model does not capture the data fully. - Outliers with unusually large deviations. - Whether the scatter of residuals is consistent with expected experimental uncertainty.

```
[ ]: # Space for analysis code
```

Chi-squared

In order to assess the quality of your fit, we need to define a criterion for what makes a “good” fitting function. The most common statistical measure for evaluating how well a model describes data is the chi-squared (χ^2) statistic.

The chi-squared statistic quantifies the total deviation between your measured values and those predicted by the fitted function, weighted by measurement uncertainties:

$$\chi^2 = \sum_i \left(\frac{E_i - E_{\text{fit},i}}{\sigma_{x,i}} \right)^2$$

where E_i is the measured energy (in keV), $E_{\text{fit},i}$ is the energy predicted by the model, and $\sigma_{x,i}$ is the uncertainty in measured channel position.

A smaller number then is generally better as you predicted energies match the known energies well. However, it depends on how many data points and free parameters are in the model. To compare models with different numbers of parameters, we calculate the reduced chi-squared statistic, defined as

$$\chi^2_{\text{reduced}} = \frac{\chi^2}{N - p}$$

where N is the number of data points, and p is the number of parameters in your fit.

A good fit will generally yield a reduced $\chi^2_{reduced}$ value close to 1: - $\chi^2_{reduced} = 1$. The model describes the data within expected uncertainties. - $\chi^2_{reduced} < 1$. The data scatter is smaller than expected (possibly overestimated errors). - $\chi^2_{reduced} > 1$. The model does not describe the data well or uncertainties are underestimated.

4. Compare and Conclude

- Plot the fitted functions over your data for visual comparison.
- Examine which model yields the smallest residuals, and a $\chi^2_{reduced}$ closest to 1.
- Select and justify the most appropriate function for your energy calibration curve.

Does adding more parameters (e.g., quadratic vs. linear) meaningfully improves the fit, or merely overfits the data.

```
[ ]: # Space for analysis code
```

6.0.1 Estimating Uncertainty on Energy

If the fitted function is:

$$E = f(x, a_0, a_1, \dots, a_n)$$

where x is the measured channel and a_0, a_1, \dots, a_n are the fitted parameters.

The uncertainty in E can be approximated as:

$$\sigma_E^2 = \left(\frac{\partial f}{\partial x} \sigma_x \right)^2 + \sum_i \left(\frac{\partial f}{\partial a_i} \sigma_{a_i} \right)^2 + \dots$$

Where: - $\frac{\partial f}{\partial x}$ is the partial derivative of the function with respect to the measured channel.
- $\frac{\partial f}{\partial a_i}$ is the partial derivative with respect to each fitted parameter.
- σ_x is the uncertainty in the channel measurement.
- σ_{a_i} is the uncertainty in the fitted parameter a_i (obtained from ODR).

Convert channel to energy and propagate the uncertainty from your chosen fit.

```
[ ]: # Space for analysis code
```

6.1 Section 3 - Determination of unknown sources

In real-life, measurements like this can be used to help determine the presence of unknown radioactive sources based on their characteristics discussed in this lab.

Using the same method as before: - take measurements for the “unknown” sources labelled A1145 and 1270, - find the channel number of their photopeak (or photopeaks) and - use your energy calibration curve to determine the photopeak energies.

Use the table below to help determine what you think the unknown sources are, justifying your answer. When more than one peak is present in the spectrum, the relative intensities of the photopeaks may provide additional information to help your identification.

Isotope

Half-life (T_{1/2})

-ray Energy (keV)

Relative Intensity (%)

Am-241

433 years

14

13.5

18

21.0

21

5.0

26

2.5

60

35.3

32–38

8.0

Cs-137

30.1 years

662

85.1

Ba-133

10.8 years

30–36

123

80

2.4

81

33.8

276
7.1
303
18.7
356
61.9
384
8.9
Cd-109
453 days
88
100
Co-57
270.5 days
14
9.4
122
85.2
136
11.1
Co-60
5.27 years
1173
99.86
1333
99.98
Na-22
2.60 years
511
181
1275
99.95

Discuss with peer(s) or staff your thoughts on the following: >- How closely do the measured energies need to match tabulated gamma energies to confidently identify a source? >- What should we do if our measured peak energy falls between two known gamma lines? >- Are there common gamma sources that show up as background contaminants in the lab?

[]: `# Analysis code here`

6.2 Section 4 - Resolution of the Detector

At this stage you should compute the Full Width Half Maximum (FWHM) for each of the photopeaks. This can be obtained by multiplying the standard deviation of the photopeak by 2.355 (specifically $2\sqrt{2\ln 2}$). The uncertainty in the FWHM should also be obtained.

The finite width of the observed photopeaks is a result of the same energy deposit in a detector not giving the same pulse height for successive measurements. This is because there is an intrinsic variation in the number of (in this case) optical scintillation photons, subject to the familiar counting statistics rules. The spread is proportional to \sqrt{N} , where N is the total number of optical scintillation photons for a given energy deposit. It is not possible to reliably simulate a detector system without first determining how this resolution varies with the energy of the incident radiation.

The resolution of a photopeak is defined as:

$$R = \frac{\text{FWHM of Peak}}{\text{Peak Value}} = kE^n$$

Where E is the photopeak gamma-ray energy and k, n are constants.

Using your energy calibrated values: - Find the resolution for each of the photopeaks. - Plot your resolution values as a function of photopeak energy. - Using the equation above, determine values for k, n . Consider also if there are any mathematical operations that can be applied to the equation to make the resulting model easier to fit.

Discuss with peer(s) or staff your thoughts on the following: >- What does the extracted value of n tell you about the operation of this type of scintillation detector? >- How does your extracted n compare to the expected value? What does this suggest?

[]: `# Analysis code here`

6.3 Section 5 - Simulation: Activity Determination with a Monte Carlo Simulation

Extracting absolute physical quantities such as activities and cross sections is one of the most difficult type of measurement. This is because absolute physical quantities require the knowledge of the absolute efficiency of the detector system or other measuring apparatus. This efficiency depends on a large number of different factors. >- What do you think some of these factors are?

The only means of determining this efficiency with any accuracy is to have a detailed simulation of the experiment. This must accurately reflect the geometry of the experimental set-up as well as the interactions of the gamma-rays in the different materials present.

A numerical simulation based upon Monte Carlo techniques is therefore employed in this experiment.

Research how these type of numerical methods are used to simulate the interaction of radiation with matter and therefore the response of detectors.

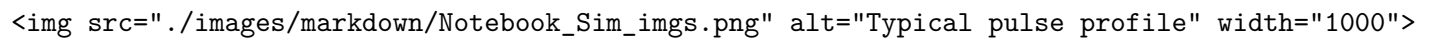
5.1. Getting started: First follow the steps in the “First Computing Set Up Steps” document, found on the Lab Moodle page, under the Lab Material→Computing section. It provides some useful introductory reading for using the School’s computing. Once you are happy with that document, log into JupyterHub in the normal way (remember if you are working remotely, you will need to first connect via VPN).

<https://jupyter.physics.gla.ac.uk/hub/login>

Then follow the simulation environment set up instructions in Section 5 of the “Experiment Computing Set Up” document from on the Lab Moodle page, under the Lab Material→Computing section. If you have any questions during this setup stage speak to the lab staff, who are happy to help you. When we execute the command

```
bash /local/cern/software/Sim/SimSetup.sh
```

during step 5 of the Experiment Computing Set Up, we are telling your Unix environment how to find the relevant simulation code. This is a Geant4 particle transport Monte Carlo simulation called SodiumIodideMC which has been designed specifically for this laboratory. The simulated set up is shown in Figure 3.

 **Figure 3:** Left shows a sodium iodide scintillation detector.

Advanced: For students interested in the Geant4 software details, the C++ class definition files can be found on Moodle in the Computing section, in a folder called “SodiumMC Source Code.” Have a look at the folders and files inside. The class implementation files are in SodiumIodideMC/src/subdirectory. Have a look through these classes if you are curious how the C++ code for a Geant4 simulation is run and ask if you have any questions or if you would like someone to explain it to you.

5.2. Run the simulation The simulation should be run four times to simulate the measurements that were taken in the lab during the energy calibration with the four different radioactive sources.

To set up the simulation, you will need to change the config.json file. This file contains all the parameters that define the simulation, and should look something like this:

```
{
  "sourceName": "Cs137",
  "nEvents": "250000",
  "detectorHolderDistance": "50",
}
```

Note: To edit this json, you may have to right-click -> Open With -> Editor.

The other four parameters used for this experiment are: - `sourceName`: This is the name of the source you are simulating. The options are Am241, Na22, Cs137 and Co60. - `nEvents`: This is the number of decay events you want to simulate. A value of 250,000 is a good compromise between computational speed and statistical quality. - `detectorHolderDistance`: This is the distance **in mm** between the source and the front face of the detector. **You will need to measure this with a ruler and input into the config yourself.**

Once you have set the set your desired simulation parameters, you can run the simulation by running the following commands in the terminal:

```
cd ~/PHYS5036-lab
runSodiumIodide.py
```

Alternatively, if you make more than one config and its not named config.json, you can specify which config to use by running:

```
cd ~/PHYS5036-lab
runSodiumIodide.py <configFileName>.json
```

This will run the simulation and generate two output files (with file format `G4_<detectorType>_<sourceName>.spe/pdf`): - an `.spe` file containing the simulated energy spectrum, and - a `.pdf` file containing a quick-look plot of the simulated spectrum.

Note that a simulated background file is also generated. The output files will be written into the data folder. You can run the simulation elsewhere if you prefer, but please note that the data files will always be created in the same folder which the simulation is executed from.

5.3. Analyse the simulated spectra Once you have ran all four source simulations (it can take a while), plot the simulated spectra for the four sources. You should be able to use the analysis method that you previously used for experimental data for the analysis of simulated data.

Discuss with peer(s) or staff your thoughts on the following: >- How well do they agree with your experimental data? >- Are there any significant differences, and what are the possible reasons for these differences?

5.4. Absolute detector efficiency The absolute efficiency of a detector for a given measurement varies with energy and is given by:

$$\varepsilon = N_{detected}/N_{generated}$$

where $N_{\{detected\}}$ is the integrated number of counts under the simulated photo peak and $N_{generated}$ is the number of radioactive decays (Bq) that were initially simulated. For the simulations you have just ran $N_{generated} = 250,000$.

- Obtain $N_{detected}$ for the six main peaks in the data-set.
- Plot and comment on how the efficiency of the detector varies with known energy.

The corrected absolute activity of the source that you measured in the lab can be extracted from the measured activity (A_m) using the relationship:

$$A_{abs} = \frac{A_m}{\varepsilon}$$

The half-life ($t_{1/2}$) for a radioactive source is related to the decay constant (λ) by the expression:

$$\lambda = \frac{\ln(2)}{t_{1/2}}$$

The activity of the sources were measured in April 2005 and are given in the table below. You can calculate your own value for the 2005 activity (A_{2005}) using the formula:

$$A_{2005} = \frac{A_{abs}}{e^{-\lambda t}}$$

Use the above formula and the data below to calculate the activity for each of the different sources.

Nuclide

Half-Life

Gamma Energies (MeV)

Relative Intensity (%)

Activity (kBq, April 2005)

241-Am

433 y

0.026

2.4

74

0.060

35.9

137-Cs

30.1 y

0.036

8.6

333

0.662

91.4

60-Co

5.27 y

1.173

49.97

74

1.333

50.03

22-Na

2.60 y

0.511

64.4

74

1.275

35.6

[]: *# Analysis code here*

Discuss with peer(s) or staff your thoughts on the following: >- What factors lead to $N_{detected}$ being smaller than $N_{generated}$? >- How does MCA dead time contribute?