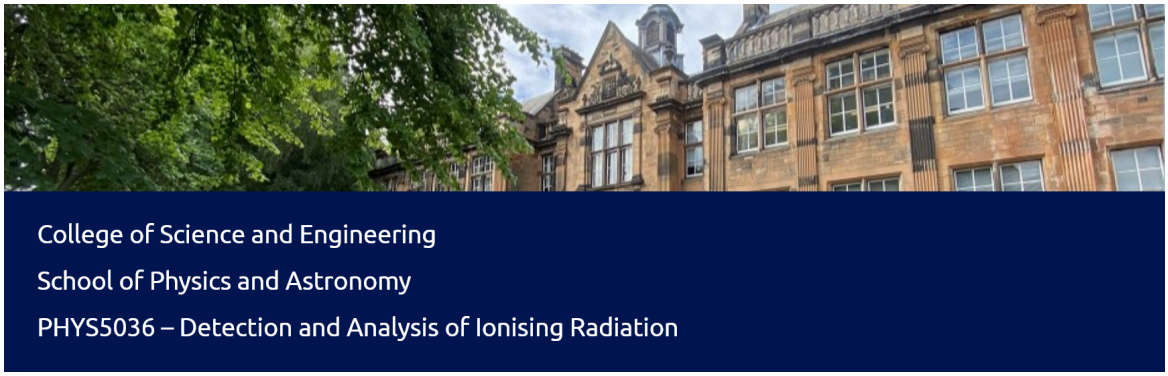

Welcome to PHYS5036. Detection and Analysis of Ionising Radiation

In [ ]:
```
# 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,
import pandas as pd                         # Common python dataframe package used in industry and acade
from scipy.optimize import curve_fit        # Commonly used fitting package - has some tricky syntax but
from scipy.odr import ODR, Model, RealData  # Another fitting package (alternative fitting package that
import os                                   # For getting current working directory
```

In [ ]:
```
%pwd
```

Gamma-ray Spectroscopy with a High Resolution Germanium Detector

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 radiation detection using different types of detector materials and also introduces environmental radiation using mineral samples to determine their radioactive isotope content.

Objectives:

- To calibrate the Sodium Iodide and Germanium Detector
- To compare the resolution of three different detector materials.
- To identify unknown radioactive isotopes within mineral samples.
- To become familiar with the use a Monte Carlo simulation to better understand detector systems and particle interactions.

NOTE: Do not remove wires or switch off power supply when using the germanium detector to avoid damage to the detector crystal. Sealed radioactive sources should remain sealed in their clear plastic bags.

Place all unused radioactive sources behind lead shielding to ensure minimum exposure time and avoid disturbance from the source being recorded. At the end of the experiment, please do not forget to return all radioactive sources to the safe.

Hints:

1. Save your data from Maestro 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.

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 demonstators. 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 a 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.

Section 1 - Background Theory

Scintillation Detector

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.

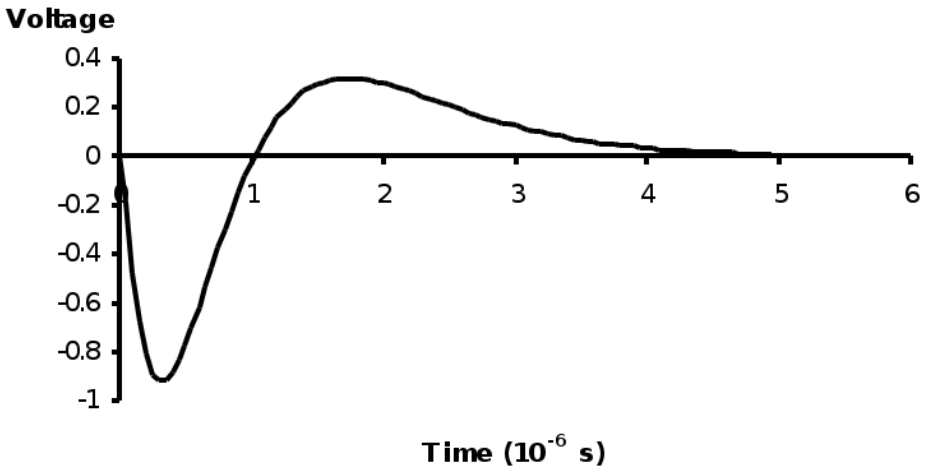

Figure 1: Typical pulse profile for a Sodium Iodide scintillation detector.

High-purity Germanium Detector

A solid-state detector is simply a reverse biased diode. Thus, there is essentially no current flowing in it. However, when radiation is incident on it, the resulting ionisation creates electron-hole pairs which carry current. Since a fixed energy is required to create an electron-hole pair, the number of pairs created is a measure of the energy deposited by the radiation. The amount of current flowing can be measured and used to produce a corresponding voltage peak which is then amplified. The output from the amplifier can be viewed on the oscilloscope to show the pulse profile. A typical pulse profile for a solid-state detector is shown in Figure 2 (note the differences between this and the scintillator pulse profile). Comment on these differences.

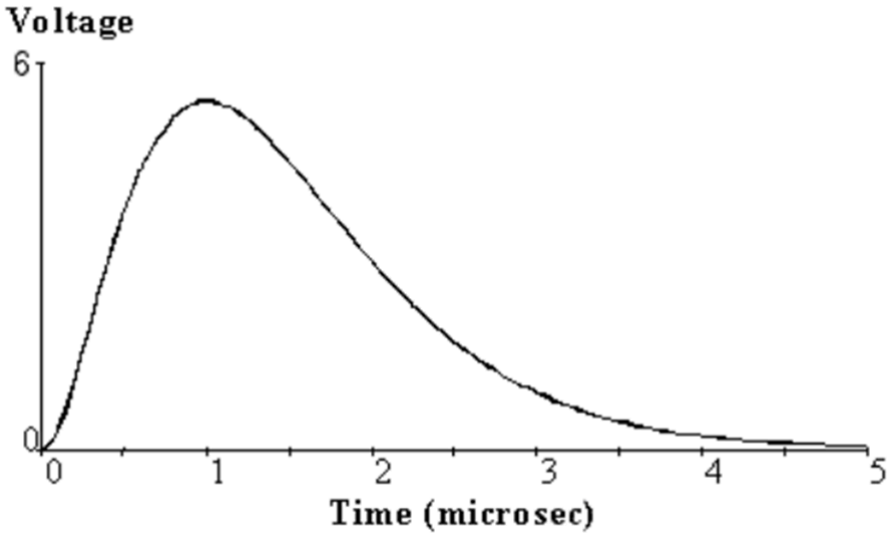

Figure 2: Typical pulse profile for a solid-state detector.

1.1. Oscilloscope testing

The signal output cable from the scintillation detector should be connected to the Input of the amplifier, the Output from which is then 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 MCA?.
- The amplifier should be used on Unipolar mode.
- Turn on the amplifier and the High Voltage (HV) module. The optimum operating voltage of each scintillation detector is different but will be in the range 700V to 1200V.
- Do not exceed this upper value without consulting a demonstrator.
- Place a 137Cs source in front of the detector, and observe the pulse profile on the oscilloscope.

- For future steps, the maximum signal capable of being measured is positive pulses up to 10V amplitude, the amplifier should be adjusted so that the maximum height of the pulses are less than this value.
- Ensure that the pulses are not too large so that the amplifier does not saturate at the top of the peak. (If there is insufficient range in the amplifier controls to obtain the correct pulse amplitude you may need to make some adjustments to the HV supply.)

We do not expect you to be an expert with a scope. Please speak to a demonstrator if you need help with this section.

1.3. 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, which are 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.

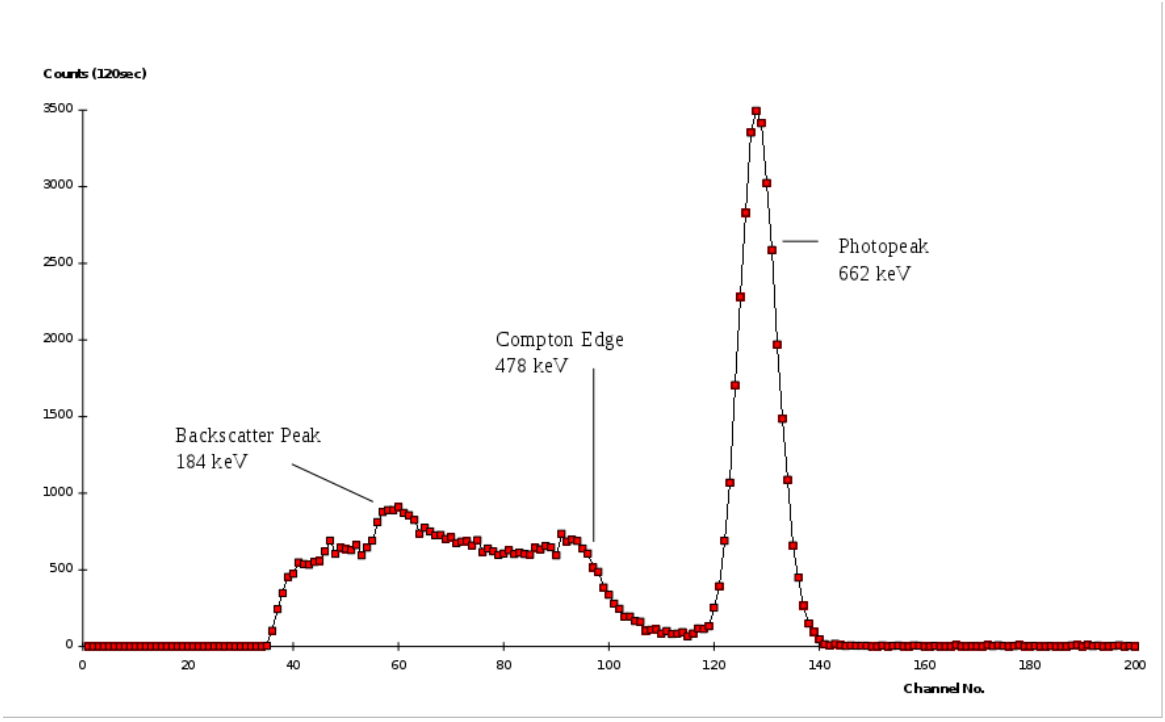

Figure 2: Sodium Iodide spectrum for 137Cs. .

Section 2:

Measurement: Digitisation of the Detector Output Signal

The raw output of the PMT is a negative electrical pulse, the height of which is dependent on the total energy deposited in the scintillator, while its shape is related to the time profile of the scintillator material. 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).

A simple way to appreciate the principle of digitisation is through the use of a Single Channel Analyser (SCA). For full spectral measurements, however, we employ a Multi-Channel Analyser (MCA).

The MCA system is:

- Hardware ADC: sorts input pulses into a selectable number of amplitude channels (256, 512, 1024, or 2048).
- Software (Maestro): controls acquisition and stores spectra.

The ADC accepts positive pulses up to 10 V. For this experiment:

- Use 512 channels with scintillation detectors.
- Use 1024 channels with the Germanium detector.

Note: Separate instructions for Maestro are provided on Moodle — read these before beginning.

2.1.1 Data Collection with the Germanium Detector

A more detailed procedure will follow this section, but please remember to take all of these readings before swapping detectors as your calibration is only valid for the day it was done.

Allow a full lab day for this part.

Procedure Overview: Record spectra:

- Record a spectrum for Cs-137 with the Ge detector.
- Adjust the amplifier so that the photopeak lies around channel 325 of 1024 (this allows us to make use of the full dynamic range of the ADC for all sources in the lab).
- Once set, do not change the amplifier gain for the remainder of the experiment.
- Acquisition time: ~10–15 minutes.
- Save spectra in .Spe format within clearly named folders.

Background correction:

- Repeat the measurement with identical settings and acquisition time, but without a source.
- Save the background file for subtraction.

> - Why do we want to take a background measurement?

Using the same procedure, acquire spectra for:

- Am-241,
- Na-22,
- Co-60
- The Unknown source
- Uranium Oxide sample
- Thorium Oxide sample
- One radioactive mineral sample (e.g. pitchblende)

Always monitor dead time in Maestro. If dead time is high, move the source further from the detector to reduce it.

2.1.2 Data Collection with the Sodium Iodide Scintillation Detector

Allow a full lab day for this part (and Section 2.1.3).

Repeat the measurements listed above with a NaI detector with these changes:

- Set MCA to 512 channels.
- Adjust high voltage to ~1200 V.
- Position the Cs-137 photopeak around channel 215.

2.1.3 Data Collection with the Plastic Scintillation Detector

Set high voltage to ~1200 V.

Acquire three spectra:

- Background
- Radioactive source
- Radioactive mineral sample

Analysis: Energy Calibration

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

To do this as accurately as possible, the photopeaks in the spectra must be fitted with a function that describes both the background under the photo peak and the peak itself.

In the following sectino you will need to perform fits for the main peaks in your data set. Enter your results in the table below as you proceed. These values will be required for later analysis.

Germanium Detector

Source	Expected energy (MeV)	Peak channel	Error (ch)	Peak σ (ch)	Error	Integral (counts)	Error
Am-241							
Co-60	1.173						

Source	Expected energy (MeV)	Peak channel	Error (ch)	Peak σ (ch)	Error	Integral (counts)	Error
Co-60	1.333						
Cs-137	0.662						
Cs-137							
Na-22							
Na-22							

Sodium Iodide (NaI) Detector

Source	Expected energy (MeV)	Peak channel	Error (ch)	Peak σ (ch)	Error	Integral (counts)	Error
Am-241							
Co-60	1.173						
Co-60	1.333						
Cs-137	0.662						
Cs-137							
Na-22							
Na-22							

Plastic Scintillation Detector

Source	Expected energy (MeV)	Peak channel	Error (ch)	Peak σ (ch)	Error	Integral (counts)	Error
Am-241							
Co-60	1.173						
Co-60	1.333						
Cs-137	0.662						
Cs-137							
Na-22							
Na-22							

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-neglibible, aim for more counts than this
- Ensure backrgound 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) feaure in the acquisition software to check the integrated peak counts in that region.

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.

In [ ]:

```
## 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=None):
    # 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 b
        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_bgsubtracted": yield3,
            "err_data_bgsubtracted": 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 {
            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.

In [ ]:
```
# 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]
```

In [ ]:
```
# 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
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
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?

2.3.1 Energy Calibration data of the Germanium Detector

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.

Germanium Detector Photopeak data

Source	Peak channel	Error (ch)	Peak σ (ch)	Error	Integral (counts)	Error
Am-241						

Source	Peak channel	Error (ch)	Peak σ (ch)	Error	Integral (counts)	Error
Co-57						
Co-60						
Co-60						
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.

In [ ]:
```
# Write a function to model a Gaussian peak + polynomial background. An example of writing a fit functio

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

def peakfit():
    return peak

def signalFit():
    return peak+background

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

In [ ]:
```
# 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?

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

Isotope	Photopeak (keV)
Co-57	122
Co-60	1172
Co-60	1333
Cs-137	662
Na-22	511
Na-22	1275

Remember to also take spectrum of the other materials as detailed in Section 2.1.

- The Unknown source
- Uranium Oxide sample
- Thorium Oxide sample
- One radioactive mineral sample (e.g. pitchblende)

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?

In [ ]:
```
# 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.

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

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:

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

# Define straight line function for ODR
```

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

    return m*x + c

# Prepare the data, including x-errors, you can add y-errors with sy
data = RealData(Channel, Energy, sx=Channel_Error)

# Create the model
model = Model(straightLine)

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

# 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.

In [ ]:
```
# 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), $E_{fit,i}$ is the energy predicted by the model, and $\sigma_{x,i}$ is the uncertainty in meausred 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

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

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?

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

Estimating Uncertainty on Energy

If the fitted function is:

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

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

The uncertainty in E can be approximated as:

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

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.

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

Repeat this above for the other 2 detectors.

Section 3 - Analysis: 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 $2\sqrt{2ln2}$). 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.

In order to compare the performance of different detector systems in a rigorous manner we must explore how this resolution varies with the energy of the incident radiation.

The measured data-sets and the extracted energy calibration will be used to obtain a relationship between resolution and gamma-ray energy for the sodium iodide and germanium detectors.

The resolution of a photopeak is defined as:

$$R = \frac{\text{FWHM}}{E} = kE^n$$

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

For both sodium iodide and germanium detectors, use the energy calibrated values to:

- extract the values of resolution for each of the six photo peaks in the data-set.
- Plot your extracted values of resolution against known energy
- fit the data with the function $R(E) = kE^n$ in order to obtain k and n.

> - What does the extracted value of n tell you about the operation of this type of scintillation detector?
> - The ideal scintillation detector would follow $R \propto 1/\sqrt{E}$ (i.e. $n = -\frac{1}{2}$). How does your extracted n comapre to this? What does this suggest?
> - A deviation from -0.5 suggests additional contributions to resolution. What could these extra contributions be due to?
> - Other?

- Plot the spectra from the plastic scintillator

> What can you say about the underlying scintillation process for this type of detector from your observations

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

Section 4 -Determination of Isotopic Composition of a Sample

Using the same method as before:

- Plot the energy-calibrated spectra for the Uranium Oxide and Thorium Oxide samples for:
  - the NaI detector
  - germanium detector
- analyse the germanium data and use one of the standard gamma-ray databases (http://nucleardata.nuclear.lu.se/toi/radSearch.asp) to identify the main gamma-ray energies in both the UO2 and ThO2 spectra.
- plot the spectrum for the mineral sample(s) and use the same technique as above for isotope identification.

Questions:

> - In light of what you found in the previous section how do the spectra taken with the different detector types compare?
> - What do these results tell you about the isotopic composition of these samples (at this point, you should be thinking in terms of the relevant decay chains and how they occur naturally).
> - What can you say about the composition of this sample?

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

Section 5 - Simulation: Run a Monte Carlo Simulation

A numerical simulation based upon Monte Carlo techniques is employed in this experiment in order to demonstrate how important and widely-used these types of software tools are in sub-atomic and detector physics.

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, if not done so already, 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 - if not done so already - as detailed in the "Experiment Computing Set Up Document" found on the Lab Moodle page, under the Lab Material→Computing section, execute the command:

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

The simulation we will use 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. We are only simulating the sodium iodide detector here, which is enough for the intended purposes of this exercise.

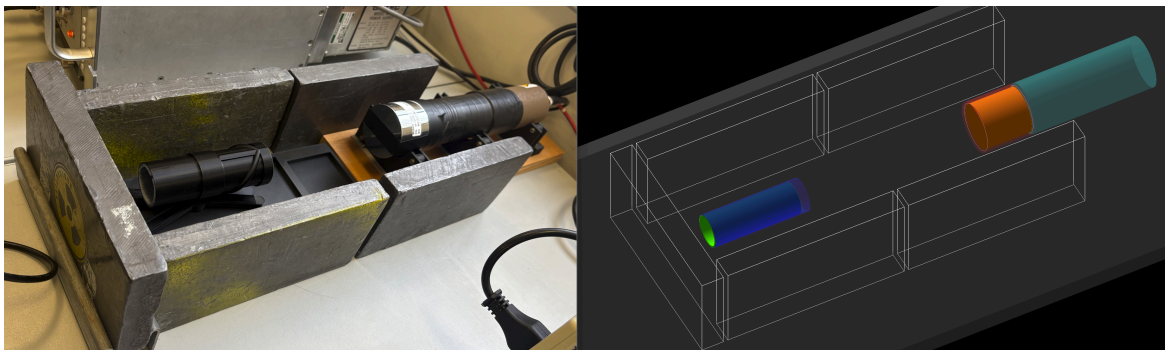

Figure 3: Left shows a sodium iodide scintillation counter in the lab. Right shows a simulation of the detector. detector.

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 several times to simulate measurements taken in the lab during the energy calibration with 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-labs
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-labs
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?