Fortgeschrittenes Physikalisches Praktikum
Sommerferien

Versuch F80 Tutor: Antonios Kontopoulos

Scintillator

This experiment introduces measurement techniques in particle and nuclear physics using both organic and inorganic scintillator detectors. The experiment involves measuring the endpoint energy of a β-decay and performing a coincidence measurement to detect cosmic muons. Through these procedures, we will gain a deeper understanding of the working mechanisms of scintillators and the practical applications of these setups. Additionally, this experiment provides foundational knowledge of detector physics in the context of particle physics.

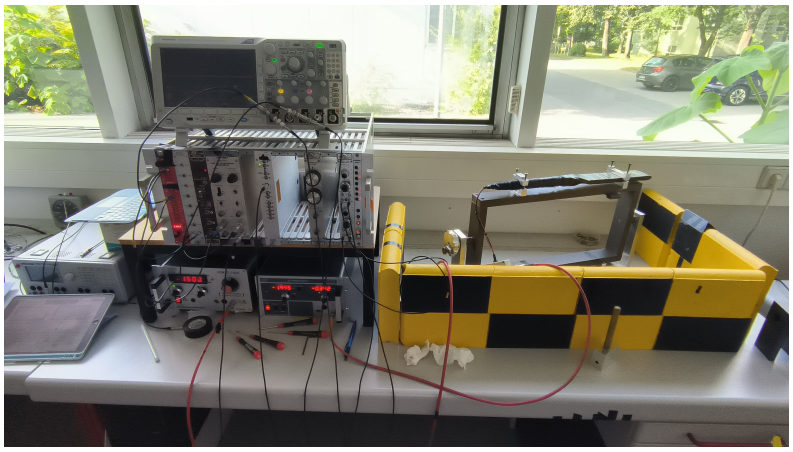


Figure 1: Experimental setup F80/F81 scintillator

1 Physical theories of this experiment

1.1 Radiation

In this experiment, we focus on two decay processes: β-decay and γ-decay. β-decay involves a three-body decay, $n \to p + e^- + \bar{\nu}_e$, and the theoretical spectrum of β-decay is continuous up to an endpoint energy, which can be determined using a Kurie plot.

A Kurie plot is a graph that plots the variable $\sqrt{\frac{\frac{dN}{dE}}{F(Z,E)\cdot(E+mc^2)\cdot\sqrt{E^2+2Emc^2}}}$ against the kinetic energy E. For allowed transitions in β-decay, the plot is linear and intersects the energy axis at $E_{\max}$, representing the maximum energy of the electrons. This method allows for accurate extrapolation of the maximum energy of the β-source.

The emission of γ-radiation occurs as a result of a preceding β- or α-decay, followed by the transition of an excited nuclear state to its ground state. This transition releases energy in the form of photons, which are emitted as the nucleus dissipates excess energy. Unlike β-decay, the spectrum of γ-radiation is discrete.

1.2 Electromagnetic interaction with matter

To detect the signals we need to konw the basic things about the Electromagnetic interaction in detector.
The intensity of electromagnetic radiation, such as photons, decreases exponentially as it passes through a material. The intensity of the outgoing photons after passing through a material with thickness x is given by:

$$I(x) = I_0 e^{-\mu x}$$

where μ is the mass absorption coefficient, determined by the material's cross-section σ, Avogadro's number N_A, and the molar mass A:

$$\mu = \frac{\sigma N_A \rho}{A}$$

Photons interact with matter through three primary processes, and these interactions contribute to the total absorption coefficient, which is the sum of the coefficients from each process:

- **Photoelectric Effect**: Dominant at very low photon energies (100 keV). A photon with energy E_γ greater than the electron's binding energy E_b can be absorbed, releasing an electron with kinetic energy $E_e = E_\gamma - E_b$.
- **Compton Scattering**: Dominant at photon energies between 1 MeV and 2 MeV, where a photon scatters off an electron, transferring part of its energy and momentum.
- **Pair Production**: Becomes significant at photon energies above 5 MeV, where a photon with energy E_γ creates an electron-positron pair. The photon must exceed a threshold energy.

1.3 Interaction of charged particles with matter

When charged particles traverse through matter, they interact with the medium in various ways, including ionization, Bremsstrahlung, Cherenkov radiation, and the emission of transition radiation.

- **Ionization**: As "heavy" charged particles (such as protons and alpha particles) move through a material, they lose energy primarily through ionization. The Bethe-Bloch formula describes the average energy loss per unit distance.
- **Cherenkov Radiation**: This occurs when a charged particle moves through a medium at a speed greater than the speed of light in that medium. The particle emits a characteristic electromagnetic radiation as nearby atoms become temporarily polarized, creating dipoles that radiate as their fields change.

  Cherenkov radiation is particularly useful for detecting light particles.

- **Bremsstrahlung**: Fast charged particles lose energy when they interact with the Coulomb field of nuclei in the medium. As these particles decelerate in the Coulomb field, they emit photons, a process known as Bremsstrahlung.

1.4 Experimental setups

Inorganic Scintillators: These include materials like NaI(Tl) (sodium iodide doped with thallium) and are used for energy measurements. They consist of ionic crystals that emit light when charged particles excite electrons from the valence to the conduction band. The energy resolution is high due to the large amount of scintillation light produced, but time resolution is limited by slow diffusion processes.

Organic Scintillators: These include plastic and liquid types and are ideal for time measurements due to their rapid light decay. They work by using a fluorescent substance that absorbs particle energy and re-emits it as visible light through a wavelength-shifting process.

Photomultipliers: Used to detect the light from scintillators, photomultipliers convert visible light into an electronic signal.

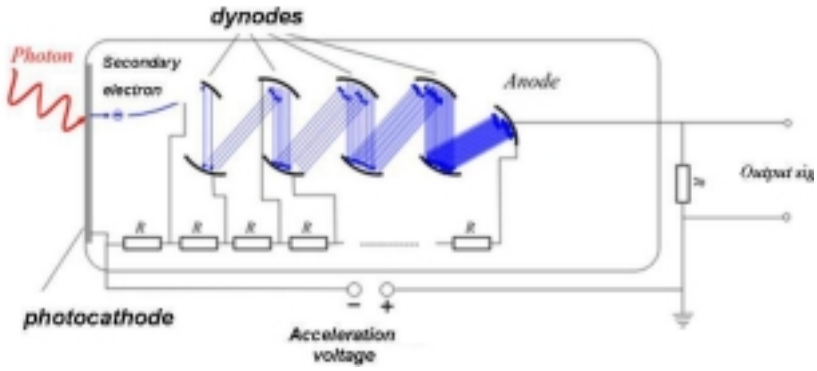


Figure 2: Photomultiplier
Reference: Sebastian Bachmann, script F80/F81 scintillator P.14

Electrons are emitted from a photocathode and amplified through a series of dynodes, with the gain G depending on the number of dynodes N and the potential difference V between them:

$$G = (KV)^N \tag{1}$$

1.5 Coincidence measurement

Coincidence measurements are a technique used in nuclear physics to study reactions or decay processes by detecting the simultaneous presence of two or more particles or signals. This method involves setting a time window, called the coincidence resolution time, to determine if signals from different detectors occur within that window.

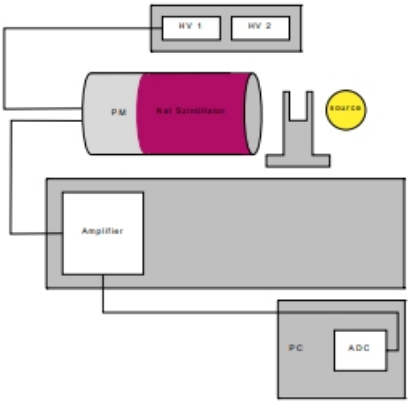
However, accidental coincidences (backgrounds) can occur when unrelated background events in the detectors align by chance within the resolution time. The rate of these accidental coincidences can be calculated using the formula

$$R_{\text{acc}} = \sigma N_1 N_2, \tag{2}$$

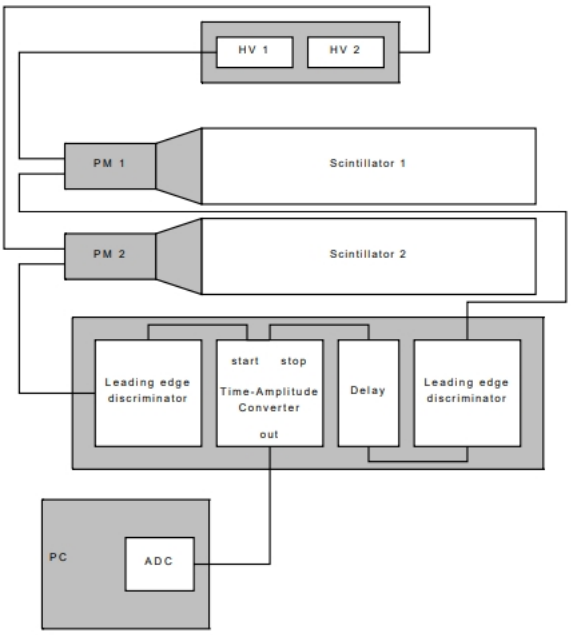
where N_1 and N_2 are the individual signal rates of each scintillator, and σ is the coincidence resolution time.

2 Experimental execution

In this experiment, three radioactive sources—^{137}Cs, ^{60}Co, and ^{90}Sr—are used. The experiment is divided into two parts: first, an energy measurement is performed using the NaI inorganic scintillator to determine the energy of gamma radiation emitted by the sources.



(a) Setup for the energy measurement



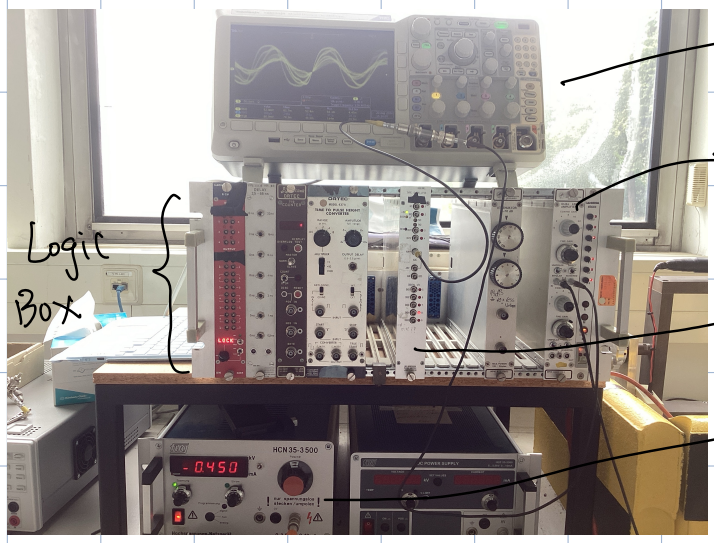
(b) Setup for the coincidecnce measurement

In the second part, a coincidence measurement is conducted with organic scintillators to detect cosmic muons. This involves analyzing the coincidence signals to compare the observed background with theoretical expectations, investigating any discrepancies and validating the experimental results.

The records during the execution can be found in the following measurement protocol.

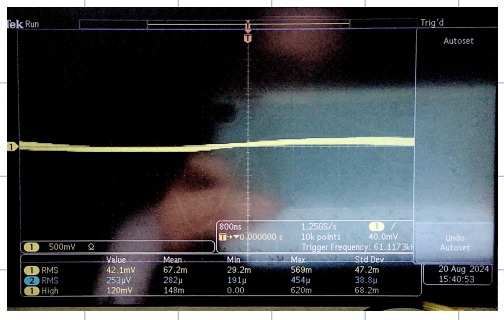
Messprotokoll F80 Yulai Shi
20. 08. 2024 Yuting Shi

Experimental setup:

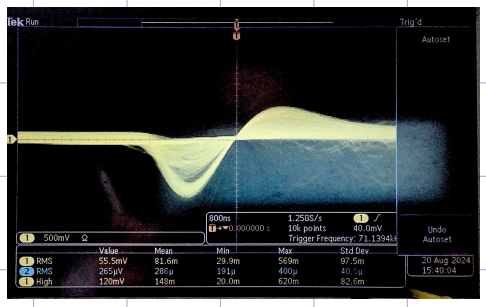


part I: Energy measurements with the NaI scintillator:

1. Signal of the scintillator:



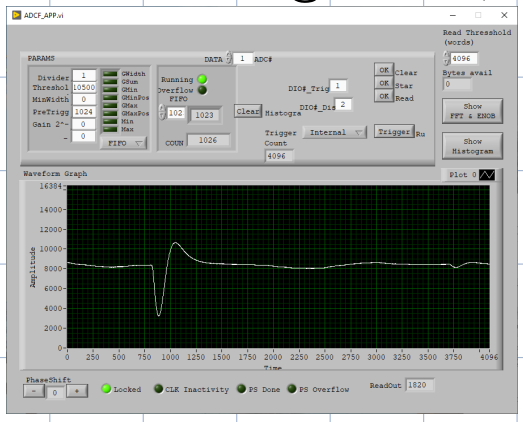
Sigal without Cs



Signal with Cs

Used: ^{137}Cs, LE816, activity: 341.265. (16.11.2022)

P.S.: Without using of amplifier we saw nothing on the oscilloscope.



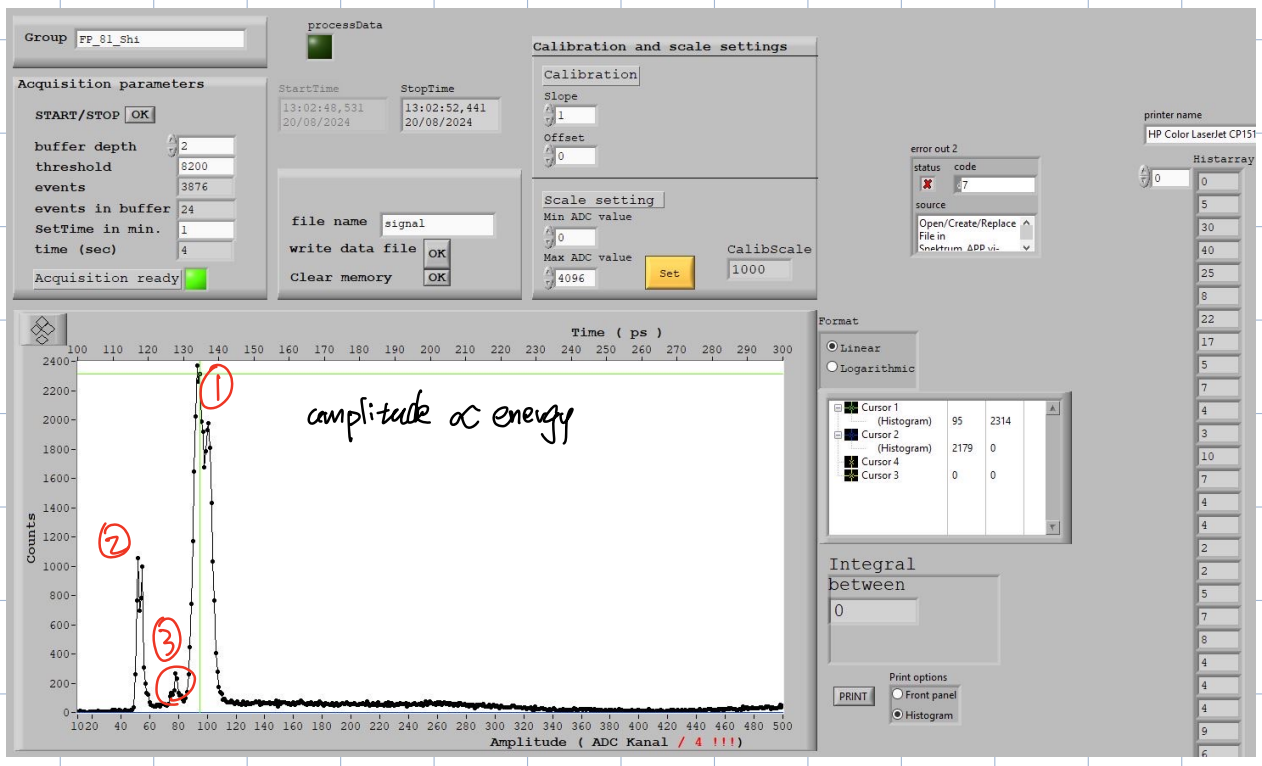
Signal observed in LabView on computer.

2. Measuring and understanding the pulse height spectrum.

Settings amplifier: Coarse gain: 64

Adjust the amplifier gain and got the maximal dynamic range.

Pulse height spectrum:



For the decay of Cesium-137, we typically observe two distinct peaks in the spectrum:

1. Main Peak at the right: This peak corresponds to the gamma-ray emitted when Barium-137 undergoes transition. After Cs decays via beta decay to Ba (in its metastable state), the metastable barium emits a gamma photon with an energy of 661.7 keV. (Peak splits)

2. Backscatter Peak: This is a lower-energy peak that appears due to Compton scattering within the detector or surrounding materials. It's not from a direct decay process but rather a result of scattered photons, typically at an energy around 200 keV. (Peak splits due to the em interactions with material and magnetic field)

Continuum: $\beta-$ decay

3. The third peak: Compton edge

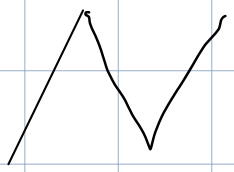
3. Pulse height as function of voltage.

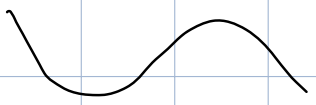
Measurement time: 10s

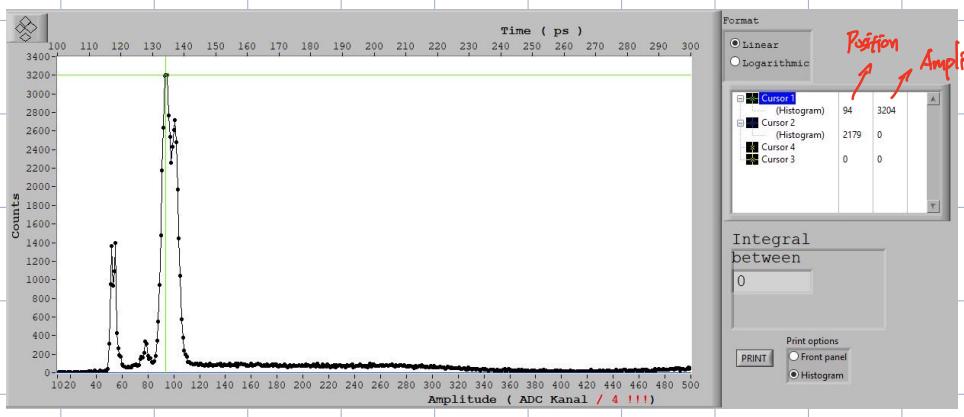
Table 1: HV and photopeak position

HV [V]	Position (channel)	Amplitude x4	
450	94	429	439
440	94		492
435	93	475	514
420	92	459	501
405	92	442	494
390	90		517
380	91	519	548
370	90	620	614
365	90	592	543

Error HV : ±1 V







4. Working voltage

(i) Table 2 : Working voltage

HV [V]	Position x4 (channel)	FWHM x4 [channel]	$\frac{\Delta E}{E}$
450	96	14	0.146
440	98	13	0.133
430	98	13	0.133

420	98	13	0.133
410	97	13	0.134

⇒ choose $\frac{\Delta E}{E}$ minimal, working vltage : 430 V

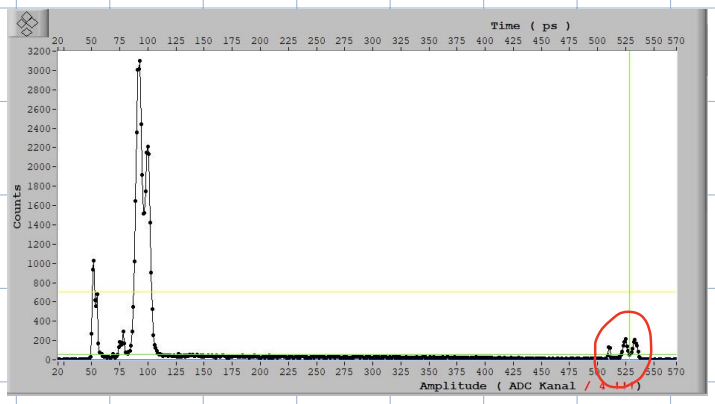
(2) Energy calibration:

^{137}Cs Position photopeak: 98 ± 2 ; E_γ = 0.66166 MeV

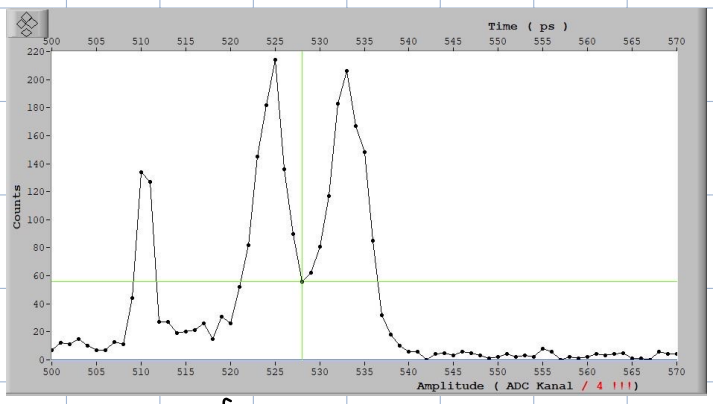
^{60}Co : activity : 189.129 (LZ 817)

γ_1 : 510 ± 2 E_{γ_1} = 1.1733 MeV

γ_2 : 528 ± 2 E_{γ_2} = 1.3325 MeV



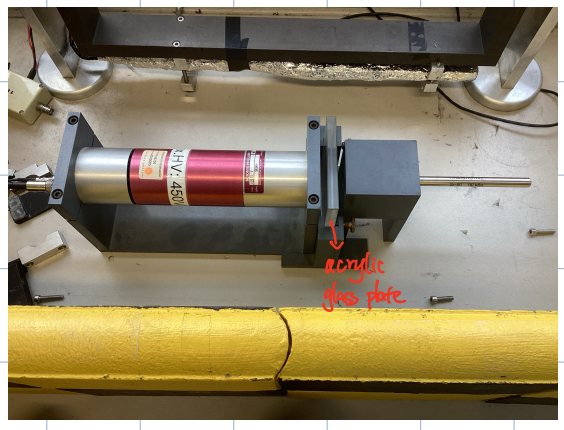
Co signal total

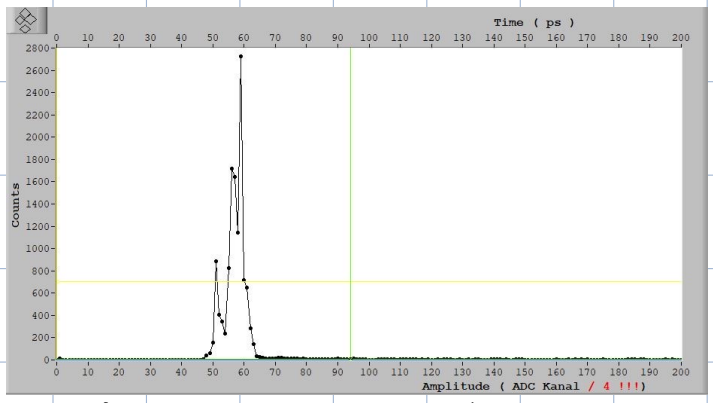


Co signal (γ_1, γ_2)

(3) Determination of the end-point energy.

used : ^{90}Sr activity : 73.601 , 0C309



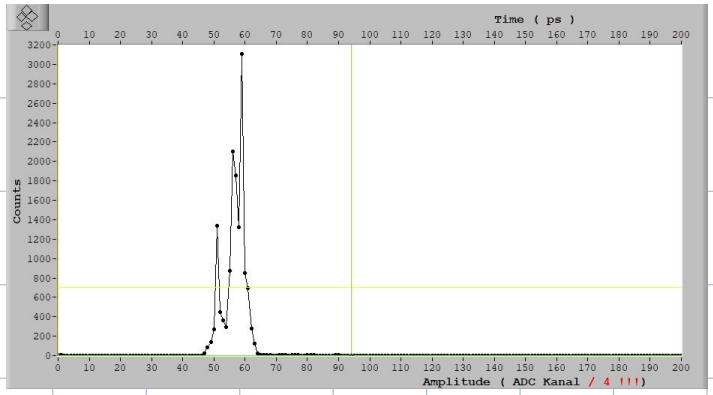


without glass plate (background spectrum)

↙

5 times measured,

to get average values

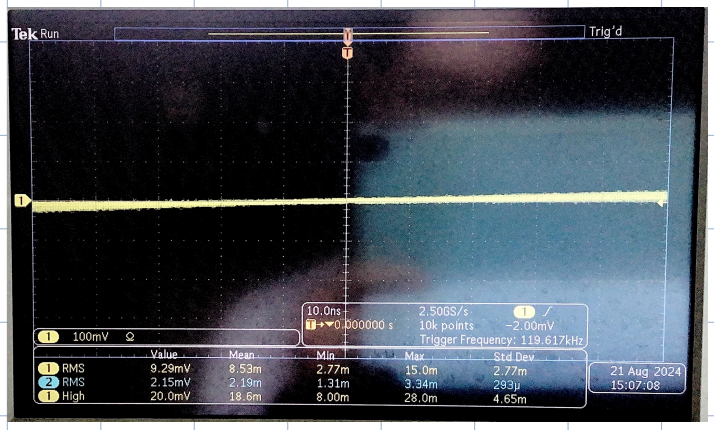


with glass plate

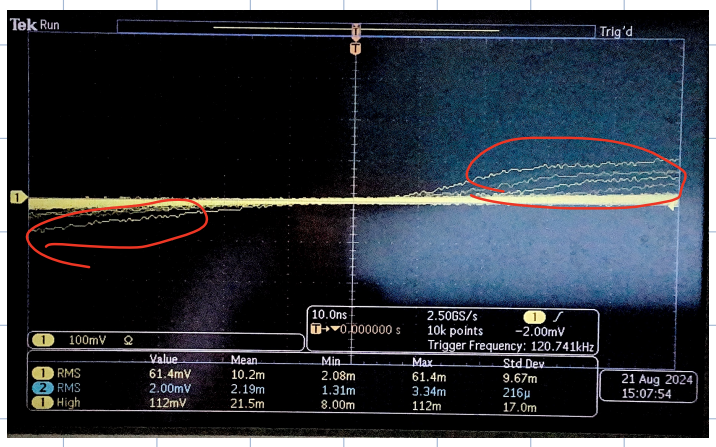
Part I: Coincidence measurements with the organic scintillator

1. Signal of the organic scintillator

Oscilloscope:

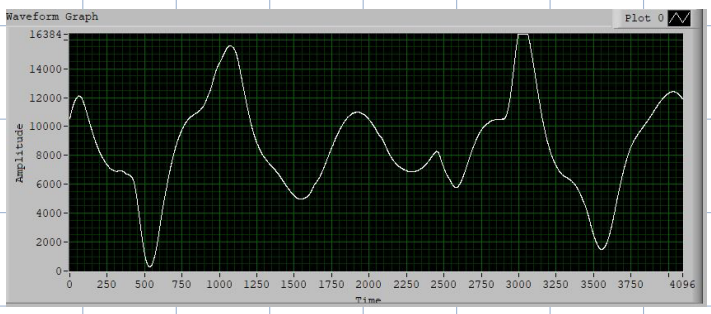


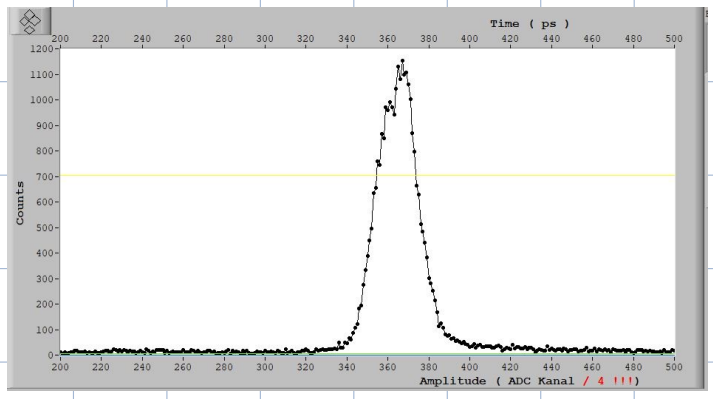
remove source



source near the scintillator

Computer:





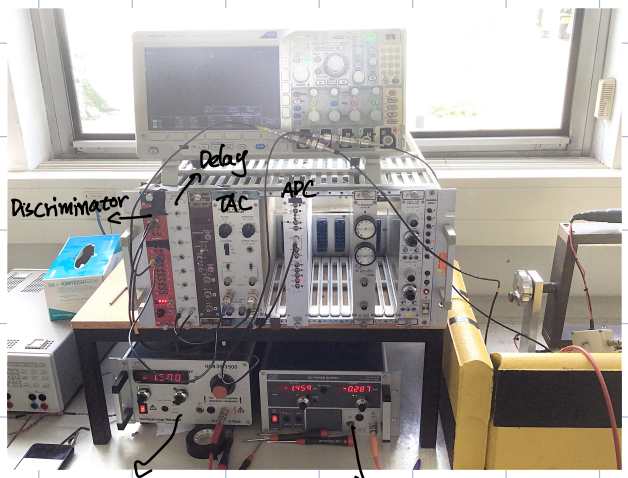
Cs - spectrum computer

peaks in lower energy levels vanished compared to spectrum using inorganic scintillator.

Reason: The organic scintillator has poorer energy resolution, this can cause

the lower- energy features to merge into the Continuum. Also organic scintillator produces less light per unit of deposited energy.

2. Time / Coincidence measurements



Connection setup coincidence measurement

PS: We have used the amplifier since we saw nothing without using of amplifier.

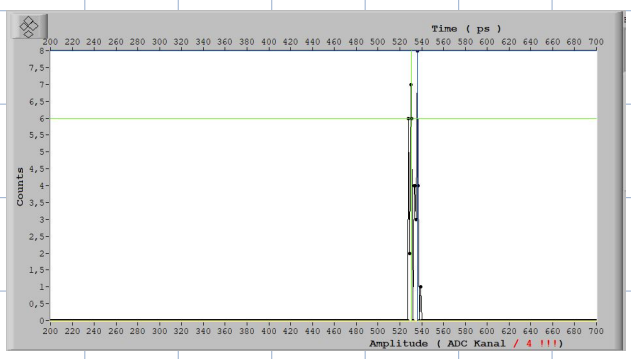
(1) Maximize $\frac{S}{S+B}$, S entries of muons, B entries of background events.

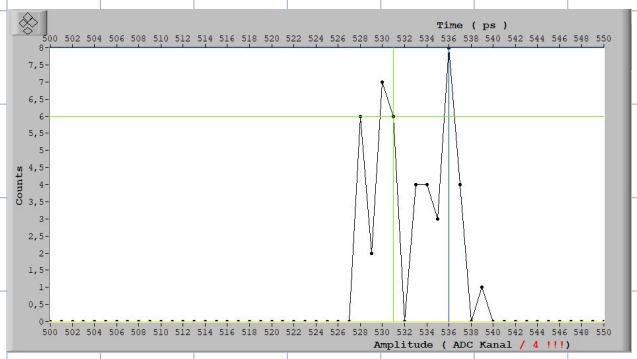
threshdd values: 6000

working voltage: Table 3: Find the working voltage

rate B = 0.06 1/s

	2. HV [V]	S [channel]	B [channel]	$\frac{S}{S+B}$	
5 min	-1500	~~27~~ 21	~~18~~ 15	~~0.58~~ 0.60	0.22
6 min	-1700	~~82~~ 52	~~65~~ 20	~~0.72~~ 0.55	=> 1700 as working voltage.
5 min	-1850	~~400~~ 519	~~464~~ 268	~~0.66~~ 0.46	
3 min	-2004	~~1121~~ ≥900	~~1948~~ 1763	~~0.62~~ 0.37	





determine Working voltage by maximizing the values of $\frac{S}{S+B}$

(2) Accidental coincidences $\sigma = 32$ ns

$R_{acc} = 6 N_1 N_2$, N_1, N_2 counting rates of scintillators 1, 2, 6 coincidence resolution time

N_2: 450 s, # events = 15 $\Rightarrow$ $N_2 = \frac{15}{450} = 0.0333$ 1/s

N_1: 450 s, # events = 13 $\Rightarrow$ $N_1 = \frac{13}{450} = 0.0289$ 1/s

3 Evaluation

3.1 Part I: Energy measurement with the NaI scintillator

A qualitative observation is first made for both the signal on the oscilloscope and via the ADC channel: All spectra with/without Cs137 as well as with/without the amplifier are recorded and presented in the measurement protocol. The primary function of the amplifier is to increase the amplitude of the input signal. By amplifying the signal, the pulse heights become more distinct and easier to resolve. This leads to a clearer separation between different energy levels in the spectrum. We have to choose the amplifier settings according to the available dynamic range as well as on the maximum energy, since the amplifier's gain (how much it amplifies the signal) directly affects the scale of the pulse height spectrum.

To manually produce a pulse height spectrum without a computer or ADC, we use an oscilloscope to observe and measure the amplitude (height) of each pulse generated by the detector. Then we categorize these pulse heights into predefined bins (e.g., by voltage range) and record the number of pulses in each bin, which would be plotted against their corresponding heights to create the pulse height spectrum.

3.1.1 Pulse height as function of voltage

The height of the scintillator pulses corresponds to the amount of energy deposited in the detector. A histogram that records the pulse height data from the scintillator signal (the pulse height spectrum) provides a qualitative estimate of the source's energy spectrum.

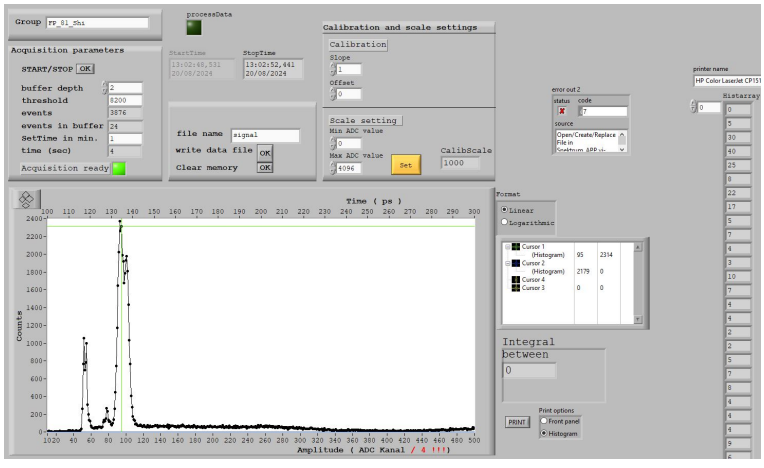
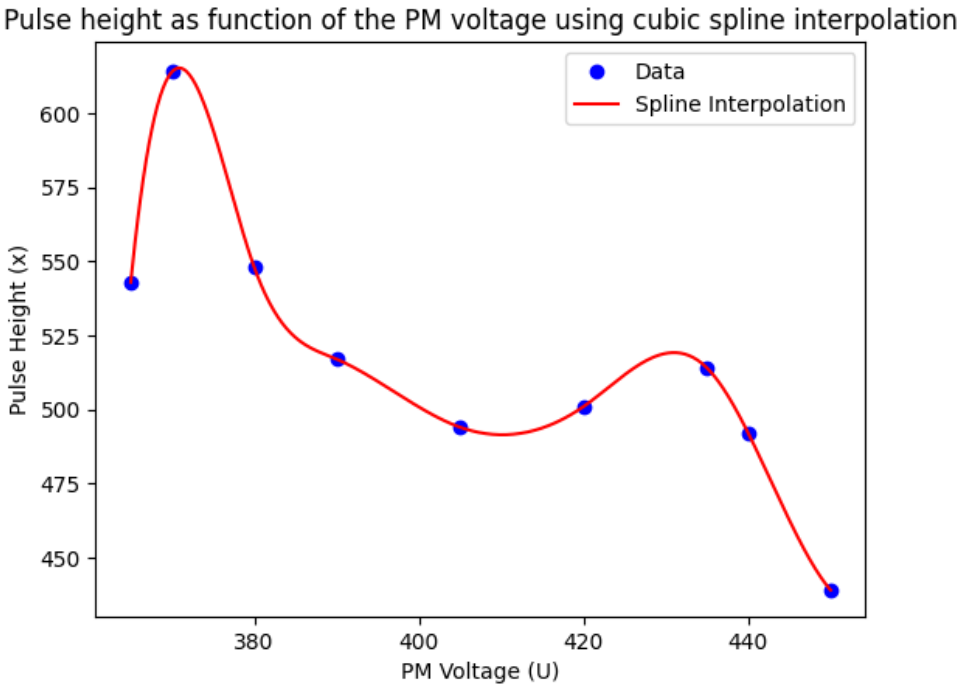


Figure 3: Pulse height spectrum of Cesium-137

In order to find the relationship between Pulse height and voltage we study the pulse height in position of the photopeak and plot the Pulse height as function of the PM voltage, the result ledads:



At fitting the cubic spline interpolation is used because an exact relationship between the two data points is not distinguishable. However, theoretically, it is expected a smooth, generally increasing curve according to eq.(1). At low PM voltages, the pulse height should be relatively small because the gain of the PM tube is low. As we increase the voltage, the amplification of the photoelectrons increases, leading to a corresponding increase in pulse height. In this region, the relationship between the pulse height and the PM voltage is approximately linear or slightly exponential. Later however our PM tube is operating in an unstable or non-ideal regime. At higher voltages, the PM tube might start to saturate, leading to an initial increase in pulse height followed by a decrease as the voltage continues to rise. This could create a peak followed by a decrease, giving the appearance of a damped oscillation.

Another reason could be manual manipulation. Since the program does not allow for an automatic setting of the measurement time to 10 seconds, one must manually monitor when the measurement ends, which can introduce an error of up to 1 second. Additionally, a significant cause is related to the scintillator and the probe itself, since these devices have been in use for over 10 years, thus the signals they generate are not necessarily uniform and stable.

3.1.2 Energy calibration

In order to choose an optimal working voltage, we determine the energy resolution $\Delta E/E$ of the detector as a function of the high voltage and for fixed E. As an estimator for ΔE, we use FWHM of the photopeak (in units of channels) and use the channel of the peak position for E. Minimize the resolution a working voltage of 430V are choosen and used for the following energy calibration.

Using the energy of the photons of the Cs137 decay and Co60 decay a energy calibration is made, which we do a linear fit to find the relation between channel number and energy.

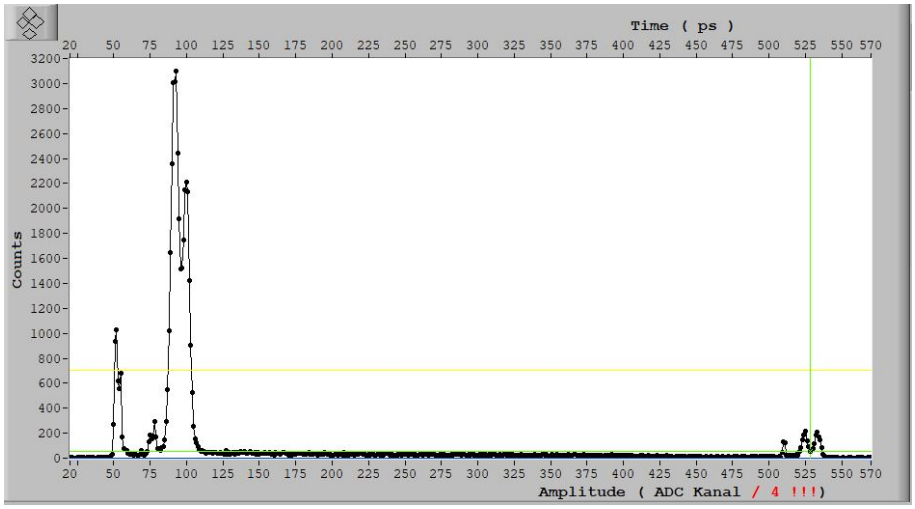


Figure 4: Pulse height spectrum of Co-137

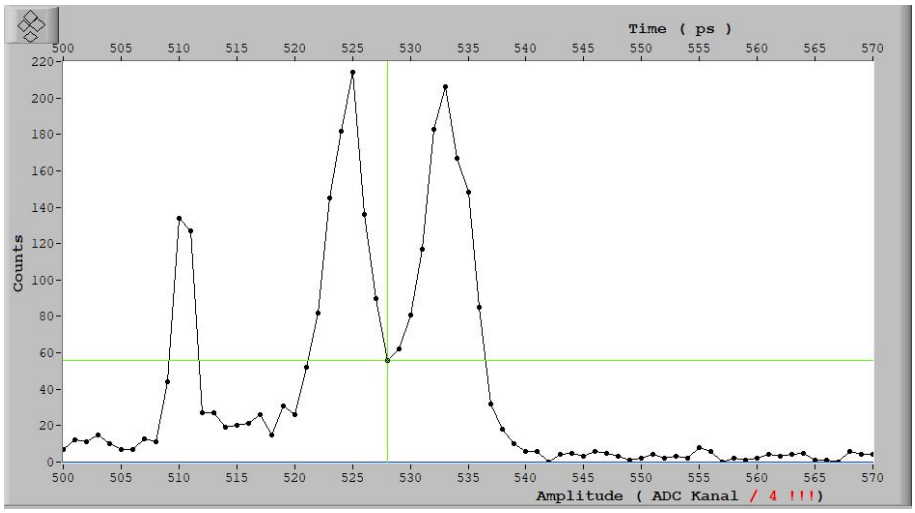
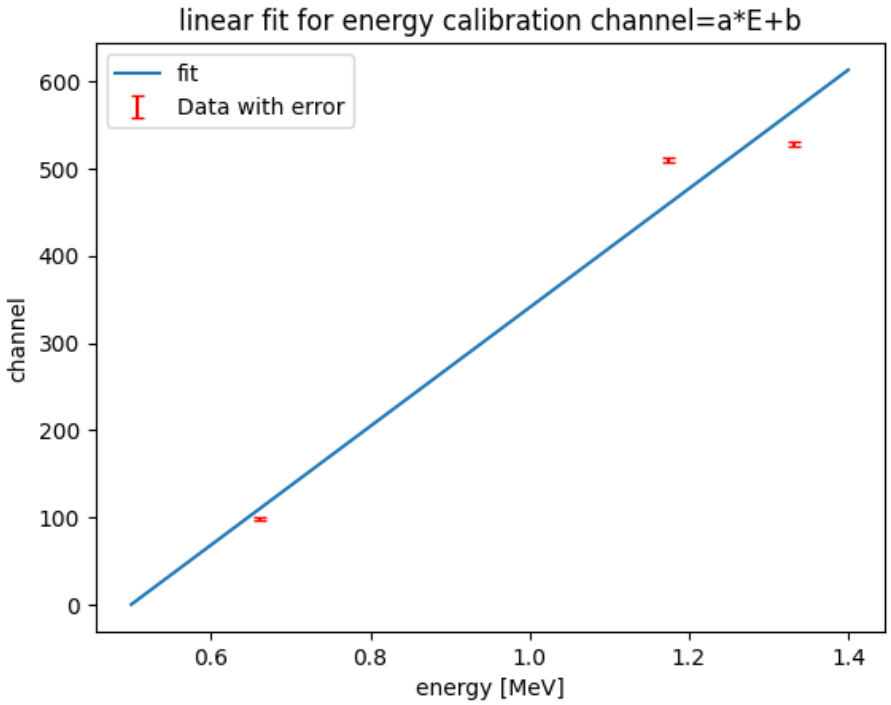


Figure 5: Zoom of Pulse height spectrum of Co-137 in second light position



As result we find:

$$\text{channel} = aE + b \tag{3}$$

with $a = (681 \pm 16)$ MeV^{-1}, $b = (-341 \pm 20)$ MeV^{-1}.

3.1.3 Determination of the end-point energy via Kurie-plot

The energy spectrum of the Sr90 source is recorded:

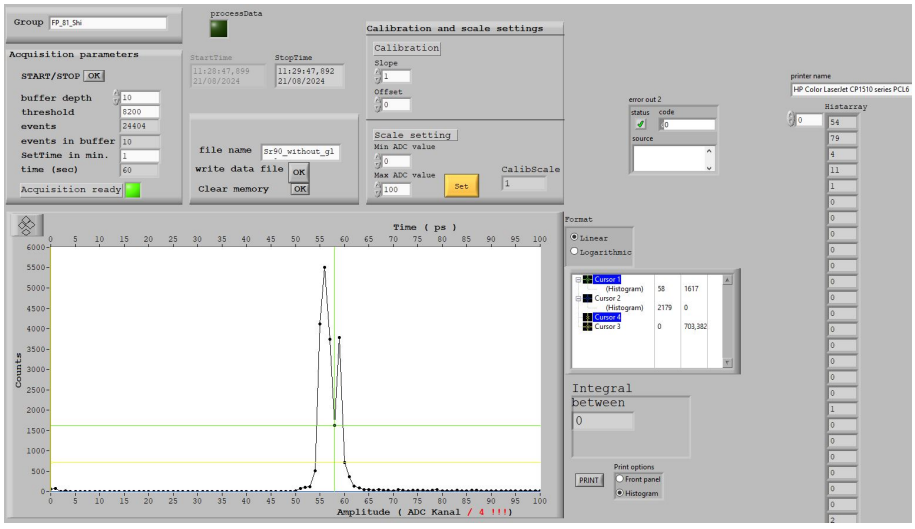


Figure 6: Pulse height spectrum of Sr-90

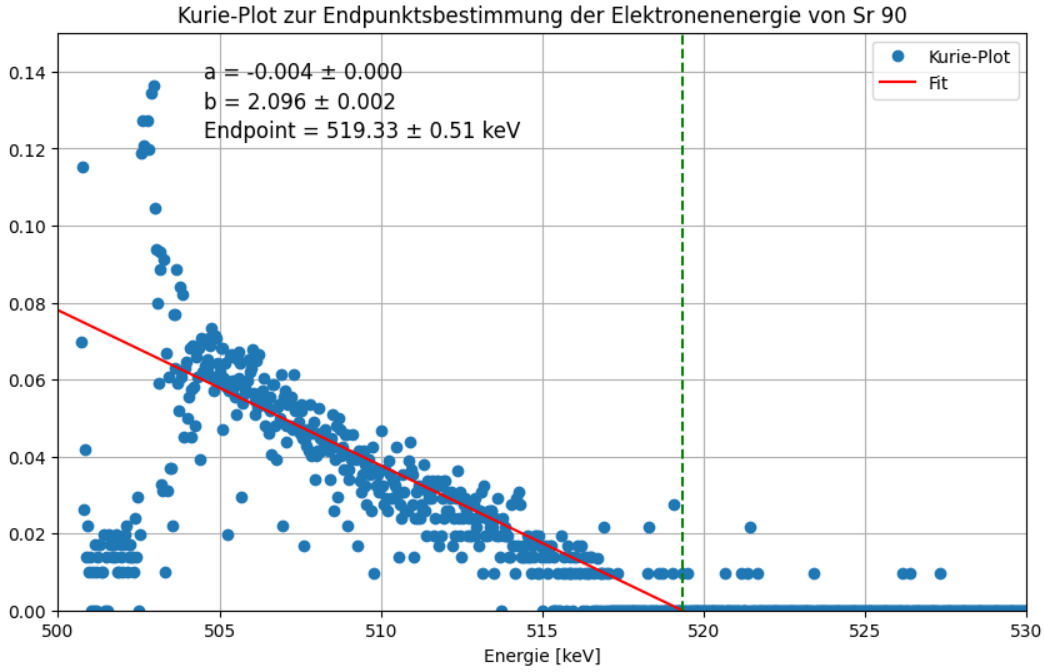
Compared to the β-decay spectrum, only one peak in the low energy levels can be observed in both spectra approximately. Sr90 decays via beta emission to Y90 (Yttrium-90), whose energy spectrum is continuous. This spectrum has a maximum endpoint energy that is characteristic of the beta decay process. We will determine this critical energy using a Kurie plot. However, for the Sr90 pulse height spectrum, an additional peak is observed due to fine structure splitting.

In addition, we have other effects that contribute to this pulse height spectrum: Beta particles lose energy as they travel through the detector material due to ionization, Bremsstrahlung, Cherenkov Radiation or other possible interaction processes for charged particles and give this part energy as photon, which can be absorbed by the photomultiplier and recorded in the pulse height spectrum.

Background Correction

Typical beta radiation can be effectively shielded with just a few centimeters of material, especially if the material has a low atomic number to reduce the production of hard bremsstrahlung radiation. Therefore, we can accurately measure the background spectrum by shielding the primary electrons. For this purpose, an acrylic glass plate is used, as acrylic glass is composed of carbon, oxygen, and hydrogen—all elements with low atomic numbers.

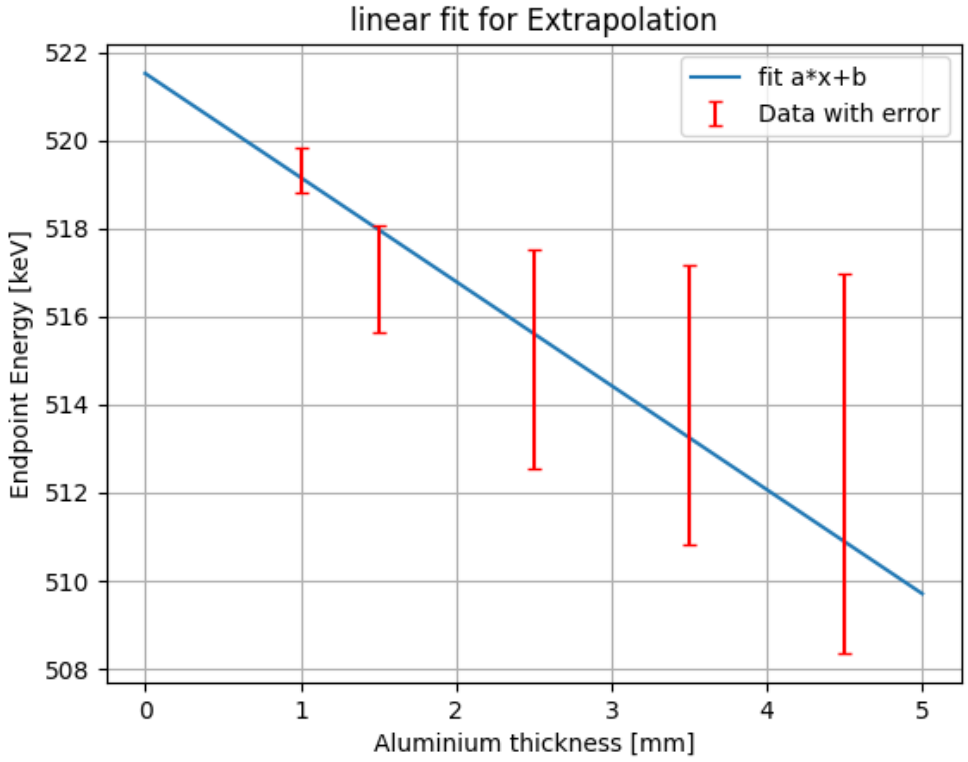
For this purpose first the ^{90}Sr source and acrylic glass with the detector can be setted with the puls height spectrum measured. Then we Remove the acrylic glass and record the ^{90}Sr spectrum for 5 times. Subtract the background spectrum from the recorded Sr90 spectrum, we can use the corrected spectrum to produce a Kurie-Plot and find out the endpoint energy of the electron.



Due to the significant instability of the signal previously mentioned, the data are not strongly linearly correlated with each other, which is why a linear fit is only possible in the middle range. Compared to the literature value $E_{\mathrm{Lit}} = 546$ keV, this results in an error deviation:

$$\frac{|E_0 - E_{Lit}|}{\Delta E_{Lit}} \approx 52.3\sigma \tag{4}$$

which is already significantly larger than 3σ and therefore significant. Due to the 1mm aluminum plate in front of the scintillator, a portion of the particles is already shielded, which is why the measurement is not exact.So we should determine the corrected endpoint energy by repeated measurements of the endpoint energy for which additional aluminium material between source and detector is introduced. In the plot of endpoint energy as function of Aluminium thickness, using extrapolation towards zero total aluminium we can get the value that may be better.



The fit parameters yield:

$$a = (-2.36 \pm 0.63)\ \mathrm{keV/mm} \qquad b = (521.52 \pm 1.13)\ \mathrm{keV} \tag{5}$$

After extrapolation, we obtained an endpoint energy for an aluminum thickness of 0 mm, $E_0 = (521.52 \pm 1.13)$ keV, which deviates from the literature value by 21.67σ. Although this deviation is significantly smaller than the deviation before extrapolation, it is still considerable.

3.2 Part II: Coincidence measurements with the organic scintillator - Measurement of cosmic muons

3.2.1 Signal of the organic scintillator

We want to compare the differences between the spectrum using organic scintillator and the inorganic scintillator.

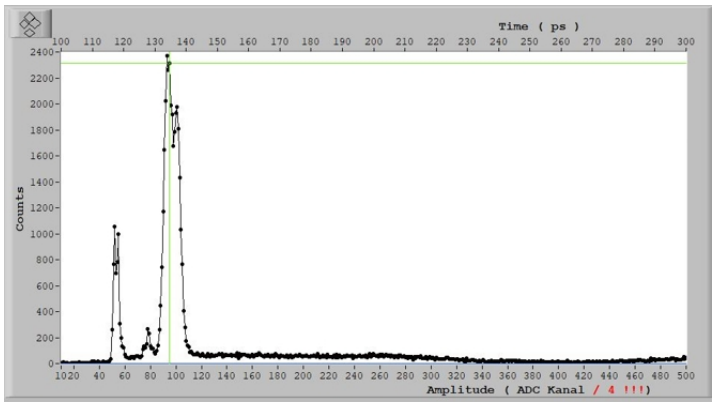


Figure 7: Cs-spectrum of the inorganic scintillator

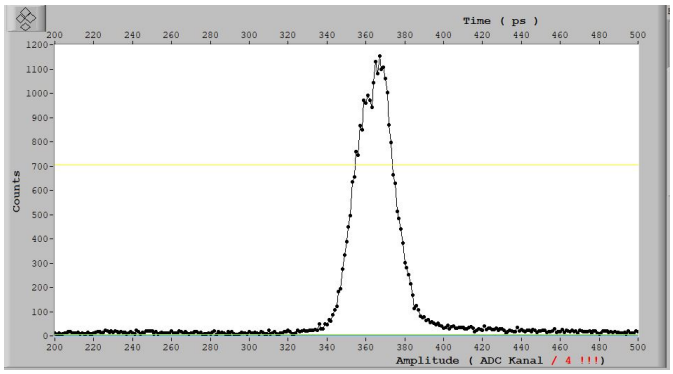


Figure 8: Cs-spectrum of the organic scintillator

We observe that the peaks in the low-energy levels of organic scintillators are less pronounced compared to those in inorganic scintillators, and the main peak appears smoother. Several factors may contribute to this observation:

1. **Energy Resolution**: Organic scintillators have poorer energy resolution compared to inorganic ones. This results in broader peaks and less distinct low-energy features, causing lower-energy peaks to merge into the continuum and become less detectable.

2. **Light Yield**: Organic scintillators produce less light per unit of deposited energy, leading to less efficient detection of low-energy events. This reduces the prominence

of low-energy peaks in the spectrum and noises of the main peak.

3. **Gamma Interaction Mechanisms**: Organic scintillators are more sensitive to Compton scattering, which contributes to a broad continuum rather than distinct peaks. In contrast, inorganic scintillators, with their higher atomic number, are more effective at detecting gamma rays via the photoelectric effect (dominant in low energies).

3.2.2 Determination of the parameters for the coincidecnce measurement

Basic keypoints of the coincidecnce measurement

- **Distinguish signals**: Firstly we introduce the method to distinguish the random (background) from true (the muon signal) events. We use a Time to Amplitude Converter (TAC) to measure the time difference between events detected by two scintillators. The TAC provides a pulse height proportional to the time difference between the start and stop signals, creating a time spectrum. True coincidences, where muons pass through both scintillators, will appear as peaks in this spectrum at **shorter time intervals**, while random coincidences will be spread over a **broader range**.

- **Delay time**: The delay time, set using the TAC, impacts the measurement of coincidences, it ensures that only signals within this time window are considered coincident. It is necessary to set up a delay because there is a distance between the two scintillators. We must ensure that the signals from both scintillators arrive simultaneously. To achieve this, we should introduce a delay to one of the signals.

- **Discriminator**: A discriminator is an electronic device that converts an analog input signal into a digital output based on a set threshold. It produces a fixed-height pulse when the input signal's amplitude exceeds this threshold, effectively filtering out low-amplitude noise. A leading edge discriminator specifically responds to the rising edge of the signal as it crosses the threshold.

Threshold adjusting

To adjust the discriminator threshold for optimal performance, set the threshold so that it balances between maximizing signal detection and minimizing background noise. Aim to maximize the signal-to-background ratio ($\frac{S}{S+B}$) by counting valid signals (S) and background events (B) and adjusting the threshold accordingly. Our threshold value is 6000.

Working voltage

Since the rate of detected signals can be adjusted by varying the high voltage, as described in Equation (1), we determine the appropriate value by maximizing the signal-to-background ratio ($\frac{S}{S+B}$). The table of all data is provided below.

Measurement time [min]	2. high voltage [V]	Ratio $\frac{S}{S+B}$	Rate B [1/s]
5	-1500	0.60	0.06
5	-1700	0.55	0.22
5	-1850	0.46	1.55
3	-2000	0.37	10.82

Table 1: Data working voltage

However, we selected a working voltage of -1700 V, even though the signal-to-background ratio at -1700 V is lower than at -1500 V. The lower voltage would not allow us to observe the discriminator signals in the next part of the experiment. Therefore, we opted for the second highest value.

It is important to emphasize that the selected values for the threshold and working voltage are not very precise or reliable due to the limited number of measurements and the inadequate resolution of the test value ranges.

3.2.3 Accidental coincidences

In this part of the experiment, we compare our theoretical background rate with the experimentally determined background rate. In the previous section, the rate of background signals was measured as 0.22 1/s. Using the formula (2)

$$R_{\text{acc}} = \sigma N_1 N_2, \tag{6}$$

where σ=32ns is the delay time, and N_1 and N_2 are determined from direct measurements of the discriminator signal rates, we find

$$N_1 = 0,0289\,1/s$$
$$N_2 = 0,0333\,1/s.$$

The theoretical value of the accidental background signal rate is

$$B_{theo} = 3,08 \cdot 10^{-11}\,1/s$$

This is an extremely small value compared with the value 0,22 1/s and suggests that, theoretically, there should be no significant background signal in our spectra. Consequently, we would expect to observe very high peaks, with the remainder of the spectrum being close to zero. Although some similar trends were observed in our diagrams, the discrepancy implies that the background signals may have been incorrectly counted, and the working voltage and threshold values may no longer be accurate.

3.2.4 Scintillator orientation

Now we will explore the effect of scintillator orientation on coincidence measurements of cosmic muons. We placed the two scintillators either parallel or anti-parallel and

recorded the pulse height spectra for both orientations. We have setted the high voltage to 2000 V to obtain more signals.

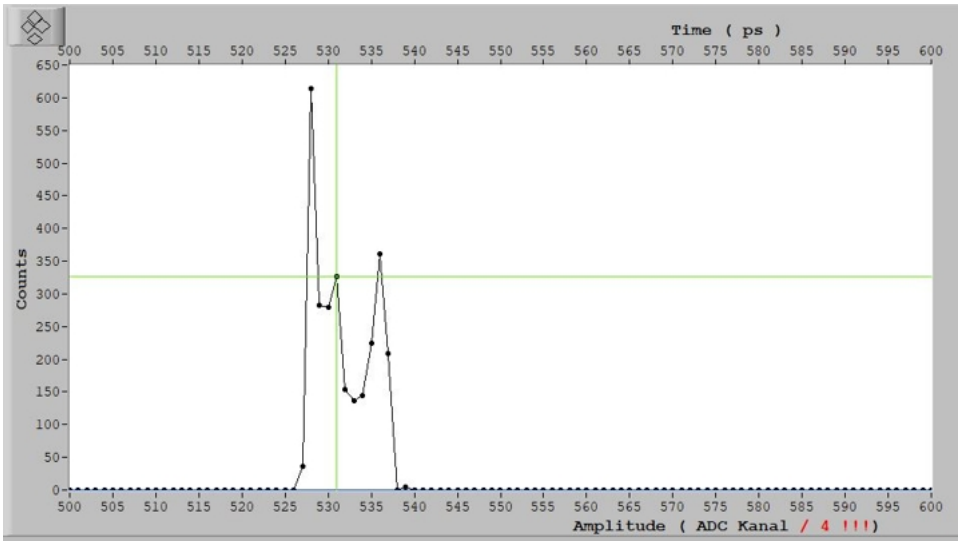


Figure 9: Anti-parallel settings

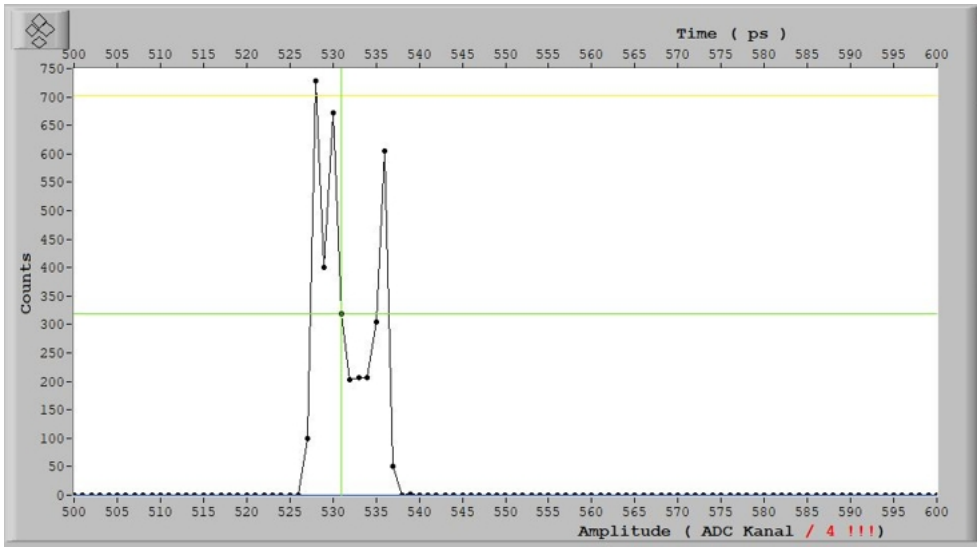


Figure 10: Parallel settings

As we saw from the two spectra, the coincidence rate of parallel orientation is obviously to be higher because cosmic muons, which travel through the scintillators in a straight line, are more likely to trigger both scintillators. The pulse height spectrum showed us also clearer peaks.

Parallel Orientation

In this configuration, the two scintillators detect cosmic muons that pass through both detectors along the same path. The muons have to travel through both scintillators, which maximizes the probability of detecting a signal in both detectors simultaneously.

Anti-Parallel Orientation

Here, the scintillators detect cosmic muons traveling in directions that may not align perfectly with the detectors' orientations. Therefore, a cosmic muon passing through one scintillator might not necessarily pass through the other, so that only a subset of muons will pass through both detectors. The pulse height spectrum might show fewer or less distinct peaks due to the reduced number of coincident events.

3.2.5 Calibration of the time

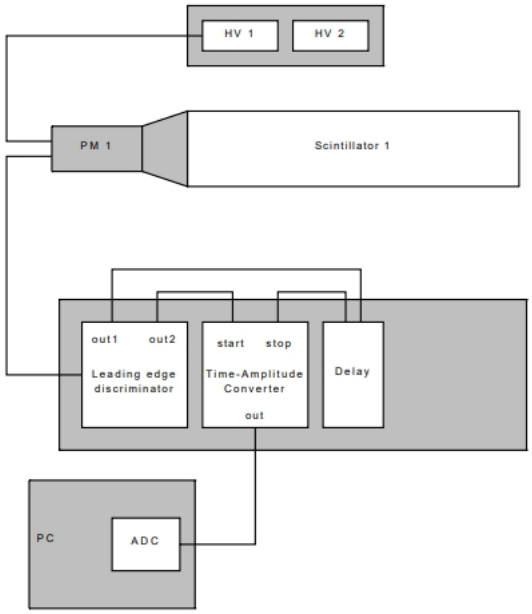


Figure 11: Setup for the time calibration

To calibrate the time resolution, we split the start signal and apply known delays to the stop signal with a Y-adapter. Record the TAC pulse height spectrum for each delay. Then we will plot the mean pulse height against delay times and use linear regression (y = ax +b) to determine the correlation between channel numbers and time differences, which will quantify the coincidence resolution time.

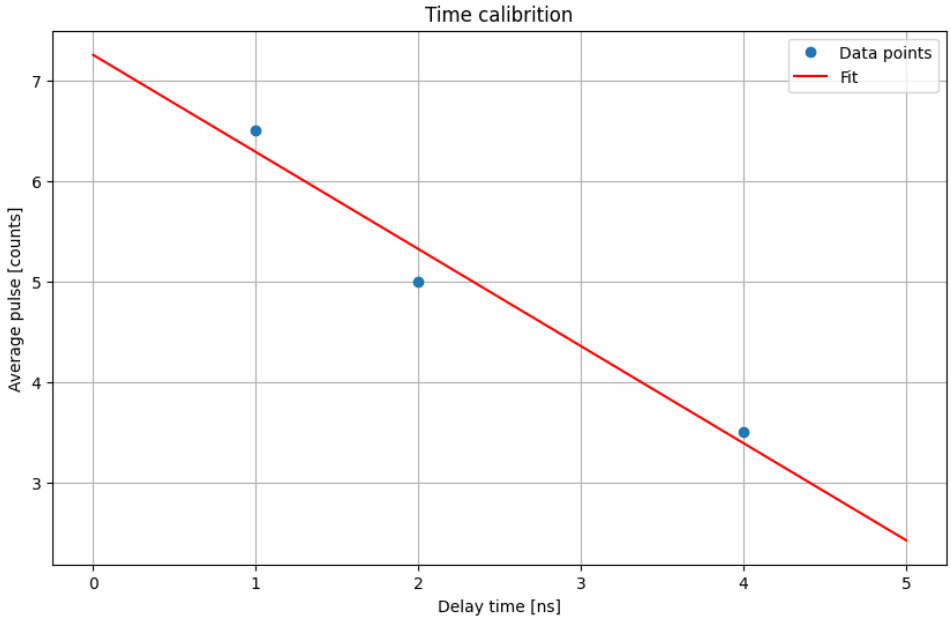


Figure 12: Linear regression for the time calibration

With Python we got the results as

$$\underline{\underline{a = (-0,964 \pm 0,186)\, 1/ns}}$$

$$\underline{\underline{b = (7,250 \pm 0,491)\, 1/ns}}$$

In this analysis, we only considered three data points to establish our fit curve because the fourth data point exhibited a significant deviation from the fitting line. Therefore, it was excluded from the analysis.

The errors in our measurements might come from both reading inaccuracies and detector limitations, such as resolution and sensitivity. Additionally, the "Walk" effect of our discriminator may contribute to these errors. This effect refers to the variation in the timing of the output pulse based on the amplitude of the input signal. As the input pulse amplitude changes, the point at which the pulse exceeds the threshold and triggers an output can shift. Consequently, higher amplitude signals might trigger slightly earlier or later than lower amplitude ones. Although this effect is usually minimized within a controlled range of input signals, significant amplitude variations can introduce timing inaccuracies in our measurements.

Reference

Sebastian Bachmann, script F80/F81 scintillator, April 10, 2018, Ruprecht-Karls-Universität Heidelberg.
`https://www.physi.uni-heidelberg.de/Einrichtungen/FP/anleitungen/F80.pdf`

FP_81_python

August 23, 2024

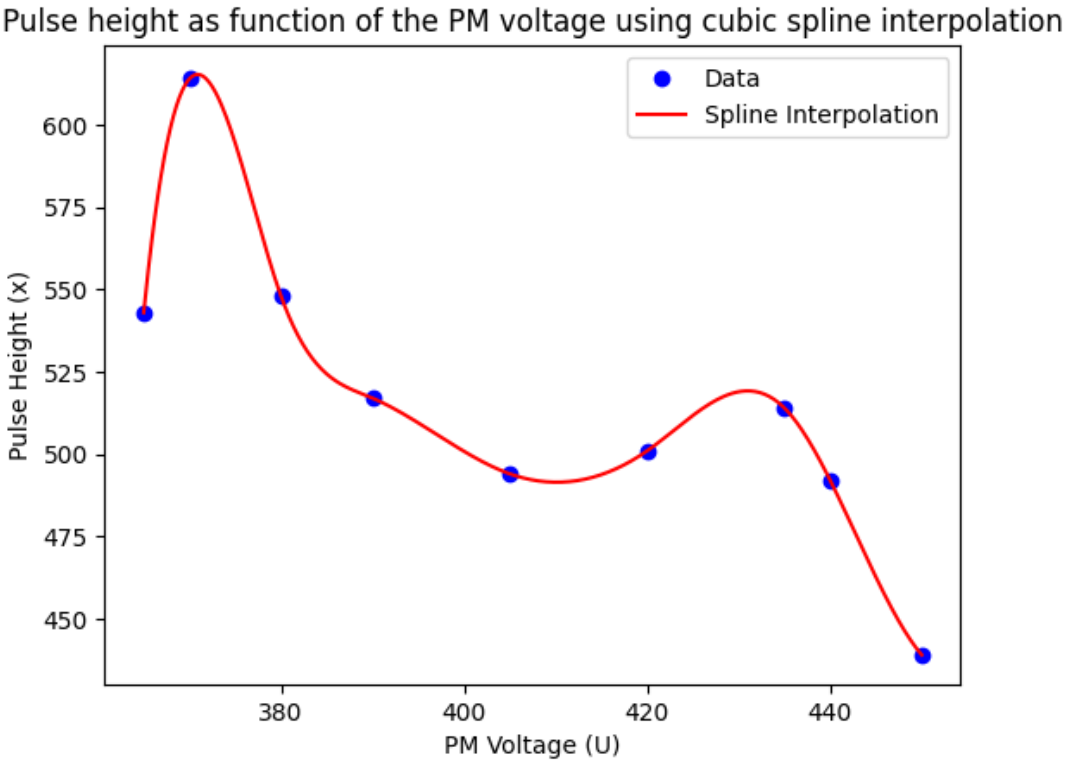
0.0.1 Part I

[73]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

pulse height as function of PM voltage

[74]:
```
from scipy.interpolate import interp1d

U = np.array([450, 440, 435, 420, 405, 390, 380, 370, 365])
x = np.array([439, 492, 514, 501, 494, 517, 548, 614, 543])

# spline interpolation
spline_interp = interp1d(U, x, kind='cubic')
U_fine = np.linspace(min(U), max(U), 500)
x_spline = spline_interp(U_fine)

# Plot the original data and the spline interpolation
plt.plot(U, x, 'o', label='Data', color='blue')
plt.plot(U_fine, x_spline, label='Spline Interpolation', color='red')  # Spline␣
 ↪fit
plt.title('Pulse height as function of the PM voltage using cubic spline␣
 ↪interpolation')
plt.xlabel('PM Voltage (U)')
plt.ylabel('Pulse Height (x)')
plt.legend()
```

[74]:
```
<matplotlib.legend.Legend at 0x1a6639662d0>
```



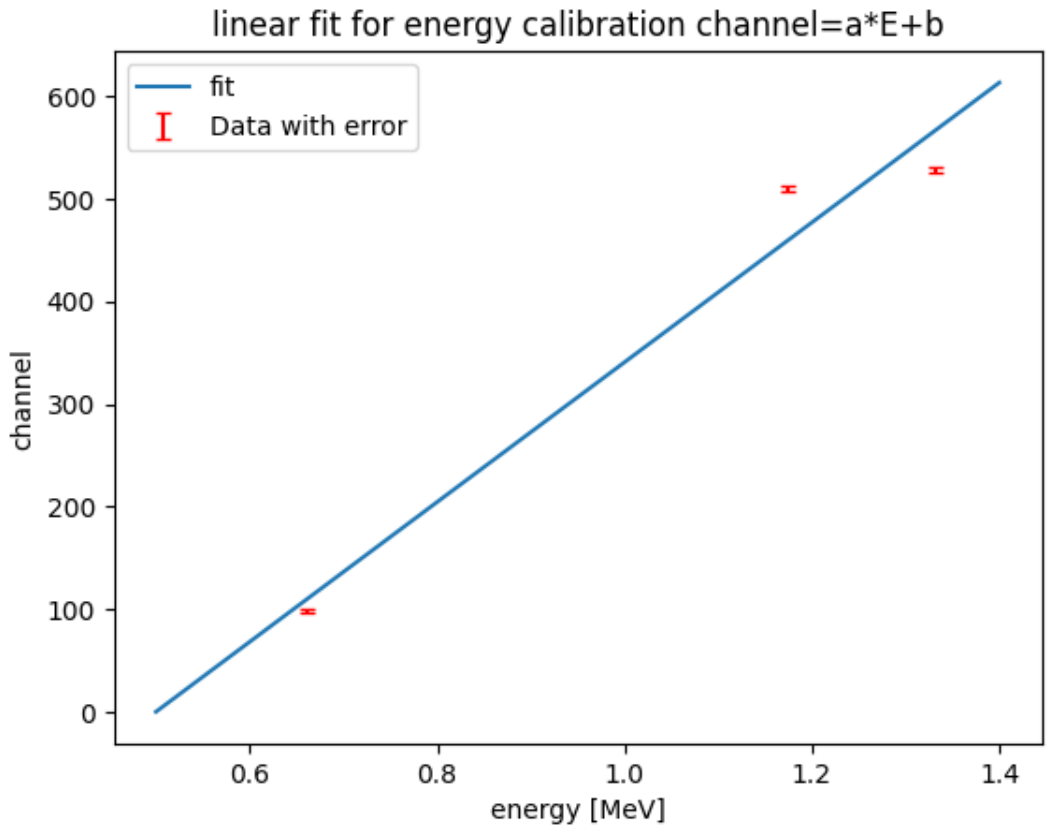
energy calibration

[75]:
```
channel=np.array([98, 510, 528])
energy=np.array([0.66166, 1.1733, 1.3325])  # unit MeV
channel_error=np.array([2,2,2])

def linear(x, a, b):
    return a*x+b

popt, pcov=curve_fit(linear, energy, channel, sigma=channel_error,␣
 ↪absolute_sigma=True)
print('a=', popt[0], '+-', pcov[0][0])
print('b=', popt[1], '+-', pcov[1][1])

range=np.linspace(0.5, 1.4, 100)
plt.plot(range, linear(range, *popt), label='fit')
plt.errorbar(energy,channel, yerr=channel_error, fmt='None', ecolor='r',␣
 ↪capsize=3, marker='s', mec='black', mfc='black', ms='2', label='Data with␣
 ↪error')
plt.xlabel('energy [MeV]')
plt.ylabel('channel')
plt.title('linear fit for energy calibration channel=a*E+b')
```

```
plt.legend()
```

```
a= 681.170598453397 +- 16.278990593212285
b= -340.526874592399 +- 19.480433624699174
```

[75]: <matplotlib.legend.Legend at 0x1a666803e60>



kurie-plot

[76]:
```
with open('C:\\Users\\shiy0\\OneDrive\\Dokumente\\FP\\F 80␣
 ↪Scintillators\\FP_81_Shi\\Sr90_with_glas.dat', 'r') as file:
    content = file.read()

content = content.replace(',', '')
background_str = content.split()
background = [int(x) for x in background_str]
print(background)
```

```
[47000, 85000, 3000, 9000, 0, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1000, 3000, 2000, 10000, 90000, 115000, 136000, 448000, 4105000, 5652000,
```

4180000, 1952000, 4404000, 808000, 511000, 180000, 95000, 23000, 17000, 19000,
20000, 12000, 20000, 16000, 14000, 22000, 10000, 16000, 20000, 18000, 13000,
16000, 12000, 9000, 6000, 2000, 17000, 9000, 2000, 8000, 8000, 8000, 5000, 5000,
11000, 7000, 5000, 5000, 8000, 4000, 6000, 5000, 6000, 4000, 1000, 7000, 5000,
3000, 4000, 4000, 3000, 3000, 2000, 1000, 2000, 3000, 3000, 1000, 3000, 3000,
3000, 3000, 1000, 2000, 1000, 2000, 4000, 2000, 3000, 6000, 2000, 1000, 2000,
3000, 2000, 1000, 2000, 0, 1000, 0, 0, 3000, 2000, 1000, 1000, 0, 1000, 0, 0, 0,
0, 1000, 2000, 1000, 0, 0, 0, 0, 1000, 0, 1000, 2000, 0, 0, 1000, 0, 1000, 1000,
0, 1000, 1000, 3000, 0, 0, 1000, 0, 3000, 0, 0, 1000, 1000, 0, 0, 0, 0, 0, 1000,
0, 1000, 0, 0, 0, 0, 0, 1000, 3000, 0, 0, 0, 1000, 0, 0, 1000, 0, 0, 1000, 0, 0,
0, 0, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 1000, 1000, 0, 0, 0, 1000, 0, 0, 0, 0, 1000,
0, 0, 1000, 0, 1000, 0, 0, 0, 0, 0, 0, 2000, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1000, 0, 1000, 0, 0, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1000, 1000, 0, 0, 0, 0, 0, 0, 0, 0,
1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1000, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1000,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1000, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]
```

[77]:

```python
# without glas 1
with open('C:\\Users\\shiy0\\OneDrive\\Dokumente\\FP\\F 80␣
↪Scintillators\\FP_81_Shi\\Sr90_without_glas1.dat', 'r') as file:
    content = file.read()
content = content.replace(',', '')
with_background1_str = content.split()
with_background1 = [int(x) for x in with_background1_str]

# without glas 2
with open('C:\\Users\\shiy0\\OneDrive\\Dokumente\\FP\\F 80␣
↪Scintillators\\FP_81_Shi\\Sr90_without_glas2.dat', 'r') as file:
    content = file.read()
content = content.replace(',', '')
with_background2_str = content.split()
with_background2 = [int(x) for x in with_background2_str]

# without glas 3
with open('C:\\Users\\shiy0\\OneDrive\\Dokumente\\FP\\F 80␣
↪Scintillators\\FP_81_Shi\\Sr90_without_glas3.dat', 'r') as file:
    content = file.read()
content = content.replace(',', '')
with_background3_str = content.split()
with_background3 = [int(x) for x in with_background3_str]

# without glas 4
with open('C:\\Users\\shiy0\\OneDrive\\Dokumente\\FP\\F 80␣
↪Scintillators\\FP_81_Shi\\Sr90_without_glas4.dat', 'r') as file:
    content = file.read()
content = content.replace(',', '')
with_background4_str = content.split()
with_background4 = [int(x) for x in with_background4_str]

# without glas 5
```

```
with open('C:\\Users\\shiy0\\OneDrive\\Dokumente\\FP\\F 80␣
 ↪Scintillators\\FP_81_Shi\\Sr90_without_glas5.dat', 'r') as file:
    content = file.read()
content = content.replace(',', '')
with_background5_str = content.split()
with_background5 = [int(x) for x in with_background5_str]
```

[78]:
```
# mean value of 5 measurements
with_background_sum = [a + b + c + d for a, b, c, d in zip(with_background1,␣
 ↪with_background2, with_background3, with_background4)]
with_background=[int(x/5) for x in with_background_sum]
print(with_background)

dN_dE = np.sqrt(np.array([m-n for m,n in zip(with_background, background)])**2)
print(dN_dE)
data=np.linspace(0, 100, len(dN_dE))      # channel uniformly distributed
print(data)
```

```
[37000, 57600, 4400, 5400, 400, 0, 200, 400, 0, 200, 0, 600, 200, 0, 400, 600,
600, 200, 200, 800, 0, 400, 0, 200, 600, 600, 600, 400, 600, 400, 800, 200, 200,
400, 400, 200, 600, 800, 1000, 200, 600, 400, 600, 400, 400, 200, 1200, 600,
1200, 2000, 9200, 60800, 81600, 105800, 361400, 3303200, 4290600, 2897000,
1257800, 2928400, 543200, 295400, 104000, 72400, 41200, 30200, 26200, 36200,
30000, 22200, 18000, 31200, 21800, 19200, 18000, 27600, 20800, 15800, 17000,
24200, 17200, 18200, 18200, 24200, 14600, 16600, 15600, 22000, 12200, 13200,
13600, 16200, 13400, 14600, 11800, 12200, 13800, 13000, 9800, 15000, 11600,
10600, 10200, 15400, 11000, 11800, 12400, 13000, 11200, 10800, 11400, 11800,
14200, 10600, 7800, 13600, 13400, 11600, 10400, 9000, 11600, 10600, 6600, 11600,
10600, 10000, 6800, 8800, 10000, 11200, 10400, 9000, 7400, 7400, 8800, 10400,
7600, 8600, 4800, 8000, 8000, 7600, 6800, 8400, 8800, 8000, 8200, 9600, 7600,
7400, 7200, 8800, 5800, 9200, 6600, 7200, 6800, 8600, 6800, 6200, 5800, 5400,
6400, 6600, 4400, 7200, 8200, 5800, 6200, 5800, 6000, 6200, 7800, 4000, 6400,
6800, 5800, 5000, 6000, 6400, 4800, 6400, 4600, 8800, 5600, 6800, 4800, 5000,
4400, 5600, 6000, 5200, 3600, 4200, 3800, 4600, 4600, 6000, 3400, 5400, 5000,
2400, 4400, 4400, 3600, 5800, 4200, 4600, 4400, 4000, 4600, 3800, 3800, 3600,
2400, 4200, 6400, 2400, 3800, 3200, 4800, 2800, 5200, 4600, 3400, 3800, 3800,
4400, 4400, 2000, 2800, 3400, 3400, 3200, 4400, 3400, 3400, 3200, 2600, 4600,
2400, 2000, 2600, 3600, 2200, 3000, 1600, 3200, 3200, 2600, 2400, 3800, 1200,
2800, 2200, 3600, 2000, 2200, 2200, 4600, 2200, 1600, 2800, 1800, 1400, 3200,
2600, 1800, 2400, 2400, 2200, 3800, 1800, 2000, 1400, 1600, 1200, 2600, 2000,
1400, 1400, 1400, 3200, 4000, 2800, 1600, 2600, 1400, 2200, 2000, 1200, 2200,
1200, 1600, 1400, 3000, 1400, 1800, 2000, 2400, 1800, 2000, 1800, 2400, 800,
1200, 2200, 2400, 600, 2200, 800, 1800, 2400, 1200, 2400, 1400, 2000, 1400,
1800, 2000, 600, 1200, 800, 1800, 2800, 1200, 2000, 800, 400, 1200, 1600, 1800,
1400, 800, 1200, 800, 800, 1200, 1600, 800, 1800, 1000, 1200, 600, 1400, 2000,
200, 2200, 800, 1200, 800, 1400, 1000, 1400, 1400, 1000, 200, 1200, 800, 1800,
800, 1400, 1000, 1200, 800, 1000, 800, 800, 800, 400, 400, 1600, 600, 600, 200,
```

1800, 600, 2000, 1600, 400, 400, 600, 400, 600, 600, 800, 400, 400, 200, 600,
400, 800, 800, 200, 200, 800, 600, 400, 0, 600, 400, 600, 200, 400, 600, 600, 0,
0, 400, 0, 0, 200, 0, 0, 800, 200, 400, 200, 0, 600, 200, 200, 400, 200, 0, 400,
200, 600, 600, 200, 200, 600, 0, 400, 200, 400, 0, 200, 200, 400, 400, 0, 200,
0, 0, 400, 0, 200, 200, 0, 200, 0, 0, 200, 0, 200, 0, 0, 0, 0, 0, 0, 0, 200, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 200, 0, 0,
0, 0, 0, 0, 0, 0, 0, 200, 0, 0, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200, 2600, 0, 0, 0,
0, 0, 0, 0, 200, 0, 0, 0, 200, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 200, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 200, 0, 0, 0, 0, 0, 200, 0, 0, 0, 0, 0, 0, 0, 200, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 200, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 200, 0, 0, 0, 0, 0, 200, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 200, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

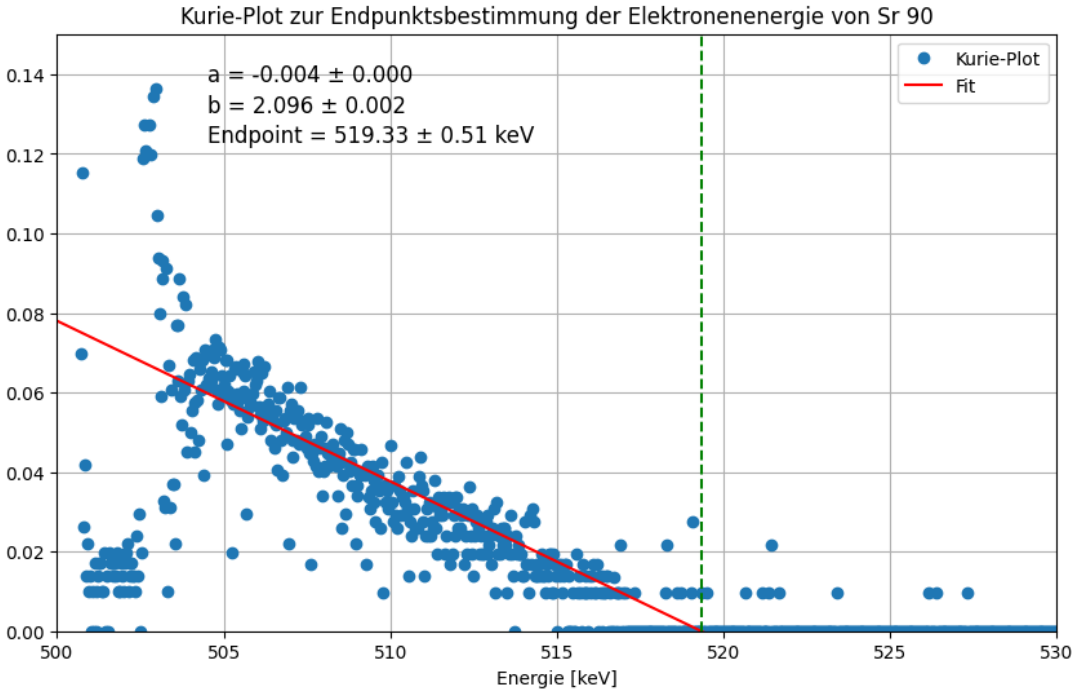
```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[10000. 27400.  1400. …     0.     0.     0.]
[0.00000000e+00 2.44259893e-02 4.88519785e-02 … 9.99511480e+01
 9.99755740e+01 1.00000000e+02]
```

```
C:\Users\shiy0\AppData\Local\Temp\ipykernel_14156\670158964.py:6:
RuntimeWarning: invalid value encountered in sqrt
  dN_dE = np.sqrt(np.array([m-n for m,n in zip(with_background,
background)])**2)
```

[79]:
```
# constants
k = 1 / 681 * 1000      # energy in keV
m = 341 / 681 * 1000
m_e = 511
alpha = 1 / 137.
Z = 38
pi = np.arccos(-1.)

# Coulomb correction
def F(x):
    term = (Z * (x + m_e)) / (1 / alpha * np.sqrt(x**2 + 2 * m_e * x))
    exp_term = np.exp(-2 * pi * (38 * (x + m_e) / (1 / alpha * np.sqrt(x**2 + 2␣
 ↪* m_e * x))))
    return (2 * pi * term) / (1 - exp_term)

x_data = data * k + m        # energy calibration
y_data = np.sqrt(dN_dE / (F(x_data) * (x_data + m_e) * np.sqrt(x_data**2 + 2 *␣
 ↪x_data * m_e)))  # Kurie-Plot y value

# Remove NaN and Inf values from y_data and corresponding x_data
valid_indices = np.isfinite(y_data) # Find valid (non-NaN and non-Inf) indices
x_data = x_data[valid_indices]       # Filter x,y_data based on valid indices
y_data = y_data[valid_indices]
print(y_data)

popt, pcov = curve_fit(linear, x_data[50:600], y_data[50:600], p0=(-0.0056, 0.
 ↪16))
a=popt[0]
b=popt[1]
da=pcov[0][0]
db=pcov[1][1]
print('a=', popt[0], '+-', pcov[0][0])
print('b=', popt[1], '+-', pcov[1][1])

# Endpoint x0=-b/a, gauss-error propagation
E0 = -b / a
sigma_E0 = np.sqrt((db / a)**2 + (b * da / a**2)**2)
print('E0=', E0, '+-', sigma_E0)


# Plotting
plt.figure(figsize=(10, 6))
```

```
plt.plot(x_data, y_data, 'o', label='Kurie-Plot')
x_range=np.linspace(490, 650, 500)
plt.plot(x_range, linear(x_range, *popt), 'r-', label='Fit')
plt.title('Kurie-Plot zur Endpunktsbestimmung der Elektronenenergie von Sr 90')
plt.xlabel('Energie [keV]')

plt.text(0.15, 0.95, f'a = {a:.3f} ± {da:.3f}', transform=plt.gca().transAxes,␣
 ↪fontsize=12, verticalalignment='top')
plt.text(0.15, 0.90, f'b = {b:.3f} ± {db:.3f}', transform=plt.gca().transAxes,␣
 ↪fontsize=12, verticalalignment='top')
plt.text(0.15, 0.85, f'Endpoint = {E0:.2f} ± {sigma_E0:.2f} keV', transform=plt.
 ↪gca().transAxes, fontsize=12, verticalalignment='top')
plt.axvline(x=E0, color='green', linestyle='--')

plt.xlim(500, 530)
plt.ylim(0, np.max(y_data) * 1.1)

plt.legend()
plt.grid(True)
plt.show()
```

```
[0.0697242  0.11540979 0.02608646 … 0.         0.         0.        ]
a= -0.004035449469224716 +- 7.791182708929652e-09
b= 2.095718080713691 +- 0.002047063291565077
E0= 519.3270530819748 +- 0.5072711988965813
```



[80]:
```
# check
print("NaN in y_data:", np.any(np.isnan(y_data)))
print("Inf in y_data:", np.any(np.isinf(y_data)))

print(len(x_data))
print(len(y_data))
```

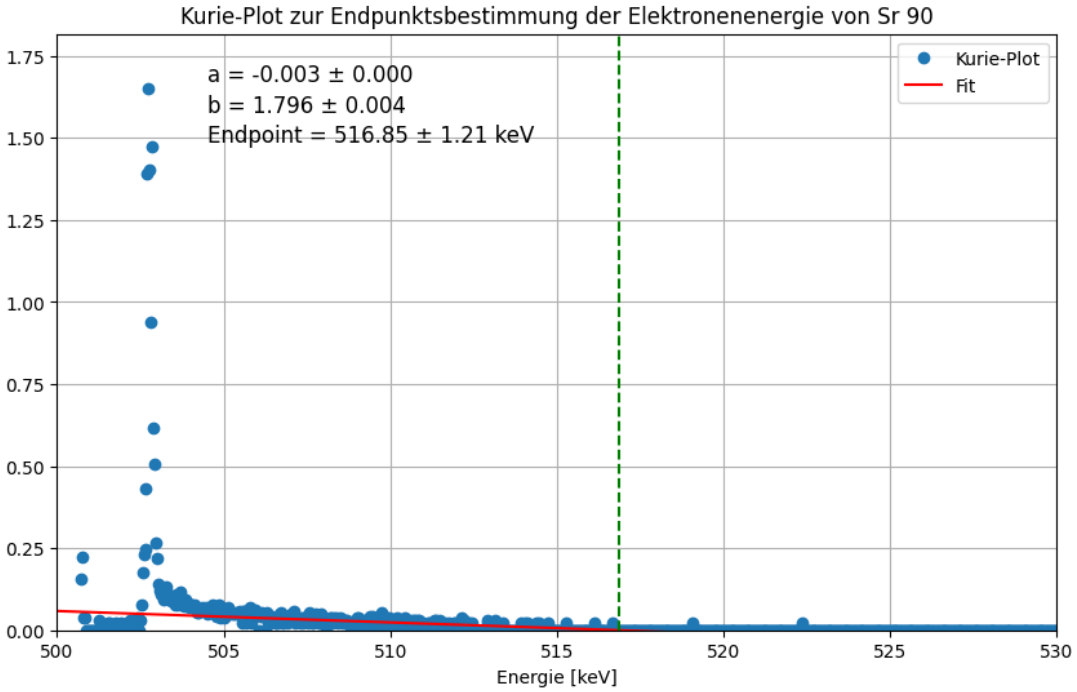
```
NaN in y_data: False
Inf in y_data: False
4090
4090
```

Extrapolation

[81]:
```
# Aluminium 0.5 mm +1 mm
with open('C:\\Users\\shiy0\\OneDrive\\Dokumente\\FP\\F 80␣
 ↪Scintillators\\FP_81_Shi\\Sr90_0.5mm_raw.dat', 'r') as file:
    content = file.read()
content = content.replace(',', '')
Al1_str = content.split()
Al1 = np.array([int(x) for x in Al1_str])

print(len(Al1))
data=np.linspace(0, 100, len(Al1))      # channel uniformly distributed
x_data1 = data * k + m         # energy calibration
y_data1 = np.sqrt(Al1 / (F(x_data1) * (x_data1 + m_e) * np.sqrt(x_data1**2 + 2␣
 ↪* x_data1 * m_e)))  # Kurie-Plot y value

valid_indices = np.isfinite(y_data1)
x_data1 = x_data1[valid_indices]
y_data1 = y_data1[valid_indices]

popt, pcov = curve_fit(linear, x_data1[100:550], y_data1[100:550], p0=(-0.0056,␣
 ↪0.16))
a=popt[0]
b=popt[1]
da=pcov[0][0]
db=pcov[1][1]

# Endpoint x0=-b/a, gauss-error propagation
E0 = -b / a
sigma_E0 = np.sqrt((db / a)**2 + (b * da / a**2)**2)
print('E0=', E0, '+-', sigma_E0)

# Plotting
plt.figure(figsize=(10, 6))
```

```
plt.plot(x_data1, y_data1, 'o', label='Kurie-Plot')
x_range=np.linspace(490, 650, 500)
plt.plot(x_range, linear(x_range, *popt), 'r-', label='Fit')
plt.title('Kurie-Plot zur Endpunktsbestimmung der Elektronenenergie von Sr 90')
plt.xlabel('Energie [keV]')

plt.text(0.15, 0.95, f'a = {a:.3f} ± {da:.3f}', transform=plt.gca().transAxes,␣
 ↪fontsize=12, verticalalignment='top')
plt.text(0.15, 0.90, f'b = {b:.3f} ± {db:.3f}', transform=plt.gca().transAxes,␣
 ↪fontsize=12, verticalalignment='top')
plt.text(0.15, 0.85, f'Endpoint = {E0:.2f} ± {sigma_E0:.2f} keV', transform=plt.
 ↪gca().transAxes, fontsize=12, verticalalignment='top')
plt.axvline(x=E0, color='green', linestyle='--')

plt.xlim(500, 530)
plt.ylim(0, np.max(y_data1) * 1.1)

plt.legend()
plt.grid(True)
plt.show()
```

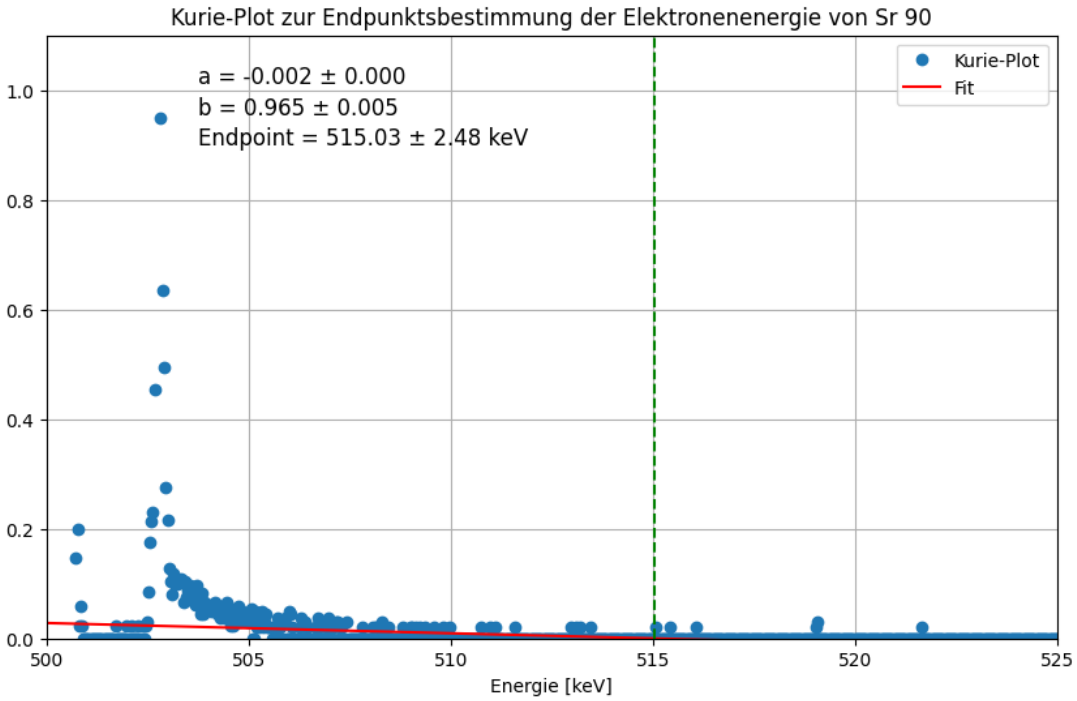
```
4095
E0= 516.8476999174222 +- 1.2149653516476433
```



[82]:
```
# Aluminium 1.5 mm + 1 mm
with open('C:\\Users\\shiy0\\OneDrive\\Dokumente\\FP\\F 80␣
 ↪Scintillators\\FP_81_Shi\\Sr90_1.5mm_raw.dat', 'r') as file:
    content = file.read()
content = content.replace(',', '')
Al2_str = content.split()
Al2 = np.array([int(x) for x in Al2_str])

x_data2 = data * k + m        # energy calibration
y_data2 = np.sqrt(Al2 / (F(x_data2) * (x_data2 + m_e) * np.sqrt(x_data2**2 + 2␣
 ↪* x_data2 * m_e)))  # Kurie-Plot y value

valid_indices = np.isfinite(y_data1)
x_data2 = x_data2[valid_indices]
y_data2 = y_data2[valid_indices]

popt, pcov = curve_fit(linear, x_data2[100: 500], y_data2[100: 500], p0=(-0.
 ↪0056, 0.16))
a=popt[0]
b=popt[1]
da=pcov[0][0]
db=pcov[1][1]

# Endpoint x0=-b/a, gauss-error propagation
E0 = -b / a
sigma_E0 = np.sqrt((db / a)**2 + (b * da / a**2)**2)
print('E0=', E0, '+-', sigma_E0)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(x_data2, y_data2, 'o', label='Kurie-Plot')
x_range=np.linspace(490, 650, 500)
plt.plot(x_range, linear(x_range, *popt), 'r-', label='Fit')
plt.title('Kurie-Plot zur Endpunktsbestimmung der Elektronenenergie von Sr 90')
plt.xlabel('Energie [keV]')

plt.text(0.15, 0.95, f'a = {a:.3f} ± {da:.3f}', transform=plt.gca().transAxes,␣
 ↪fontsize=12, verticalalignment='top')
plt.text(0.15, 0.90, f'b = {b:.3f} ± {db:.3f}', transform=plt.gca().transAxes,␣
 ↪fontsize=12, verticalalignment='top')
plt.text(0.15, 0.85, f'Endpoint = {E0:.2f} ± {sigma_E0:.2f} keV', transform=plt.
 ↪gca().transAxes, fontsize=12, verticalalignment='top')
plt.axvline(x=E0, color='green', linestyle='--')

plt.xlim(500, 525)
plt.ylim(0, 1 * 1.1)
```

```
plt.legend()
plt.grid(True)
plt.show()
```

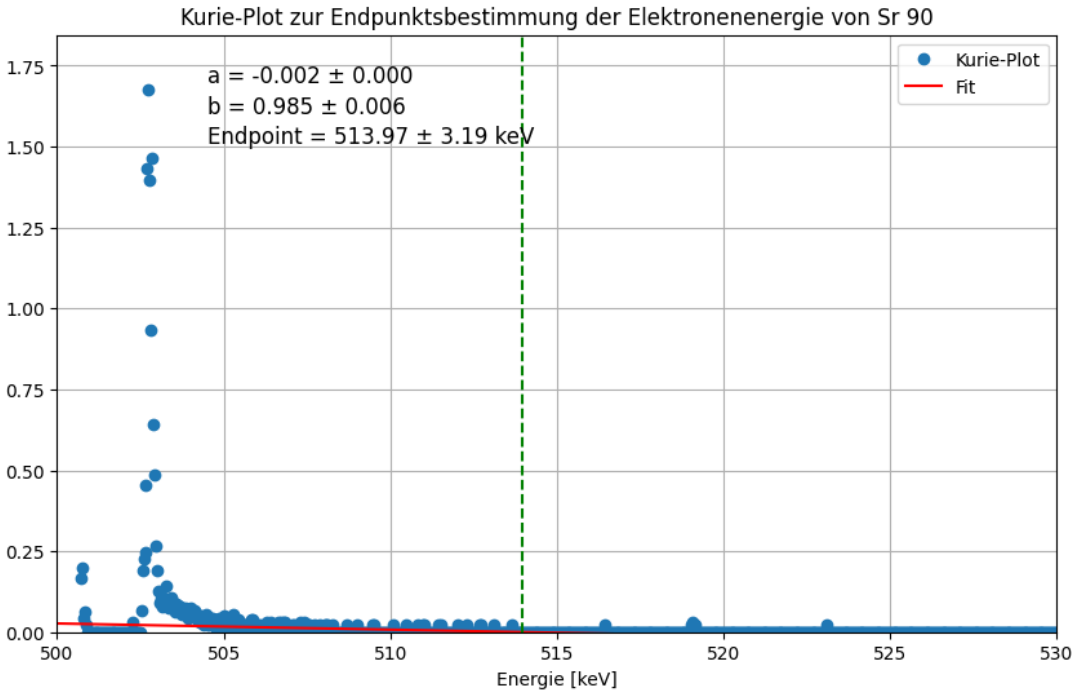
```
E0= 515.0321254553097 +- 2.4836119226758
```



[83]:
```
# Aluminium 2.5 mm + 1 mm
with open('C:\\Users\\shiy0\\OneDrive\\Dokumente\\FP\\F 80␣
 ↪Scintillators\\FP_81_Shi\\Sr90_2.5mm_raw.dat', 'r') as file:
    content = file.read()
content = content.replace(',', '')
Al3_str = content.split()
Al3 = np.array([int(x) for x in Al3_str])

x_data3 = data * k + m        # energy calibration
y_data3 = np.sqrt(Al3 / (F(x_data3) * (x_data3 + m_e) * np.sqrt(x_data3**2 + 2␣
 ↪* x_data3 * m_e)))  # Kurie-Plot y value

valid_indices = np.isfinite(y_data1)
x_data3 = x_data3[valid_indices]
y_data3 = y_data3[valid_indices]

popt, pcov = curve_fit(linear, x_data3[100: 450], y_data3[100: 450], p0=(-0.
 ↪0056, 0.16))
```

```
a=popt[0]
b=popt[1]
da=pcov[0][0]
db=pcov[1][1]

# Endpoint x0=-b/a, gauss-error propagation
E0 = -b / a
sigma_E0 = np.sqrt((db / a)**2 + (b * da / a**2)**2)
print('E0=', E0, '+-', sigma_E0)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(x_data3, y_data3, 'o', label='Kurie-Plot')
x_range=np.linspace(490, 650, 500)
plt.plot(x_range, linear(x_range, *popt), 'r-', label='Fit')
plt.title('Kurie-Plot zur Endpunktsbestimmung der Elektronenenergie von Sr 90')
plt.xlabel('Energie [keV]')

plt.text(0.15, 0.95, f'a = {a:.3f} ± {da:.3f}', transform=plt.gca().transAxes,␣
 ↪fontsize=12, verticalalignment='top')
plt.text(0.15, 0.90, f'b = {b:.3f} ± {db:.3f}', transform=plt.gca().transAxes,␣
 ↪fontsize=12, verticalalignment='top')
plt.text(0.15, 0.85, f'Endpoint = {E0:.2f} ± {sigma_E0:.2f} keV', transform=plt.
 ↪gca().transAxes, fontsize=12, verticalalignment='top')
plt.axvline(x=E0, color='green', linestyle='--')

plt.xlim(500, 530)
plt.ylim(0, np.max(y_data3) * 1.1)

plt.legend()
plt.grid(True)
plt.show()
```

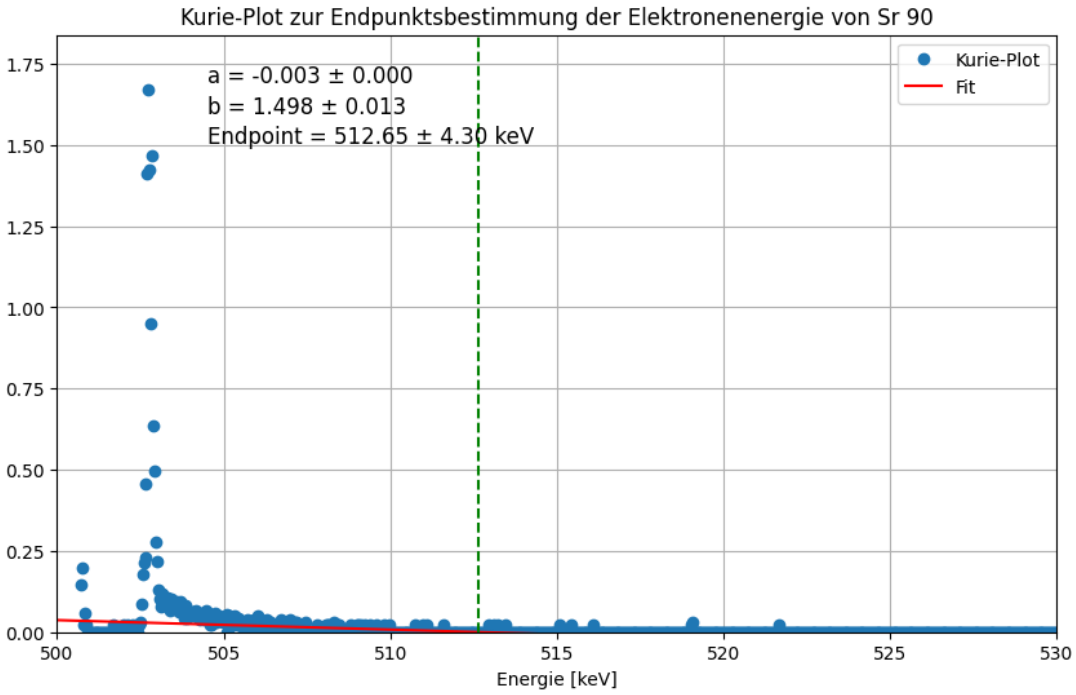
```
E0= 513.9687879701313 +- 3.192751882781536
```



[84]:
```
# Aluminium 3.5 mm + 1 mm
with open('C:\\Users\\shiy0\\OneDrive\\Dokumente\\FP\\F 80␣
 ↪Scintillators\\FP_81_Shi\\Sr90_1.5mm_raw.dat', 'r') as file:
    content = file.read()
content = content.replace(',', '')
Al4_str = content.split()
Al4 = np.array([int(x) for x in Al4_str])

x_data4 = data * k + m        # energy calibration
y_data4 = np.sqrt(Al4 / (F(x_data4) * (x_data4 + m_e) * np.sqrt(x_data4**2 + 2␣
 ↪* x_data4 * m_e)))  # Kurie-Plot y value

valid_indices = np.isfinite(y_data1)
x_data4 = x_data4[valid_indices]
y_data4 = y_data4[valid_indices]

popt, pcov = curve_fit(linear, x_data4[100: 400], y_data4[100: 400], p0=(-0.
 ↪0056, 0.16))
a=popt[0]
b=popt[1]
da=pcov[0][0]
db=pcov[1][1]

# Endpoint x0=-b/a, gauss-error propagation
```

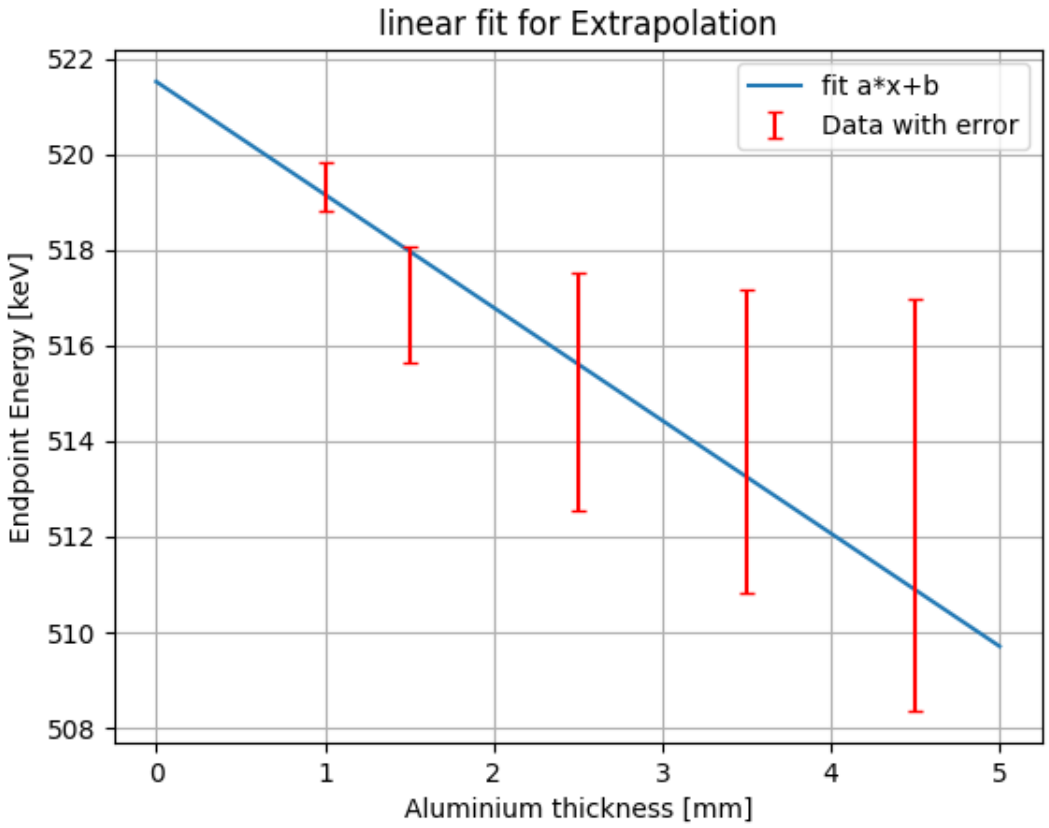
```
E0 = -b / a
sigma_E0 = np.sqrt((db / a)**2 + (b * da / a**2)**2)
print('E0=', E0, '+-', sigma_E0)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(x_data4, y_data4, 'o', label='Kurie-Plot')
x_range=np.linspace(490, 650, 500)
plt.plot(x_range, linear(x_range, *popt), 'r-', label='Fit')
plt.title('Kurie-Plot zur Endpunktsbestimmung der Elektronenenergie von Sr 90')
plt.xlabel('Energie [keV]')

plt.text(0.15, 0.95, f'a = {a:.3f} ± {da:.3f}', transform=plt.gca().transAxes,␣
↪fontsize=12, verticalalignment='top')
plt.text(0.15, 0.90, f'b = {b:.3f} ± {db:.3f}', transform=plt.gca().transAxes,␣
↪fontsize=12, verticalalignment='top')
plt.text(0.15, 0.85, f'Endpoint = {E0:.2f} ± {sigma_E0:.2f} keV', transform=plt.
↪gca().transAxes, fontsize=12, verticalalignment='top')
plt.axvline(x=E0, color='green', linestyle='--')

plt.xlim(500, 530)
plt.ylim(0, np.max(y_data4) * 1.1)

plt.legend()
plt.grid(True)
plt.show()
```

```
E0= 512.6486795136873 +- 4.303726999818204
```



[86]:
```
# Extrapolation linear Fit
Al=np.array([1, 1.5, 2.5, 3.5, 4.5])     # mm
E=np.array([519.33, 516.85, 515.03, 513.99, 512.65])    # keV
dE=np.array([0.51, 1.22, 2.48, 3.19, 4.30])

popt, pcov=curve_fit(linear, Al, E, sigma=dE, absolute_sigma=True)
print('a=', popt[0], '+-', pcov[0][0])
print('b=E(d=0mm):', popt[1], '+-', pcov[1][1])

range=np.linspace(0, 5, 100)
plt.plot(range, linear(range, *popt), label='fit a*x+b')
plt.errorbar(Al,E, yerr=dE, fmt='None', ecolor='r', capsize=3, marker='s',␣
 ↪mec='black', mfc='black', ms='2', label='Data with error')
plt.xlabel('Aluminium thickness [mm]')
plt.ylabel('Endpoint Energy [keV]')
plt.title('linear fit for Extrapolation')

plt.legend()
plt.grid(True)
plt.show()
```

```
a= -2.363450684148429 +- 0.6260318374755037
b=E(d=0mm): 521.5168990087611 +- 1.1236003020837508
```


linear fit for Extrapolation
fit a*x+b
Data with error
Endpoint Energy [keV]
Aluminium thickness [mm]
522
520
518
516
514
512
510
508
0
1
2
3
4
5

[ ]:

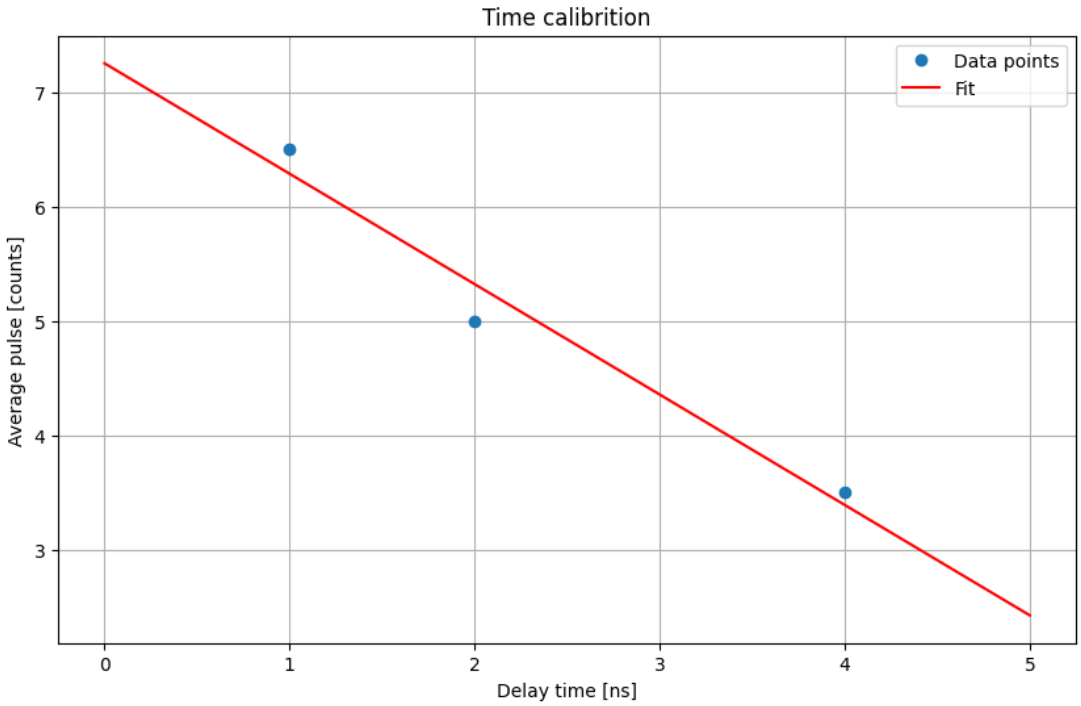
Python F81

August 23, 2024

0.0.1 Part II: Time calibration

[8]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

[9]:
```
pulse1 = np.array([13.000, 10.000, 10.000, 6.000, 3.000, 2.000, 1.000, 7.000,␣
 ↪11.000, 2.000])
pulse2 = np.array([6.000, 7.000, 3.000, 7.000, 3.000, 4.000, 4.000, 7.000, 8.
 ↪000, 1.000])
pulse3 = np.array([12.000, 1.000, 3.000, 5.000, 2.000, 1.000, 4.000, 2.000, 4.
 ↪000, 1.000])
#pulse4 = np.array([8.000, 7.000, 6.000, 4.000, 5.000, 0.000, 3.000, 7.000, 5.
 ↪000])
pulse = np.array([np.mean(pulse1), np.mean(pulse2), np.mean(pulse3)])
delay = np.array([1, 2, 4])

def linear_fit(x, a, b):
    return a * x + b

popt, pcov = curve_fit(linear_fit, delay, pulse)
x_fit = np.linspace(0, 5, 100)
plt.figure(figsize=(10, 6))
plt.plot(delay, pulse, 'o', label='Data points')
plt.plot(x_fit, linear_fit(x_fit, *popt), 'r-', label='Fit')
plt.title('Time calibrition')
plt.xlabel('Delay time [ns]')
plt.ylabel('Average pulse [counts]')
plt.legend()
plt.grid(True)
plt.show()

print("a =", popt[0], "+/-", pcov[0,0]**0.5)
print("b =", popt[1], "+/-", pcov[1,1]**0.5)
```




```
a = -0.9642857142899983 +/- 0.1855768722407518
b = 7.250000000013628 +/- 0.49099025303297683
```