Evaluation

December 15, 2025

[1]:
```python
import numpy as np
import pandas as pd
import yaml
from IPython.display import display, Math
from uncertainties import ufloat
```

1 Exercise 1: Calibration

The calibration data and a Gaussian fit are used to get the peak position and the calibration factor.
The results are:
Peak position: 412.54
Energy calibration factor: 1.2387 keV/unit

This is used to process all further data.

2 Exercise 2: Parity violation

The processed data of the positive and the negative setting of the magnetic field is examined to determine wether parity is violated.

[2]:
```python
# load the yaml file
with open('DataFolder/processed_files.yaml', 'r') as file:
    data = yaml.safe_load(file)['results_df']

# get the measurement values
val = [data[:][i]['Entries_Above_Threshold'] for i in range(len(data))]
val_p = val[:30] # positive field
val_m = val[30:] # negative field
sum_p = sum(val_p)
sum_m = sum(val_m)
p = ufloat(sum_p, np.sqrt(sum_p))
m = ufloat(sum_m, np.sqrt(sum_m))

# show the results
display(Math(rf'\text{{positive: }} {p.n:.0f} \pm {p.s:.0f} \text{{ entries␣
↪above treshold}}'))
```

```
display(Math(rf'\text{{negative: }}  {m.n:.0f} \pm {m.s:.0f} \text{{ entries␣
    ↪above treshold}}'))
display(Math(rf'\text{{mean difference: }}  {p.n - m.n} \text{{ entries}}'))
```

positive: 346388 ± 589 entries above treshold

negative: 336719 ± 580 entries above treshold

mean difference: 9669.0 entries

Here the deviation is the square root of the values.

The difference is way larger than the deviation. That means it can be assumed that the parity is violated on β-decay.

2.1 2.1: Comparison of errors

To compare the errors of different estimations we use the value of the asymmetry $E = \frac{N_+ - N_-}{N_+ + N_-}$

One method uses the squareroot of the sums N_+ and N_-
The other one estimates E for each pair of values N_+ and N_- with their statistical deviation and the mean deviation $\frac{\sigma}{\sqrt{N}}$ where N is the amount of pairs of values.

[3]:
```
E1 = (p - m) / (p + m)
E2 = [(val_p[i] - val_m[i]) / (val_p[i] + val_m[i]) for i in range(30)]
E2 = ufloat(np.mean(E2), np.std(E2) / np.sqrt(30))

display(Math(rf'\text{{For the sum: }} E_{{Sum}} = {E1.n:.6f}\pm{E1.s:.6f}'))
display(Math(rf'\text{{For each pair of values: }} E_{{Pairs}} = {E2.n:.
    ↪6f}\pm{E2.s:.6f}'))
```

For the sum: $E_{Sum} = 0.014154 \pm 0.001210$

For each pair of values: $E_{Pairs} = 0.014184 \pm 0.001185$

So its almost the same. For this evaluation the values of the first method are used.

3 Exercise 3: Degree of Polarisation of γ

The polarisation P_C of the bremsstrahlung is determined using this formula:

$$E = \frac{N_+ - N_-}{N_+ + N_-} = f P_C \frac{\phi_C^-}{\phi_0}$$

$$\Rightarrow P_C = \frac{N_+ - N_-}{N_+ + N_-} \frac{\phi_0}{\phi_C^-} \frac{1}{f}$$

With
$f = \frac{1}{13}$
$\frac{\phi_C^-}{\phi_0} = 0.52 \pm 0.05$

It follows:

[4]:
```python
E = E1
phi_ratio = ufloat(0.52, 0.05)
f = 1 / 13

P_C = E / (phi_ratio * f)

display(Math(rf'P_C = {P_C.n:.3} \pm {P_C.s:.2}'))
```

$P_C = 0.354 \pm 0.046$

So the degree of polarisation is about $35\% - 36\%$ for the bremsstrahlung.

4 Exercise 4: Longitudinal Polarisation of e^-

Here the longitudinal polarisation is the same as the helicity of the electrons ($\rightarrow$ blue book).
The helicity is calculated as follows:

$$H = \frac{P_C}{L}$$

where L is the helicity transfer. It's value is taken from the $2.5 MeV$ curve in figure 7.7 in the blue book (Note to Shivam: Could you link this to the figure in the protocol if it's included? Thank you)
To get the value the ratio of the Energies is needed.
$E_\gamma \geq 1 MeV$ as this was the criterium in the first place.
$E_{e,max} = 2.28 MeV$ (source: https://en.wikipedia.org/wiki/Yttrium-90, please add this to the references)
Only the kinetic energy is transferred. It follows: $E_{kin,max} = E_{e,max} - m_e c^2 = 2.28 MeV - 0.51 MeV = 1.77 MeV$
That means the minimum of $\frac{E_\gamma}{E_{kin}}$ is $\frac{1}{1.77} = 0.56$.
It follows: $0.56 \leq L \leq 0.92$, $L = 0.74 \pm 0.18$.

So the longitudinal polarisation is:

[5]:
```python
L = ufloat(0.74, 0.18)
H = P_C / L
display(Math(rf'H = {H.n:.3f} \pm {H.s:.3f}'))
```

$H = 0.478 \pm 0.132$