Contents

1. Theoretic Background

In this section, a brief description of the theoretical concepts required to understand the Mößbauer effect and its application experiment will be given. If not stated otherwise, the discussion is based on the Instructions [10, 14].

1.1. Interaction of Electromagnetic Radiation with Matter

In this experiment, gamma radiation is used to investigate the spectrum of natural iron and stainless steel. Gamma radiation is high-energy electromagnetic radiation which can be caused by radioactive decay of excited atomic nuclei. The background caused by interaction of electromagnetic radiation with matter has to be taken into account when evaluating this experiment. The main contributions to the interaction of electromagnetic radiation with matter are described by the photoelectric effect, Compton scattering and pair creation, the mechanisms of which will be briefly sketched in the following.

Photoelectric Effect A photon transfers all of its energy to an electron of an atom, which then has enough energy to leave the atom. It is left with a kinetic energy of $E_{\text{kin}} = hf - E_{\text{bin}}$, where hf is the energy of the photon and E_{bin} is the binding energy of the electron. In this process, the photon is completely absorbed by the electron. This is only possible with bound electrons, as the existence of the nucleus is necessary for the conservation of momentum. As the probability for the photoelectric effect is proportional to the fifth power of the nuclear charge number, this effect mainly occurs in heavy elements.

Compton Scattering Compton Scattering occurs when a photon and a free electron scatter. During this process, the photon transfers part of its energy to the electron and as a result experiences red shifting. The energy of the photon after the interaction depends on the angle at which the scattering occurs.

Pair Creation If a photon has an energy of at least 1022 keV, it can transform all of its energy into a electron-positron-pair. The energy is converted to the rest mass of the electron and the positron (511 keV each).

Attenuation of Gamma Radiation The absorbers used in this experiment are enclosed in acrylic glass. Therefore, the absorption of gamma radiation in acrylic glass also has to be taken into account. The ratio of the intensity I and the initial intensity I_0 is given by

$$R := \frac{I}{I_0} = \exp\left(-\frac{\mu}{\rho} \cdot \rho d\right),$$

where μ/ρ is the mass attenuation coefficient, ρ is the density of the absorber material and d is the thickness of the absorber.

In this experiment, the focus is on the 14.4 keV emission line of iron. The other main emission lines are at 122.2 keV and 136.6 keV [10]. As the threshold for pair creation is 1022 keV, pair creation does not contribute to the background in this experiment. A photon interacting via the photoelectric effect is completely absorbed. It can not be detected by the scintillator and thus the photoelectric effect does not contribute to the background in this experiment as well. The dominant contribution is therefore Compton scattering. This can also clearly be seen in Fig. 1, in which the absorption coefficients due to interaction with matter are shown at various photon energies.

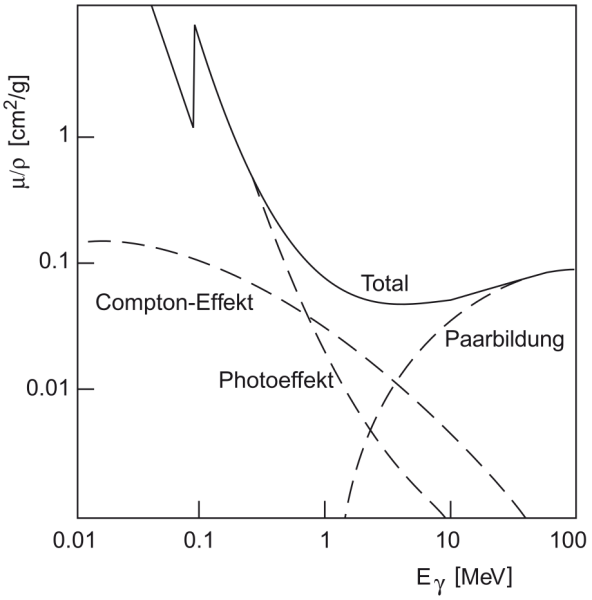


Figure 1: The photon absorption coefficient μ/ρ in terms of photon energy E_γ taken from [7]. The contribution to the total absorption coefficient of the three main effects, which are Compton scattering, photoelectric effect and pair production, is pictured separately. At the energy range of interest in this experiment, 14 keV to 136 keV, pair production does not contribute.

1.2. The Doppler Effect

The relativistic Doppler effect describes the change of frequency of light that occurs when a source and a receiver are moving relative to each other. The frequency f_r observed by the receiver is given by

$$f_r = \sqrt{\frac{1-\frac{v}{c}}{1+\frac{v}{c}}}\, f_s\,, \tag{1}$$

where f_s is the frequency emitted by the source and v is the relative velocity of source and receiver. Expanding Eq. (1) to first order in v/c, one finds

$$f_r \approx \left(1-\frac{v}{c}\right) f_s\,. \tag{2}$$

Inserting the energy for a photon, $E_\gamma = hf$, where h is the Planck constant, one finds that the frequency shift induced by the relativistic Doppler effect leads to a shift in energy given by

$$E_{\gamma,r} \approx \left(1 - \frac{v}{c}\right) E_{\gamma,s} \,. \tag{3}$$

1.3. The Mößbauer Effect

The main idea of this experiment is to use the 14.4 keV emission line of an excited state of ^{57}Fe for spectroscopy. That is why it is important to understand the effects that can lead to shifts in the energy of the emitted photons.

Recoil in a classical two-body decay The simplest way to model the decay of an excited nucleus is by assuming that it is confined within a large rigid lattice. In the initial state, the lattice is at rest and has an internal energy of E_2 due to the excited nucleus. In the final state, the rigid body has lowered its energy to E_1. It now has a momentum, $p = mv$, where m is the mass of the rigid body and v is its final state velocity. Additionally, a photon with energy $E_\gamma = hf$ and momentum $p_\gamma = hf/c$ was emitted. Using conservation of momentum and energy,

$$0 = \frac{hf}{c} + mv \qquad \text{and} \qquad E_2 = E_1 + E_\gamma + \frac{mv^2}{2} \,,$$

the energy of the photon can be calculated:

$$E_\gamma = E_2 - E_1 - \frac{h^2 f^2}{2mc^2} =: E_0 + E_R \,, \quad \text{where} \quad E_0 := E_2 - E_1 \,. \tag{4}$$

From Eq. (4), it is immediately clear that the energy of the photon is not equal to the energy difference E_0. This is due to the recoil experienced by the rigid body, causing it to no longer be at rest. The energy loss due to recoil can be estimated using the molecular mass $m_{\text{iron}} = (55.845 \pm 0.002)\,\text{g mol}^{-1}$ [13]. For one iron atom, $m_{\text{iron}} \approx 52\,\text{GeV}$, resulting in $E_R \approx 2\,\text{meV}$, while the energy loss due to recoil for one mol of iron is $E_R \approx 3.3{\cdot}10^{-27}\,\text{eV}$. As in this experiment, the 14.4 keV line of iron is used, $E_0 = 14.4\,\text{keV}$. Since $E_0 \gg E_R$, the energy lost to nuclear recoil is negligible in this case.

However, recoil is not the only effect that can lead to energy loss. The model of a rigid body for the iron source considers only classical particles and neglects interaction between the atoms in the lattice.

The Law of Dulong-Petit The law of Dulong Petit is an experimentally found approximation for the behavior of the heat capacity of a solid at high temperatures. In the following, several models which aim to formulate a theoretical derivation for the heat capacity of a solid body will be discussed.

Einstein Model The Einstein model describes a solid as a lattice of independent particles bound by an external potential which for all atoms has the same eigenfrequency. The potential the atoms are bound in is a quantized harmonic potential. The Einstein model recovers the law of Dulong-Petit at temperatures that are large in comparison to the eigenfrequency. However, at lower temperatures, the effects caused by the interaction of the atoms can no longer be neglected, and the predictions of the Einstein model do not match experimental findings anymore.

Debye Model The Debye model takes the correlated movement of atoms into account. In analogy to Planck's law of black body radiation, which treats electromagnetic radiation as photons in a box, the Debye model describes vibrations of the atomic lattice as phonons in a box. Since the propagation medium of a phonon is the atomic lattice, and there is only a finite amount of vibrating atoms in the lattice, a phonon can not have an arbitrarily high frequency. The maximal frequency can be fixed by requiring the Dulong-Petit law for the heat capacity in the high-temperature limit. The Debye model also reproduces the experimentally found low-temperature dependency of the heat capacity.

Debye-Waller factor In this experiment, photons emitted through a nuclear decay of iron are used for spectroscopy. The previous sections have discussed effects that can shift the frequency of these photons and thus broaden the emitted line. For the purpose of spectroscopy, the emitted line should be as narrow as possible. Therefore, it is advantageous to use spectral lines originating from an emission that does not cause lattice vibrations. The occurrence of such spectral lines is called the Mößbauer effect. The ratio of emissions that do not cause lattice vibrations quantifies the Mößbauer effect and is called Debye-Waller factor. The Debye-Waller factor of ^{57}Fe and ^{187}Re as functions of temperature are pictured in Fig. 2.

The Debye-Waller factor of the source can be computed by

$$f_{\mathrm{S}} = \frac{\dot{N}(\infty) - \dot{N}(0)}{\dot{N}(\infty)} \left(1 - \exp\!\left(-\frac{1}{2} T_{\mathrm{A}}\right) \mathrm{J}_0\!\left(\frac{1}{2} i T_{\mathrm{A}}\right)\right)^{-1}, \tag{5}$$

as given in Ref. [5]. In the above equation, T_A is the dimensionless effective absorber thickness and J_0 denotes the zeroth Bessel function.

1.4. Nuclear Transitions

Natural Linewidth of Spectral Lines The population of an excited state follows an exponential distribution,

$$N(t) = N(0) e^{-\Gamma t}, \tag{6}$$

where $N(t)$ denotes the population of the excited state at a given time t and $\tau = \hbar/\Gamma$ is the lifetime of the excited state. The frequencies then follow a Breit-Wigner distribution, the Full-Width-Half-Maximum (FWHM) of which is given by Γ. Γ is called the natural decay width.

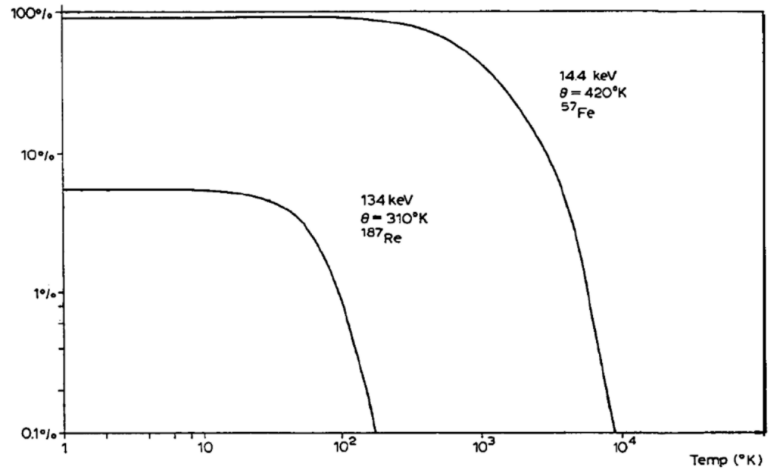


Figure 2: Fractions of recoil-free nuclear transitions (Debye-Waller factors) in ^{57}Fe and ^{187}Re, shown as functions of the temperature, taken from [14]. At room temperature ($\approx 294\,\mathrm{K}$), the Debye-Waller factor of ^{57}Fe used in this experiment is near 1, corresponding to a large fraction of recoil-free nuclear transitions in the source.

Broadening of Spectral Lines Due to thermal movement of the atoms, the frequencies of the emitted photons are Doppler shifted. This leads to a broadening of the spectral line. Since the Doppler shift only depends on the velocity component in the direction of the detector, its distribution is Gaussian. The frequency shift caused by scattering in the absorber can also be modeled by a Gaussian.

The distribution of the sum of two independent random variables is given by the convolution of the distribution of the variables. Thus, the spectral lines are distributed via the convolution of a Breit-Wigner distribution and a Gaussian distribution. This convolution is called Voigt-profile. It is important to note that the lifetime of a state is related to the natural linewidth and not the width of the convolution. It can be estimated by fitting a Voigt-profile to the data and extracting the parameter Γ of the original Breit-Wigner distribution. The lifetime is then given by $\hbar/\Gamma$.

1.5. Isomer Shift

The energy levels of an atomic nucleus are influenced by exterior electric and magnetic fields. This includes the fields caused by the electrons surrounding the nucleus. For atoms confined in a lattice, the electron density for an atom may vary slightly, causing a shift in the energy level. If the chemical composition of source and absorber slightly differs (such as in the case of an iron emitter and a stainless steel absorber), the energy difference to the excited states of source and absorber is shifted. This is called isomer shift, and since it depends on the chemical composition, it is also called chemical shift.

1.6. Hyperfine Splitting

This subsection is based on the instructions [10] as well as Ref. [12]. An external magnetic field, such as the one caused by electrons surrounding the nucleus, lifts energy degeneracies of nuclear states. States with the same angular momentum given by the quantum number I but different quantum numbers m_I, which is the projection of the angular momentum on the z-axis, now have slightly different energies. This splitting is referred to as hyperfine splitting. The energy shift is given by

$$E_{\text{hyp}} = -\frac{\mu m_I B}{I},$$

where μ is the magnetic moment and B is the absolute value of the surrounding magnetic field. Therefore, the energy a photon must have to excite a state specified by (E_1, I_1, $m_{I,1}$) to a state (E_2, I_2, $m_{I,2}$) is given by

$$E_\gamma = (E_2 - E_1) + E_{\text{iso}} - \left(\frac{\mu_2 m_{I,2}}{I_2} - \frac{\mu_1 m_{I,1}}{I_1}\right) B\,, \tag{7}$$

where E_1 and E_2 are the energies of the degenerate states corresponding to I_1 and I_2. E_{iso} denotes the relative isomeric shift caused by slight chemical differences between source and absorber. The selection rules (for dipole order in perturbation theory) state that transitions can only occur between states where the differences in quantum numbers I and m_I are

$$\Delta I = \pm 1 \qquad \text{and} \qquad \Delta m_I = 0, \pm 1\,.$$

In this experiment, the $I_{\text{g}} = 1/2$ state of iron is excited to $I_{\text{e}} = 3/2$. The hyperfine splitting leads to six distinct energy levels: $m_I = \pm 1/2$ for I_{g} and $m_I = \pm 1/2, \pm 3/2$ for I_{e}. The selection rules allow for six transitions, pictured in Fig. 3. The absorption lines corresponding to these transitions can be observed in this experiment. The energies of these absorption lines can be calculated from Eq. (7) and are listed in Table 1.

Absorption line	m_g	m_e	Energy
A	$-1/2$	$-3/2$	$E_0 + E_{\text{iso}} - (-\mu_e + \mu_g)B$
B	$-1/2$	$-1/2$	$E_0 + E_{\text{iso}} - (-\frac{1}{3}\mu_e + \mu_g)B$
C	$-1/2$	$1/2$	$E_0 + E_{\text{iso}} - (\frac{1}{3}\mu_e + \mu_g)B$
D	$1/2$	$-1/2$	$E_0 + E_{\text{iso}} - (-\frac{1}{3}\mu_e - \mu_g)B$
E	$1/2$	$1/2$	$E_0 + E_{\text{iso}} - (\frac{1}{3}\mu_e - \mu_g)B$
F	$1/2$	$3/2$	$E_0 + E_{\text{iso}} - (\mu_e - \mu_g)B$

Table 1: Energies of the hyperfine absorption lines of natural iron, calculated using Eq. (7). E_0 denotes the 14.4 keV line of iron. μ_g and μ_e are the magnetic moments of the ground state and the excited state. The letters correspond to those in Fig. 3.

Note that the energies are symmetric around $E_0 + E_{\text{iso}}$, where E_0 is the 14.4 keV absorption line of the degenerate states in iron.

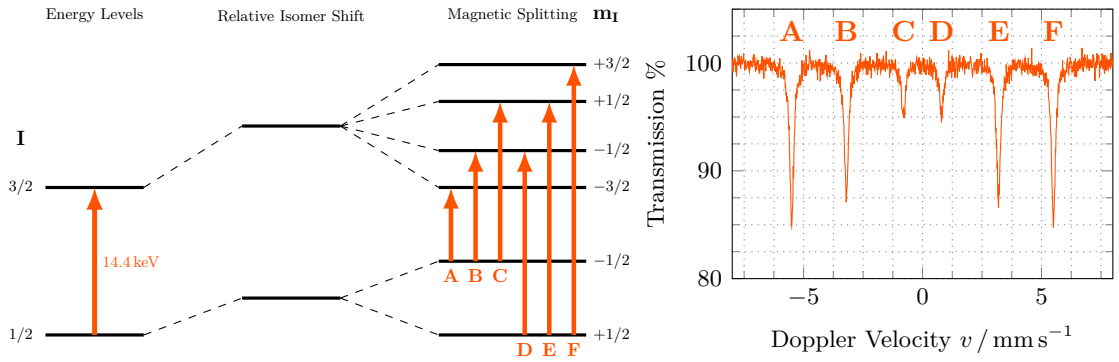


Figure 3: Energy level diagram of natural iron. The transitions expected to be observed in this experiment are marked by arrows and denoted with A to F. The corresponding spectrum is pictured on the right-hand side. Note that the data is generated and not result of a measurement. The figure is adapted from Refs. [10, 12].

1.7. Decay of ^{57}Co

The radioactive source used in this experiment is a ^{57}Co-source. The cobalt decays via electron capture to excited states of ^{57}Fe. The decay to the $I = 5/2$-state of iron is the dominant decay channel with a probability of 99.8 %. The $I = 5/2$-state has a half-life of 8.9 ns and decays to 89 % to the $I = 3/2$-state, emitting a 122.2 keV and to 11 % to the ground state, which produces a 136.6 keV emission line. The $I = 3/2$-state has a half-life of 98 ns and decays to the ground state. The transition of $I = 3/2$-state to the ground state emits a 14.4 keV line, which is the emission line of interest in this experiment. The decay scheme of ^{57}Co with the relevant transitions for this experiment is pictured in Fig. 4.

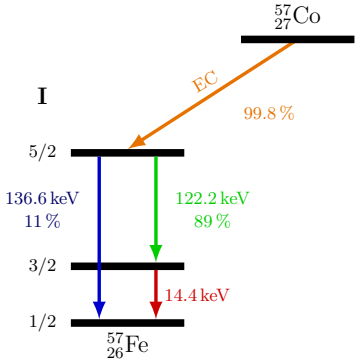


Figure 4: Simplified decay scheme of ^{57}Co. The emission line of interest in this experiment is the 14.4 keV-line (red). The data is taken from Ref. [14].

2. Setup of the Experiment

In this section, the setup of the experiment will be described. The electronic devices used in the setup will also briefly be discussed.

2.1. Setup

The main setup of the experiment is made up of a ^{57}Co-source, a mount on which the absorbers can be placed, and a NaI scintillator. The mount can be moved by an engine which can be set to velocities at a high precision. A picture of this part of the setup is shown in Fig. 5. The engine is connected to a computer and can be controlled by a program. The scintillator is connected to a shaping amplifier. The unipolar output of the amplifier is connected to the SCA as well as a delay unit and then to the input of a linear gate. The positive SCA output is used as an enable signal for the linear gate. The linear gate output then connects to a multi channel analyzer (MCA), which can be configured using the computer. A schematic sketch of the NIM modules can be found in Fig. 6.

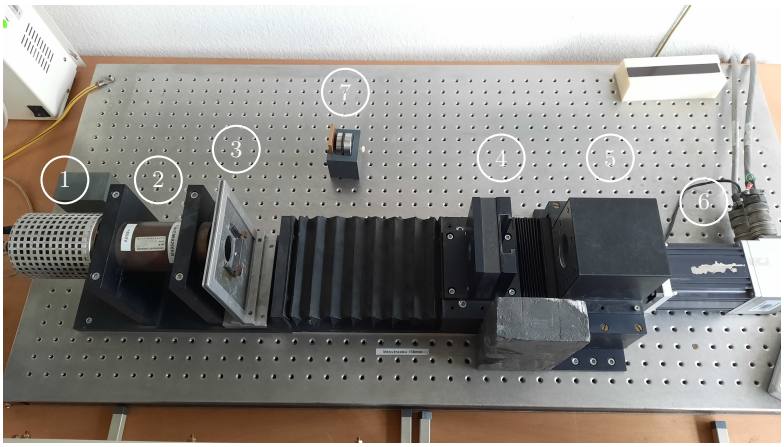


Figure 5: Main setup of the experiment. The components from left to right are the photomultiplier (1), the scintillator (2), a mount for the aluminum shielding for the background measurement (3), the movable mount for the stainless steel and natural iron absorbers (4), the ^{57}Co-source (5) and the engine drive (6). The calibration source (7) is also visible.

2.2. Electronic Devices

In the following, the scintillator and NIM modules used for data processing are briefly described.

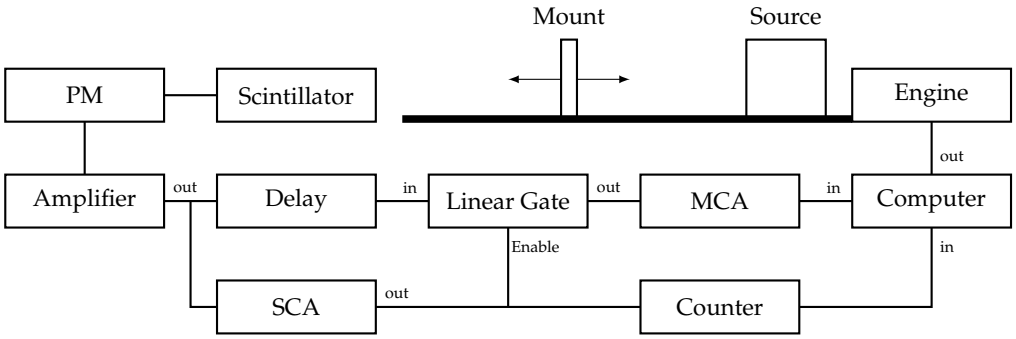


Figure 6: Schematic setup of the experiment.

Scintillator A scintillator is a device used to detect ionizing particles. In general, two different types of scintillators are distinguished: Inorganic scintillators and organic scintillators. In this experiment, an inorganic NaI scintillator is used to detect high-energy photons.

Inorganic scintillators are doped semiconducting crystals. Their behavior can be explained using the band model of semiconductors. The incoming particles interact with the crystal, transferring their energy to the atoms. As a result, electrons are lifted from the valence band to the conduction band. When the electron relaxes back into the valence band, the excess energy is emitted as a photon. In a crystal which is not doped, this photon might have the same energy as the originally incoming particle. However, doping a crystal locally deforms the conduction band and creates additional energy levels an electron can occupy which are situated between the valence band and the conduction band. When the electron relaxes from this energy level to ground state, the energy of the emitted photon is not high enough to excite another atom. These photons can then reach the photomultiplier. Inorganic scintillators in general have a high density which leads to a relatively large amount of detectable photons and leads to a good resolution in the measurement of the energy of the original signal. However, the scintillation decay time is rather large, which means the resolution in time is not very good. Organic scintillators have a better time resolution.

Photomultiplier A photomultiplier is used to convert the signal produced by a scintillator into a measurable current. An optical fiber transfers the photons produced by the scintillator directly onto a photocathode, where the photons are absorbed and electrons are emitted via the photoelectric effect. These primary electrons are then accelerated by an electric field and hit a dynode. Several dynodes are placed after each other and arranged in a way that leads to very little loss of electrons between the dynodes. At the first dynode, the impact of each primary electron leads to the emission of secondary electrons. The number of secondary electrons depends on the applied voltage and often lies between a factor of 3 and 10. All electrons are then accelerated to the next dynode. As each dynode is held at a higher potential than the previous one, the process repeats,

so that at the end a large current which is proportional to the energy of the original signal is produced.

Main Amplifier The main amplifier is used to amplify the signal as well as change the pulse shape. This is done by specifying a shaping time which can be set in a range of 0.5 µs and 10 µs. The amplifier determines the maximum amplitude within the shaping time, which is then used to generate the outgoing signal. Cutting of the signal after the shaping time is done to prevent later signals from overlapping with the long decay time of the earlier signal, which would then lead to the registration of only one signal with a false value for the amplitude. It is important to ensure that the maximum amplitude is reached within the shaping time. However, long shaping times lead to long dead times in which new signals can not be detected, thus limiting the detectable frequency of signals. The other purpose of the main amplifier is to amplify the signal. This is done by adjusting the gain and the coarse gain settings.

There are two outputs of the main amplifier: a unipolar and a bipolar signal. As the energy of the original signal is proportional to the amplitude, the unipolar signal should be chosen if determining the energy is the main goal. However, if the main interest lies in the timing of the signal, it is advantageous to use the bipolar signal instead because its zero-crossing can easily be determined.

For this experiment, the unipolar output was chosen.

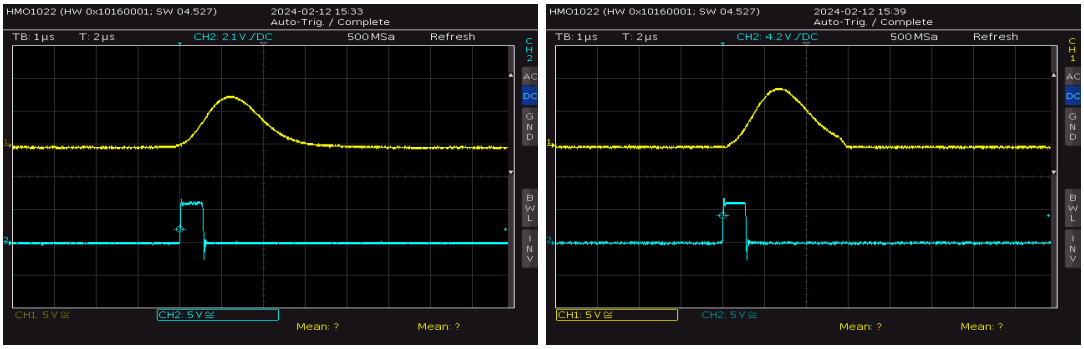
Single Channel Analyzer A single channel analyzer (SCA) sorts incoming signals by amplitude, which is proportional to the energy of the original signal. It is possible to select an energy range for the incoming signal. If a signal falls into this range, a logical signal is passed on by the SCA. It is also possible to specify a delay with which the logical signal is generated.

Linear Gate A linear gate is an electronic module which outputs an incoming signal only if it registers an enable signal. It can be used to analyze signals within a specific energy range. To do so, it can be connected to the main amplifier and use the SCA output as an enable signal. The linear gate will then pass on the signal only if it is within the energy range specified at the SCA and block it otherwise.

Multi Channel Analyzer A multi channel analyzer sorts incoming signals by amplitude into channels. The channels then contain the number of signals with the corresponding amplitude. The multi channel analyzer is connected to a computer. Suitable software can be used to evaluate and illustrate the spectrum and write the count rates to a file. After an energy calibration it is possible to assign the channel numbers to the corresponding energies.

3. Procedure

Calibration In order to perform an energy calibration of the MCA, a variable x-ray source was used. The source consists of a primary ^{241}Am source and six x-ray fluorescent targets on a stainless steel wheel mount. The fluorescent targets used in the calibration were Rubidium (Rb), Molybdenum (Mo), Silver (Ag), Barium (Ba) and Terbium (Tb).
First, the SCA window was set to its maximum range. As the count rate of the ^{57}Co Mößbauer-source was very low, the Tb-source was placed in front of the scintillator for the setup of the experiment. By checking the MCA output, the amplification at the shaping amplifier was set to the maximum at which all calibration peaks were still visible. At the same time, the oscilloscope was used to ensure that the signal was not saturated. The final settings were a coarse gain of 100 and a gain of 14.8. The delay at both SCA and delay unit were adjusted until the signal was centered around the middle of the window opened by the linear gate, such that the signal was cut off symmetrically. The final settings were a delay of 8.8 at the SCA and 3 µs at the delay unit. After the linear gate, both sides of the peak were slightly cut off. However, it was not possible to adjust the with of the linear gate. Instead, the shaping time at the shaping amplifier was lowered to 0.5 µs, at which the amount of the signal which was cut off was minimized. The final signal used as input for the MCA can be seen in Fig. 7b.



(a) Without SCA enable (b) With SCA enable

Figure 7: Final signals (yellow) observed on the oscilloscope, right after the linear gate (see Fig. 6) without and with the enable signal of the SCA. It can be seen that the signal is centered in the SCA window and that applying the window cuts off both sides of the signal symmetrically. The signals in b) were directly used for further analysis as input to the MCA.

Then, for each of the calibration sources, the spectrum was recorded at the MCA. The measurements were stopped after $\approx$ 800 s which corresponds to a measurement time where the smallest emission peak (of Rubidium) reached high counts larger than 1500 counts to be able to perform sensible Gaussian fits. The fits used for the energy calibration (see Section 4.1) can be found in Fig. 13.

In the next step, the windows of the SCA were set in order to isolate a neighborhood of 14.4 keV, at which the iron peak is expected. The energy calibration resulted in a corresponding channel of 2830 ± 160. As the Co-source had very low count rates and the resolution of the scintillator leads to a broadening of the observed peaks, a large window was chosen to ensure that the majority of the 14.4 keV peak was recorded. After the window was set, its boundaries were confirmed using a calibration source with a high count rate at the relevant energies. Unfortunately, if the lower limit of the SCA window was too low, the MCA recorded a lot of noise in the lowest bins. In the end, the lower limit was set as low as possible while not recording noise, and the upper limit was set such that the window limits were symmetric around the 2830 channel at which the 14.4 keV peak was expected. The final settings at the SCA were 1.6 for the lower level and 2.0 for the upper level, resulting in a window of $\approx \pm 1000$ channels around the channel of the iron decay energy.

Background Measurements In order to estimate the attenuation caused by the acrylic glass, a measurement over 600 s was performed both with and without acrylic glass in the mount. The time was chosen such that the relative error on the measurement was at about 2 %. The thickness of the acrylic glass was measured at (2.0 ± 0.5) mm and assumed to be equal to the acrylic glass covering the natural iron and the stainless steel.

The Compton background can be determined by performing measurements with aluminum sheets of varying thickness placed in front of the scintillator. First, a measurement without aluminum over 200 s was performed once with stainless steel and once without an absorber. This was done to estimate the absorption caused by the stainless steel, as the higher energy radiation is expected not to be attenuated by the Plexiglas shielding around the absorber. The results are 401 counts with stainless steel and 758 counts without stainless steel. Therefore, the absorption due to stainless steel absorber can not be neglected. In fact, subtracting the Compton background without considering stainless steel would have led to negative count rates. Thus, the Compton background measurements were performed with stainless steel placed in the movable mount. Stainless steel was chosen over the iron absorber as the background corrections are especially important for the stainless steel analysis to correctly determine the Debye-Waller factor of the source. All measurements were performed over 600 s at a mount speed of $1\,\mathrm{mm\,s^{-1}}$.

Note that it does not matter at which speed the measurements are performed: The Compton background stems from emissions at higher energies which do not cause excitation in the stainless steel and thus are not affected by the velocity. The count rate in terms of absorber thickness is expected to follow a probability density function given by the sum of two exponential decays. While the presence of the steel does lead to lower overall count rates, it only affects the part of the sum which does not correspond to the Compton background. Therefore, by fitting a sum of exponential decays to the data and extracting the coefficients, the Compton part will still be estimated correctly. The analysis of the Compton background is detailed in Section 4.2.

Stainless Steel and Natural Iron The spectra of stainless steel and iron were recorded by measuring count rates at different velocities. As the motor or the software controlling the motor had a habit of randomly stopping, every velocity was measured for 120 s. The software was set to repeat these runs several times.

Unfortunately, it was hard to see a signal as the count rate of the Co-source was very low. In order to maximize the count rates in the relevant region of the spectrum, stainless steel was only measured over a range of $0.05\,\mathrm{mm\,s^{-1}}$ to $1.50\,\mathrm{mm\,s^{-1}}$ in steps of $0.05\,\mathrm{mm\,s^{-1}}$.

For natural iron, the spectrum has to be measured over a wider energy range, so it was not possible to restrict the range of velocities. Instead, a larger step size of $0.2\,\mathrm{mm\,s^{-1}}$ was chosen. The measurement was performed over a range of $0.02\,\mathrm{mm\,s^{-1}}$ to $8.00\,\mathrm{mm\,s^{-1}}$.

Velocity-dependent Background On the last day, the data obtained from the measurement of iron clearly showed an unexpected wavy form. To investigate whether this form is of physical origin or a velocity-dependent background effect, the acrylic glass was placed on the mount and measured over a range of $1\,\mathrm{mm\,s^{-1}}$ to $8\,\mathrm{mm\,s^{-1}}$ in steps of $1.75\,\mathrm{mm\,s^{-1}}$ over a time of 100 s at each step.

4. Data Analysis

The data analysis of the whole experiment was performed in python. If not mentioned otherwise, the fits were conducted performing a weighted least squares minimization using `scipy.optimize.curve_fit` which takes y-uncertainties of the data points into account. The python code that was used is shown in Appendix B.

4.1. Energy Calibration

After the experiment was set up as described in Section 3, an energy calibration was performed to find a suitable discriminator window around the expected iron decay energy of 14.4 keV.

The emission spectra of the five different targets were plotted in absolute counts against channels and Gaussian fits of form

$$G(x) = A \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) + C \tag{8}$$

were performed to estimate the channels of the K_α decay peaks (see Fig. 13). Although the emitted line of each decay is described by a Lorentzian function in theory, the fits were performed using a Gaussian model. This model has been used because the resolution of the recorded spectrum is limited by the resolution of the scintillator. Each line visible in the spectrum is a convolution of the Lorentzian peak of the decay itself and the Gaussian-shaped signal of the scintillator. As the natural linewidth of the decay

is expected much smaller than the resolution of the scintillator, the Lorentzian part of the convolution can be neglected in this case.
Performing the fits yielded the channels of the K_α decays, visible in Fig. 13 and Table 4. Combining these results with the known K_α decay energies of the five targets taken from Ref. [14], an energy calibration could be performed which converts channels to energy via a linear relation. When compared to the width of the peaks, one can see that the uncertainties of the peak positions resulting from the Gaussian fit listed in Table 4 overestimate the accuracy of the fits. Consequently, they were not taken into account for the linear fit used for the energy calibration. Instead, the uncertainties of the fit result from the deviations of the data points from the linear model.

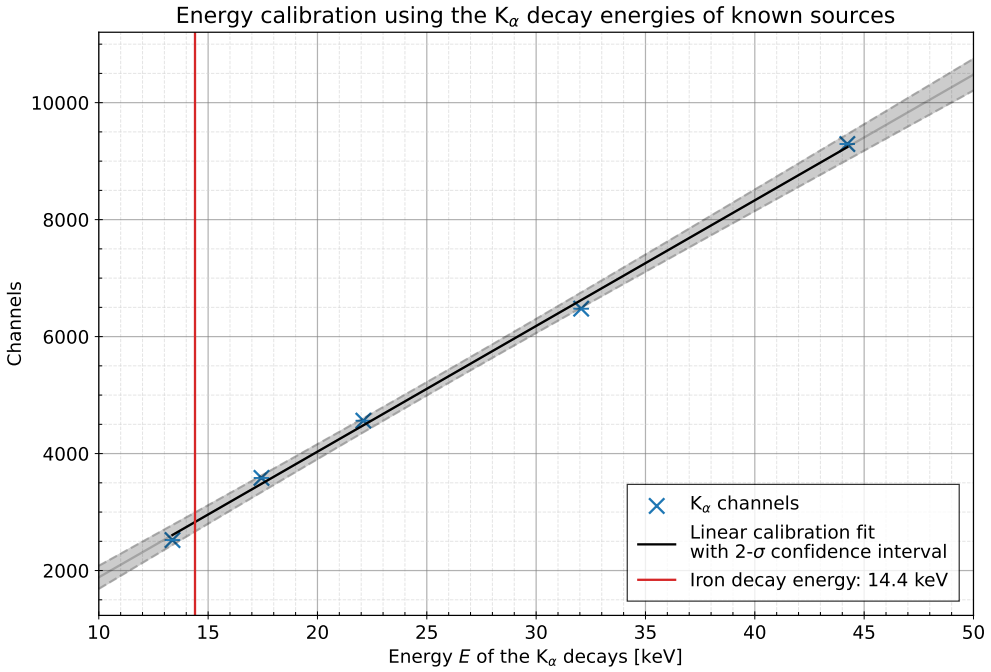


Figure 8: Energy calibration of the MCA with the given setup. The data points show the determined channels of the K_α decays against the known energies taken from [14]. The uncertainties on the channels can be seen in Table 4 and are to small to be seen in the plot. A linear fit was performed to obtain the linear relation between channels and energy. The results can be seen in Eq. (9).

The linear fit yields the relation

$$\text{Channel} = m \cdot E + c \tag{9}$$

with

$$m = (215 \pm 5)\,\text{keV}^{-1} \tag{9a}$$
$$c = -260 \pm 140\,. \tag{9b}$$

This allows to calculate the channel $\text{Channel}_{\text{iron}} = 2830 \pm 160$ at which the 14.4 keV peak of the iron decay is expected. The uncertainties on the channel resulting from the energy calibration are much smaller than the average width of a peak in the MCA spectrum as can be seen for example in Fig. 13.

Additionally, the Co-source was very weak (it has surpassed more than 6 half-lives). This is why it was decided to choose a much wider window at the discriminator around the 14.4 keV channel to ensure that the majority of the signal is recorded by the counter. In the end, a window of 2800 ± 1000 channels was chosen to perform the measurements with. The size of the window in channels was verified using the Rb-source. The MCA spectrum of Rubidium after applying the discriminator window can be seen in Fig. 14.

4.2. Estimation of the Relevant Background Effects

Calculation of the Acrylic Glass Attenuation Both absorber sheets are enveloped by a thin layer of acrylic glass. To quantify the attenuation of the signal due to this Plexiglas window, two measurements with and without a Plexiglas sheet (of same thickness as the acrylic glass that envelopes the absorber sheets) between source and scintillator were performed. Within the chosen measurement time of $t = 600\,\text{s}$, $N_{\text{P}} = 1963\,\text{counts}$ were measured with the Plexiglas absorber and $N_0 = 2336\,\text{counts}$ without the Plexiglas. As the data is acquired in absolute counts, the count rate (in counts per second [cps]) can be calculated by

$$\dot{N} = \frac{N}{t} \tag{10}$$

with the corresponding Poisson error

$$s_{\dot{N}} = \frac{\sqrt{N}}{t}\,. \tag{11}$$

As expected, the relative uncertainties of the measurements lie around 2 %, which would correspond to an absolute number of 2500 counts

$$\frac{\sqrt{N}}{N} \leq 2\,\% \implies N \geq 2500\,\text{counts}\,.$$

From these values, the measured damping factor R_{meas} can be calculated to

$$R_{\text{meas}} = \frac{\dot{N}_{\text{P}}}{\dot{N}_0} = 0.84 \pm 0.03\,.$$

The uncertainty was calculated from the Poisson uncertainties by Gaussian error propagation.

As the attenuation of gamma-radiation in matter follows an exponentially decreasing law (as described in Section 1.1), the theoretically expected damping factor can be calculated from the thickness of the Plexiglas shielding and the attenuation coefficient for a given photon energy

$$\dot{N}(d) = \exp\left(-\frac{\mu}{\rho}\rho d\right) \cdot \dot{N}_0 = R_{\text{theo}}(d) \cdot \dot{N}_0\,. \tag{12}$$

With the measured Plexiglas thickness of $d = (0.200 \pm 0.005)\,\mathrm{cm}$, the mass attenuation coefficient $\mu/\rho = 1.101\,\mathrm{cm^2\,g^{-1}}$ for photons of energy $E_\gamma = 15\,\mathrm{keV} \approx E_{\mathrm{iron}}$ [14] and the Plexiglas density of $\rho = 1.19\,\mathrm{g\,cm^{-3}}$ [14], the calculation yields $R_{\mathrm{theo}} = 0.769 \pm 0.005$.

Both results are not compatible within their 1-σ uncertainties. This can be explained by the fact that the mass attenuation coefficient depends on the energy of the incoming radiation. Thus the 14.4 keV-signal is more strongly attenuated than the higher energy radiation of the iron-decay at 122.2 keV and 136.6 keV [10]. Via Compton scattering in the scintillator, these higher energy photons can produce signals in the that are recorded in the selected energy window. Thus the measured count rate is higher than the expected count rate when considering only the 14.4 keV radiation.

In the next step, the contribution of the Compton background is discussed. The measured value for the Plexiglas attenuation R_{meas} is used for the count rate correction in the following analysis because it can better describe the experimental attenuation effects.

Estimation of the Compton Background As described in Section 4.2, the iron source does not only emit the 14.4 keV radiation but also higher energy photons at 122.2 keV and 136.6 keV [10]. The higher energy photons interact with the scintillation material via the Compton effect and produce photons in the selected energy window that do not stem from the 14.4 keV decay peak. This manifests in a systematically higher count rate throughout the measurement which can not be attributed to the relevant 14.4 keV decay and is independent of the velocity at which the target is moving.

To estimate the Compton Background, aluminum shielding of various thickness was placed in front of the scintillator. The absorption of the gamma radiation in matter depends on the energy of the incoming particle and the thickness of the absorber. As it follows an exponentially decreasing law with increasing absorber thickness, the superposition of two exponential decay functions with different decay constants is expected to be visible when plotting the count rate against the thickness of the Al-shielding. This expected distribution consists of one fast decaying part, resulting from the 14.4 keV decay and a slower decaying part, originating from the higher energy radiation. Extrapolating this slower decaying part to an Al-thickness of 0 mm then yields the contribution of the Compton background. A plot of the measured count rate against the thickness of the Al-shielding is visible in Fig. 9 with the corresponding Poisson errors.

As motivated above, a fit of two exponentially decaying functions of the form

$$\dot{N}(d) = A \cdot \exp\left(-\frac{d}{\tau}\right) + B \cdot \exp\left(-\frac{d}{\tilde{\tau}}\right) \tag{13}$$

was performed. The goodness of the fit is rather bad, resulting in a reduced χ^2 value of $\chi^2_\nu = 2.2$, which can be explained by the partially large fluctuations of the data points around the model and the lack of more statistics. The fit yields a Compton background at an absorber thickness of 0 mm of

$$A = \dot{N}_{\mathrm{Compton}}(0) = (1.61 \pm 0.04)\,\mathrm{cps}\,.$$

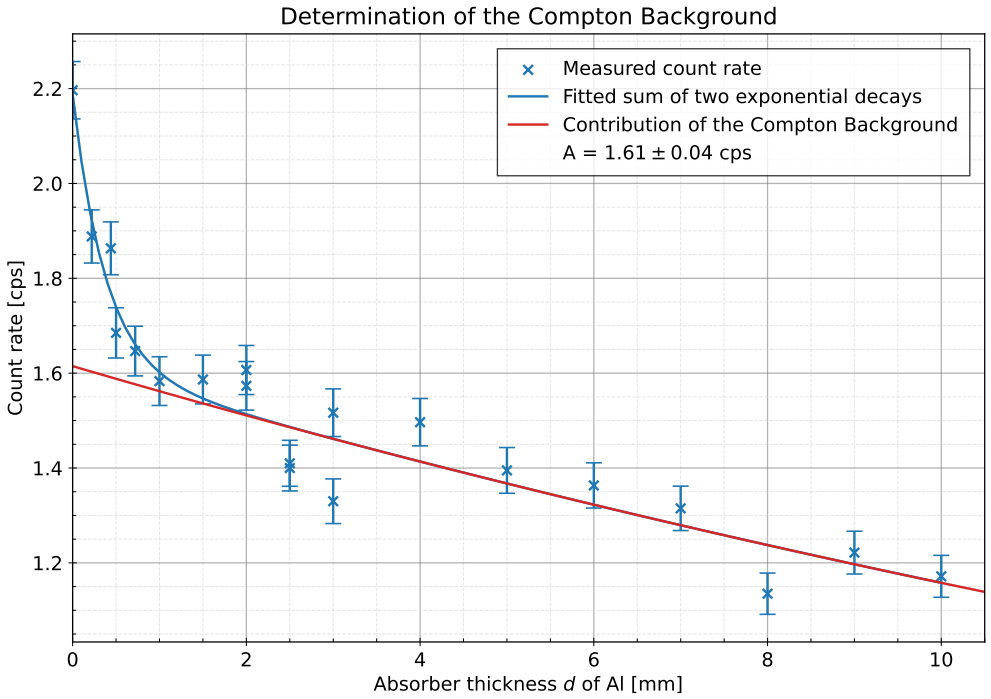


Figure 9: Measured count rate at the scintillator for different thicknesses of aluminum shielding placed in front of it. The values and uncertainties have been calculated using Eqs. (10) and (11). A fit of form Eq. (13) has been performed, resulting in a reduced χ^2 value of $\chi^2_\nu = 2.2$. The exponential contribution of the Compton background is shown in red.

The uncertainties were estimated due to the fluctuations of the data points around the model.

For further analysis, this value will be subtracted from all count rates to get an estimation of the raw 14.4 keV signal without contribution of the Compton background. It does not need to be corrected for the acrylic glass attenuation because the Compton background was measured with the stainless steel absorber, which is already encased in acrylic glass. The final correction which was applied to all count rates for further analysis then reads

$$\dot{N} = \frac{\dot{N}_{\text{meas}} - \dot{N}_{\text{Compton}}}{R_{\text{meas}}}\,, \tag{14}$$

with uncertainties calculated by Gaussian error propagation.

It is of note that in the fit shown in Fig. 9, the contribution of the 14.4 keV peak is already close to zero at a aluminum thickness of approximately 2 mm. This is in agreement with the plot in Fig. 15 which shows a transmission of the 14.4 keV line at a thickness of 2 mm of around 2 %.

4.3. Measurement with the Stainless Steel Absorber

After the relevant background effects were determined, the measurement with the stainless steel absorber was started. For velocities from $-1.5\,\mathrm{mm\,s^{-1}}$ to $1.5\,\mathrm{mm\,s^{-1}}$ in steps of $0.05\,\mathrm{mm\,s^{-1}}$, the counts were recorded using the counter.

As different velocities of the sledge can be translated directly to different excitation energies, peaks in the spectrum are expected to have a negative Lorentzian shape. It can be described by

$$\dot{N}_L(v; A, \mu, \gamma, C) = \frac{A\gamma}{\pi} \frac{1}{(v-\mu)^2 + \gamma^2} + C\,, \tag{15}$$

where the amplitude A is expected negative, as an absorption spectrum is observed. Statistical effects in the measurement such as changes in the room temperature, varying sledge velocities or the difference in solid angles caused by different distances to source and absorber follow a Gaussian distribution. It is described by

$$\dot{N}_G(v; A, \mu, \sigma, C) = \frac{A}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(v-\mu)^2}{2\sigma^2}\right) + C\,. \tag{16}$$

The limited resolution of the scintillator will affect the form of the absorption peaks as well. The finally detected absorption peaks in the spectrum will thus follow a Voigt form, which is a convolution of the Gaussian and the Lorentzian part and reads

$$\dot{N}_V(v; A, \mu, \sigma, \gamma, C) = A \cdot \int_{-\infty}^{\infty} G(v'; \sigma) L(v - \mu - v'; \gamma) \mathrm{d}v' + C\,. \tag{17}$$

The Voigt profile has no exact analytical representation. For the analysis, the scipy-method `scipy.special.voigt_profile` was used. The spectrum of stainless steel after having corrected for the relevant background effects using Eq. (14) are visible in Fig. 10. As motivated above, all three model fits (see Eqs. (15) to (17)) were performed. The fit results are listed in Table 2.

Parameters	Gaussian Fit	Lorentzian Fit	Voigt Fit
A [cps]	-0.08 ± 0.03	-0.14 ± 0.06	-0.14 ± 0.06
μ [$\mathrm{mm\,s^{-1}}$]	0.23 ± 0.06	0.22 ± 0.06	0.22 ± 0.06
σ [$\mathrm{mm\,s^{-1}}$]	0.18 ± 0.06		0 ± 170
γ [$\mathrm{mm\,s^{-1}}$]		0.22 ± 0.10	0.22 ± 0.11
C [cps]	0.591 ± 0.016	0.61 ± 0.02	0.61 ± 0.02

Table 2: Fit results of the stainless steel analysis for the fit functions described in Eqs. (15) to (17). The data points and fits are shown in Fig. 10. The reduced χ^2 value yields $\chi^2_\nu = 0.6$ for all three fits.

The reduced χ^2 value of $\chi^2_\nu = 0.6 < 1$ for all three fits is rather small and indicates very large uncertainties of the data points compared to the fluctuations around the model

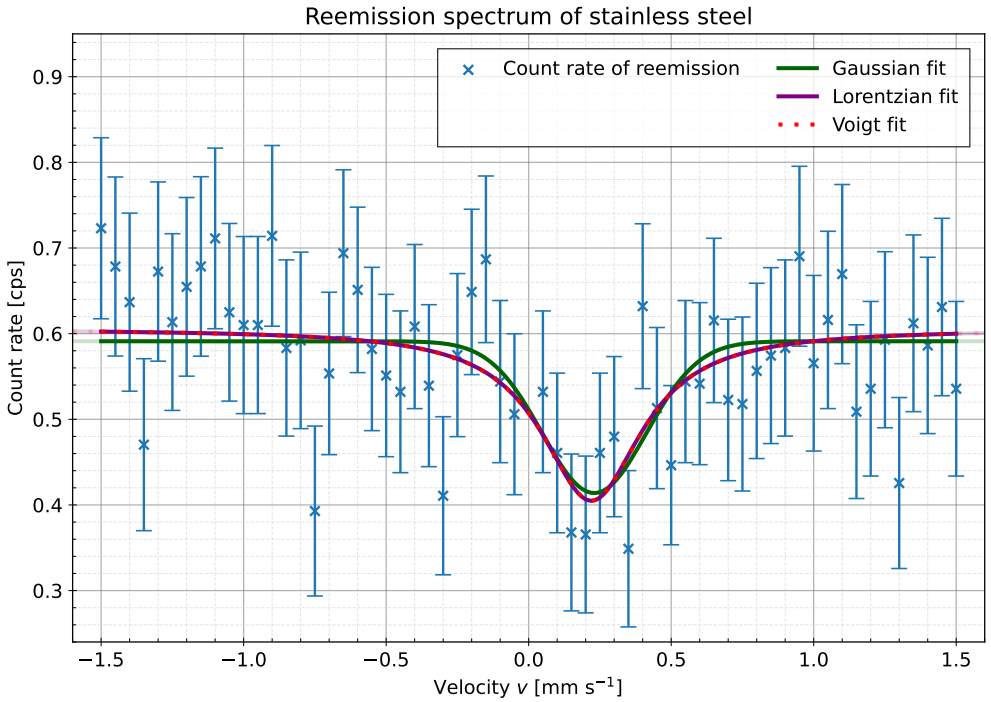


Figure 10: Results of the measurement with the stainless steel absorber during an overall data taking period of 7.5 h, resulting in a net measurement time of approximately 450 s per data point. The count rate correction according to Eq. (14) has already been performed. Fits of Gaussian (Eq. (16)), Lorentzian (Eq. (15)) and Voigt (Eq. (17)) form were performed and are visible as well. As the Lorentzian and the Voigt fit differ only slightly, the difference between both models is not well visible. The fit results can be taken from Table 2. The reduced χ^2 value yields $\chi^2_\nu = 0.6$ for all three fits.

functions. This is consistent with Fig. 10. All in all, it can be said that the data is not sufficient to see differences between the different model functions, as all the parameter values are compatible within their uncertainties.

4.3.1. Isomer Shift

The deviation of the absorption minimum ($\mu \neq 0$) from the zero velocity is caused by the isomer shift and can be used to draw conclusions about the chemical composition of the absorber (see Section 1). The energy shift corresponding to this velocity can be calculated using the Doppler effect (see Section 1) and reads

$$\Delta E = \frac{\mu}{c} E_{\text{iron}} \,, \tag{18}$$

where μ is the sledge velocity at the absorption minimum and $E_{\text{iron}} = (14.412\,95 \pm 0.000\,31)$ keV [1] is the energy of the 14.4 keV iron emission line. As the fit uncertainties

are rather large, all three profiles yield the same result. The isomer shift of the stainless steel absorber can be calculated to

$$E_{\text{iso}} = (11 \pm 3)\,\text{neV}\,,$$

with the uncertainties calculated by Gaussian error propagation. Again this value has a large uncertainty, resulting in a relative uncertainty of 27 %. As the uncertainties only depend on the fit uncertainties, which are much larger than the uncertainty of the iron decay energy, longer measurement times and a stronger source could significantly reduce the relative error. As no literature value for this specific absorber is given, no comparison can be made.

4.3.2. Calculation of the Effective Absorber Thickness

To estimate the factor of recoilless emissions from the source, described by the Debye-Waller factor f_{S} of the source, the dimensionless quantity T_A, the effective absorber thickness, needs to be determined. It can be calculated using

$$T_{\text{A}} = f_{\text{A}} n_{\text{A}} \beta \sigma_0 d_{\text{A}}\,, \tag{19}$$

where the used quantities are the following [14]:

$$\begin{aligned} f_{\text{A}}(20\,^\circ\text{C}) = 0.8 &: \text{Debye-Waller factor of the absorber} \\ n_{\text{A}} &: \text{number of iron atoms per m}^3 \\ \beta = 0.022 &: \text{fraction of } {}^{57}\text{Fe in the isotope mixture} \\ \sigma_0 &: \text{resonant absorption cross section} \\ d_{\text{A}} = 25\,\mu\text{m} &: \text{absorber thickness}\,. \end{aligned}$$

All following uncertainties were calculated using Gaussian error propagation.

The number n_{A} of iron atoms per m^3 can be determined from the Avogadro constant N_{A}, the iron content in the absorber $p = 0.70 \pm 0.05$ [14], the density of iron $\rho_{\text{iron}} = 7.874\,\text{g cm}^{-3}$ [8] and the molar mass of iron $m_{\text{iron}} = (55.845 \pm 0.002)\,\text{g mol}^{-1}$ [13] with

$$n_{\text{A}} = p \frac{\rho_{\text{iron}} N_{\text{A}}}{m_{\text{iron}}}\,. \tag{20}$$

The calculation yields

$$n_{\text{A}} = (5.9 \pm 0.4) \cdot 10^{28}\,\text{m}^{-3}\,.$$

Using $E_{\text{iron}} = hc/\lambda_{\text{iron}}$ the resonant absorption cross section σ_0 can be calculated with

$$\sigma_0 = \frac{1}{2\pi}\left(\frac{hc}{E_{\text{iron}}}\right)^2 \left(\frac{2I_{\text{e}}+1}{2I_{\text{g}}+1}\right)\frac{1}{1+\alpha}\,, \tag{21}$$

as given in [4]. In the above equation, $I_{\text{e}} = 3/2$ and $I_{\text{g}} = 1/2$ are the contributing spin states of the excited state and the ground state, respectively, $E_{\text{iron}} = (14.412\,95 \pm$

0.000 31) keV [1] is the excitation energy and $\alpha = 8.55 \pm 0.12$ [1] the conversion coefficient. Calculating the cross section yields

$$\sigma_0 = (2.47 \pm 0.03) \cdot 10^{-22}\,\mathrm{m}^2\,.$$

Combining these results and the other quantities as described in Eq. (19), the effective absorber thickness T_A can be calculated to

$$T_\mathrm{A} = 6.5 \pm 0.5\,,$$

which has a relative uncertainty of 7.7 %. The largest contribution to this uncertainty stems from the iron content p in the absorber which itself has a large relative uncertainty.

4.3.3. Debye-Waller Factor of the Source

With the effective absorber thickness T_A and the obtained fit results, the Debye-Waller factor f_S of the source can be calculated using Eq. (5) as explained in Section 1.

The factor $(\dot{N}(\infty) - \dot{N}(0))/\dot{N}(\infty)$ describes the depth of the absorption peak, normalized by the expected count rates at infinite velocities. So $\dot{N}(0)$ is not the count rate at zero sledge velocity but the one in the absorption minimum which is shifted by the isomer shift μ. The expected count rate at infinite velocities is then given by the vertical offset C of the fit. With these results, Eq. (5) can be written as

$$f_\mathrm{S} = \frac{C - \dot{N}(\mu)}{C} \frac{1}{1 - \exp\left(-\frac{1}{2}T_\mathrm{A}\right) \mathrm{J}_0\left(\frac{1}{2}iT_\mathrm{A}\right)}\,. \tag{22}$$

The uncertainties can be calculated using Gaussian error propagation

$$s_{f_\mathrm{S}} = \sqrt{\left(\frac{\partial f_\mathrm{S}}{\partial C} s_C\right)^2 + \left(\frac{\partial f_\mathrm{S}}{\partial \dot{N}(\mu)} s_{\dot{N}(\mu)}\right)^2 + \left(\frac{\partial f_\mathrm{S}}{\partial T_\mathrm{A}} s_{T_\mathrm{A}}\right)^2} \tag{23}$$

with

$$\frac{\partial f_\mathrm{S}}{\partial C} = -\frac{\dot{N}(\mu)}{C^2} \frac{1}{1 - \exp\left(-\frac{1}{2}T_\mathrm{A}\right) \mathrm{J}_0\left(\frac{1}{2}iT_\mathrm{A}\right)} \tag{23a}$$

$$\frac{\partial f_\mathrm{S}}{\partial \dot{N}(\mu)} = -\frac{1}{C} \frac{1}{1 - \exp\left(-\frac{1}{2}T_\mathrm{A}\right) \mathrm{J}_0\left(\frac{1}{2}iT_\mathrm{A}\right)} \tag{23b}$$

$$\frac{\partial f_\mathrm{S}}{\partial T_\mathrm{A}} = -\frac{C - \dot{N}(\mu)}{2C} e^{-\frac{1}{2}T_\mathrm{A}} \frac{\mathrm{J}_0\left(\frac{1}{2}iT_\mathrm{A}\right) + i\mathrm{J}_1\left(\frac{1}{2}iT_\mathrm{A}\right)}{\left(1 - \exp\left(-\frac{1}{2}T_\mathrm{A}\right) \mathrm{J}_0\left(\frac{1}{2}iT_\mathrm{A}\right)\right)^2}\,. \tag{23c}$$

The uncertainty s_C is obtained directly from the fits. The uncertainties of the fit functions evaluated at their minima $s_{\dot{N}(\mu)}$ can be estimated using the uncertainty of the

minimum position s_μ by determining the count rate difference $\dot{N}(\mu + s_\mu) - \dot{N}(\mu)$. The Bessel-functions J_0 and J_1 have been calculated in python, using the `scipy.special.jv` method. Performing the calculations yields a result of

$$\begin{aligned} f_{\mathrm{S,\,G}} &= 0.39 \pm 0.03\,, \\ f_{\mathrm{S,\,L}} &= 0.43 \pm 0.04\,, \\ f_{\mathrm{S,\,V}} &= 0.43 \pm 0.04 \end{aligned}$$

for the Debye-Waller factors, determined with the three different fit functions. As expected from previous results, all three values are compatible within their 1-σ uncertainties which again shows that the different analysis methods do not have a major impact on the results with the available data. More data would have to be taken to differentiate between the analysis methods. As no literature value for comparison is available, only the qualitative assessment can be made that a percentage of around 40 % of recoilless emissions from the source seems reasonable.

4.3.4. Lifetime of the 14.4 keV state in ^{57}Fe

Normally, the known relation $\tau = \hbar/\Gamma_{\mathrm{nat.}}$ holds between the natural linewidth $\Gamma_{\mathrm{nat.}}$ and the lifetime of a state τ. Due to the absorption and reemission of the radiation in the absorber, the measured linewidth $\Gamma_{\mathrm{meas.}}$ appears larger than the natural linewidth is. This is due to an overlap of the absorption and emission spectrum. To correct for this relative line broadening, a correction factor W needs to be considered, as derived in [2, 4, 6]. The results of a numerical approach are shown in Fig. 11.

It can be seen that for effective source- and absorber thicknesses $T_{\mathrm{A}} = T_{\mathrm{S}} = 0$, the relative line broadening is $\Gamma_{\mathrm{meas.}}/\Gamma_{\mathrm{nat.}} \approx 2$. In other words, the observed absorption line has twice the natural linewidth.

To correct for this effect, the effective source thickness T_{S} needs to be calculated. As in Eq. (19), the relation

$$T_{\mathrm{S}} = f_{\mathrm{S}} n_{\mathrm{S}} \beta \sigma_0 d_{\mathrm{S}} \tag{24}$$

holds with f_{S} the Debye-Waller factor of the source as calculated in Section 4.3.3, $n_{\mathrm{S}} = n_{\mathrm{A}}$ as in Eq. (20), $\beta = 1$, σ_0 from Eq. (21) and $d_{\mathrm{S}} = \mathcal{O}(100\,\text{Å})$ [14].
Performing the calculations with the results of the different fit functions yields

$$\begin{aligned} T_{\mathrm{S,\,G}} &= 0.057 \pm 0.004\,, \\ T_{\mathrm{S,\,L}} &= 0.063 \pm 0.005\,, \\ T_{\mathrm{S,\,V}} &= 0.063 \pm 0.005 \end{aligned}$$

and therefore $T_{\mathrm{S}} \approx 0$. In this case, Visscher's formula gives an analytic approximation of the relative line broadening [2, 4]. It describes the relative line broadening using a second order polynomial for $4 \leq T_{\mathrm{A}} \leq 10$

$$W = \Gamma_{\mathrm{meas.}}/\Gamma_{\mathrm{nat.}} \approx 2\left(1.01 + 0.145\,T_{\mathrm{A}} - 0.0025\,T_{\mathrm{A}}^2\right)\,. \tag{25}$$

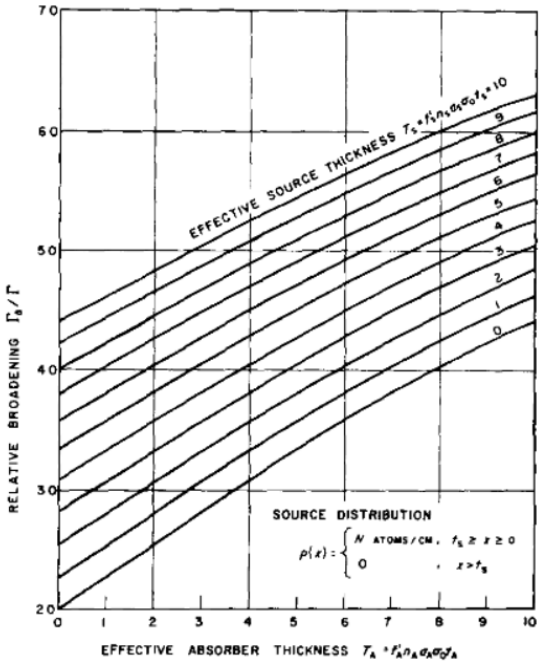


Figure 11: Relative line broadening $\Gamma_{\text{meas.}}/\Gamma_{\text{nat.}}$ as a function of the effective absorber thickness T_{A} for different effective source thicknesses T_{S}, taken from [6].

As $T_{\text{A}} = 6.5 \pm 0.5$ (see Section 4.3.2), the relation holds.
Using Gaussian error propagation, the correction factor can be calculated to

$$W = 3.68 \pm 0.11\,.$$

Applying this correction, the natural linewidth of the absorption peak can be estimated using the width of the three different model functions. It holds that

$$\begin{aligned}
\Gamma_{\text{meas., G}} &= 2\sqrt{2\ln(2)}\sigma = (0.43 \pm 0.14)\,\text{mm}\,\text{s}^{-1}\,,\\
\Gamma_{\text{meas., L}} &= 2\gamma_{\text{L}} = (0.4 \pm 0.2)\,\text{mm}\,\text{s}^{-1}\,,\\
\Gamma_{\text{meas., V}} &= 2\gamma_{\text{V}} = (0.4 \pm 0.2)\,\text{mm}\,\text{s}^{-1}\,,
\end{aligned}$$

where the width of the Voigt function is only determined by the Lorentzian part to eliminate the statistical broadening effects described by the Gaussian part. The natural linewidth can then be calculated using the correction described in Eq. (25). Using Eq. (18) to convert the unit to energy in neV results in values for the natural linewidth of

$$\begin{aligned}
\Gamma_{\text{nat., G}} &= (5.6 \pm 1.9)\,\text{neV}\,,\\
\Gamma_{\text{nat., L}} &= (6 \pm 3)\,\text{neV}\,,\\
\Gamma_{\text{nat., V}} &= (6 \pm 3)\,\text{neV}\,.
\end{aligned}$$

The lifetime of ^{57}Fe can then be calculated using $\tau = \hbar/\Gamma_{\text{nat.}}$,

$$\begin{aligned}\tau_{\text{G}} &= (120 \pm 40)\,\text{ns}\,,\\ \tau_{\text{L}} &= (120 \pm 60)\,\text{ns}\,,\\ \tau_{\text{V}} &= (120 \pm 60)\,\text{ns}\,.\end{aligned}$$

Comparing these results to the literature value of $\tau(^{57}\text{Fe}) = 141\,\text{ns}$ [3] shows a good accordance within the 1-σ uncertainties of the values. However, it has to be said that the uncertainties are very large, resulting in relative uncertainties of up to 50 %. As a consequence the good accordance with the literature value is not surprising. As the major contribution to these uncertainties originates directly from the fits (see Fig. 10, compare with uncertainties in Table 2), the uncertainties for the lifetime of ^{57}Fe could have significantly been reduced by taking more data.

4.4. Measurement with the Natural Iron Absorber

As with the stainless steel absorber, an analysis with an absorber made out of natural iron was performed to investigate its hyperfine splitting. Both absorbers are expected to be manufactured in the same way, so the enveloping layer of Plexiglas is expected to have the same thickness. As a consequence, the same background considerations can be made, resulting in the background correction Eq. (14) as used before.

As argued in Section 1.6, six distinct absorption peaks are expected, caused by the excited transitions shown in Fig. 3. The recorded spectrum is expected to show symmetric absorption peaks with respect to a common energy offset caused by the isomer shift. The energies of the expected absorption spectrum are listed in Table 1. Translated to the recorded spectrum in count rates against velocities, this means that six dips are expected, symmetric to a common horizontal offset.

The spectrum was recorded in an overall data taking period of 18.2 h, resulting in a net measurement time of $\approx 820\,\text{s}$ per data point. It is shown in Fig. 12 after having applied the relevant background corrections according to Eq. (14). Unfortunately due to the weak source and the lack of more statistics, only the two innermost of the expected six absorption peaks are visible. For velocities of $|v| \geq 2\,\text{mm}\,\text{s}^{-1}$, huge fluctuations from the baseline occur. Unfortunately it was not possible to find an explanation for those.

To investigate the hyperfine splitting and the isomer shift for the iron target, again fits of Gaussian (Eq. (16)), Lorentzian (Eq. (15)) and Voigt (Eq. (17)) form were performed to describe the absorption peaks. As motivated in Section 1.6, the symmetry-condition was added as a constraint for the fit function, reducing the number of free parameters and increasing the accordance with the theoretically expected behavior. The model functions fitted to the selected data points were of the form

$$\begin{aligned}\text{Model}(v) = A \cdot F(v - \mu - \mu_\alpha; \ldots)\\ + A \cdot F(v - \mu + \mu_\alpha; \ldots) + C,\end{aligned} \tag{26}$$

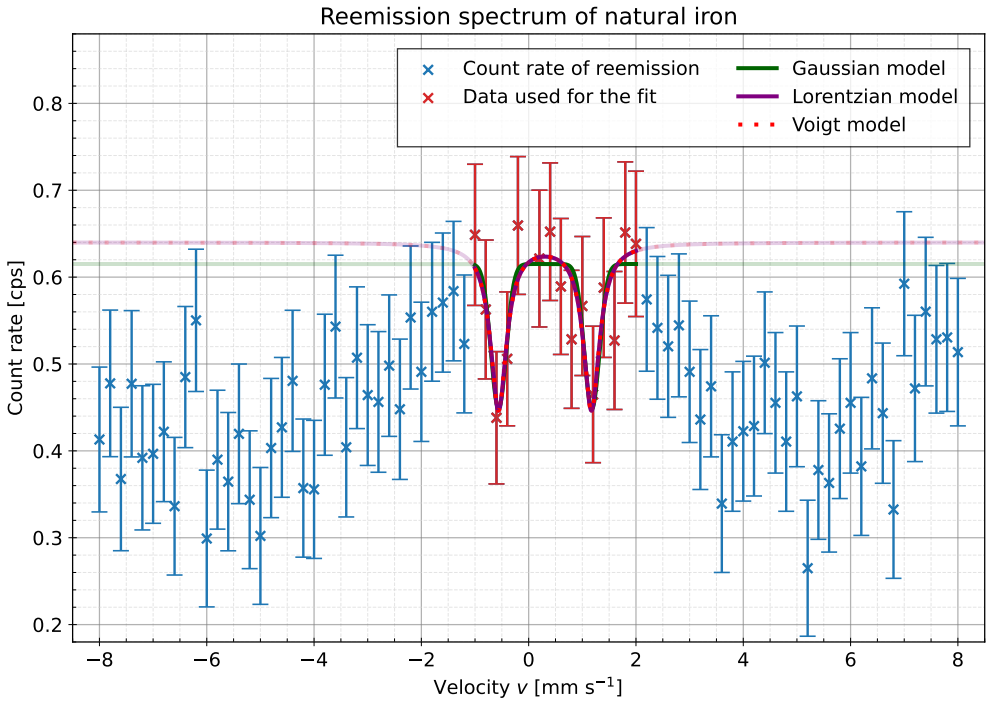


Figure 12: Results of the measurement with the iron absorber during an overall data taking period of 18.2 h, resulting in a net measurement time of ≈ 820 s per data point. The count rate correction according to Eq. (14) has already been performed. Fits of Gaussian (Eq. (16)), Lorentzian (Eq. (15)) and Voigt (Eq. (17)) form were performed, taking the symmetry conditions described in Section 1.6 and Eq. (26) into account. The fit results can be taken from Table 3. The reduced χ^2 value yields $\chi^2_\nu = 0.4$ for all three fits.

where F denotes Gaussian, Lorentzian or Voigt functions of the form described in Eqs. (15) to (17). The argument $v - \mu \pm \mu_\alpha$ enforces the symmetry constraints of a common offset corresponding to the isomer shift μ and the velocity μ_α at which the first (and only visible) hyperfine transition occurs. The fit results are listed in Table 3.

4.4.1. Isomer Shift

As explained in Section 4.3.1, the isomer shift can be computed via Eq. (18) from the corresponding sledge velocity. With a fit function of a form described in Eq. (26), the velocity of the isomer shift directly corresponds to the parameter μ in the performed fits, the values of which are listed in Table 3. Using these results and Eq. (18), the isomer

Parameters	Gaussian Fit	Lorentzian Fit	Voigt Fit
A [cps]	-0.059 ± 0.019	-0.11 ± 0.05	-0.11 ± 0.12
μ [mm s^{-1}]	0.31 ± 0.04	0.30 ± 0.04	0.30 ± 0.04
μ_α [mm s^{-1}]	0.87 ± 0.04	0.87 ± 0.05	0.87 ± 0.06
σ [mm s^{-1}]	0.14 ± 0.04		0 ± 20
γ [mm s^{-1}]		0.18 ± 0.08	0.2 ± 0.4
C [cps]	0.615 ± 0.018	0.64 ± 0.03	0.64 ± 0.06

Table 3: Fit results of the iron analysis for the fit functions Eqs. (15) to (17), using the symmetry constraints as presented in Eq. (26). The data points and fits are visible in Fig. 12. The reduced χ^2 value yields $\chi^2_\nu = 0.4$ for all three fits.

shift of the iron target can be calculated to

$$E_{\text{iso, G}} = (14.8 \pm 1.8)\,\text{neV}\,,$$
$$E_{\text{iso, L}} = (15 \pm 2)\,\text{neV}\,,$$
$$E_{\text{iso, V}} = (15 \pm 2)\,\text{neV}\,.$$

4.4.2. Calculation of the Magnetic Field Strength at the Nucleus

If it had been possible to see three peaks to each side, the fit of the iron spectrum would have resulted the energy of the isomer shift E_{iso} and the absolute distances of the dips to E_{iso}. Due to the fit constraints, there would have been three different energy distances to E_{iso}, which in the following will be referred to as ΔE_α, ΔE_β and ΔE_γ, where $\Delta E_\alpha \leq \Delta E_\beta \leq \Delta E_\gamma$. The relation of these values to the magnetic moments of the excited state μ_e and the ground state μ_g as well as the magnetic field at the nucleus B can be calculated from Eq. (7). This results in

$$\Delta E_\alpha = (\frac{1}{3}\mu_e + \mu_g)\,B\,, \tag{27a}$$
$$\Delta E_\beta = (-\frac{1}{3}\mu_e + \mu_g)\,B\,, \tag{27b}$$
$$\Delta E_\gamma = (-\mu_e + \mu_g)\,B\,. \tag{27c}$$

These equations can now be used to determine μ_e and B from the fits as well as the literature value for μ_g. Unfortunately, it was only possible to fit the two innermost peaks, which correspond to ΔE_α. As a result, ΔE_β and ΔE_γ are unknown and only Eq. (27a) can be used. As Eq. (27a) has three unknown variables, they can not be determined uniquely. Instead, in the following, a literature value for μ_e as well as μ_g will be used to determine the magnetic field strength B at the nucleus. This is done by rearranging Eq. (27a),

$$B = \frac{\Delta E_\alpha}{(1/3)\mu_e + \mu_g}\,. \tag{28}$$

The uncertainty is estimated by Gaussian error propagation,

$$s_B = \sqrt{\left(\frac{1}{\mu_e/3+\mu_g} s_{\Delta E_\alpha}\right)^2 + \left(\frac{1}{3}\frac{\Delta E_\alpha}{(\mu_e/3+\mu_g)^2} s_{\mu_e}\right)^2 + \left(\frac{\Delta E_\alpha}{(\mu_e/3+\mu_g)^2} s_{\mu_g}\right)^2}. \quad (29)$$

As before, ΔE_α is calculated using Eq. (18) and Gaussian error propagation. The magnetic moments $\mu_g = 0.090\,44 \pm 0.000\,07\,\mu_N$ and $\mu_e = -0.1549 \pm 0.0002\,\mu_N$ are taken from "Table of nuclear magnetic dipole and electric quadrupole moments" [11], and $\mu_N = 3.152\,45 \cdot 10^{-8}\,\mathrm{eV\,T^{-1}}$ [9] is the nuclear magneton. For the three fits, the results for the magnetic field are

$$B_\mathrm{G} = (34.2 \pm 1.5)\,\mathrm{T}\,,$$
$$B_\mathrm{L} = (34.0 \pm 1.8)\,\mathrm{T}\,,$$
$$B_\mathrm{V} = (34 \pm 2)\,\mathrm{T}\,.$$

The literature value fo the magnetic field strength at the nucleus at a temperature of 300 K is given in Ref. [3] as $B_\mathrm{lit} = 33.0\,\mathrm{T}$. The literature value is within one sigma of the magnetic field strength calculated from the measurement for all three fit functions, which indicates good agreement of the data with the literature value.

5. Summary and Discussion

5.1. Results

In the first part of the experiment, the multi channel analyzer was calibrated using five sources with known spectra. The channel to energy conversion was determined to

$$\text{Channel} = (215 \pm 5)\,\text{keV}^{-1} \cdot E - (260 \pm 140)$$

and was used to set the discriminator window.

Next, the effects contributing to the background were investigated. The damping factor of acrylic glass was determined at

$$R_{\text{meas}} = 0.84 \pm 0.03\,,$$

which is not compatible within one sigma with the theoretical dampening factor

$$R_{\text{theo}} = 0.769 \pm 0.005$$

calculated from the thickness of the acrylic glass. The deviation can be explained by the influence of higher-energy emission lines of the Co-source, for which the mass attenuation coefficient used theoretical calculation does not apply.

The measurement of the Compton background was performed with the stainless steel absorber placed in the mount. As the stainless steel absorber is also encased in acrylic glass, the result for the Compton background determined at

$$\dot{N}_{\text{Compton}} = (1.61 \pm 0.04)\,\text{cps}$$

already incorporates the attenuation due to the acrylic glass.

The spectra of both stainless steel and natural iron were recorded by measuring count rates in dependence on the velocity with which the absorbers were moved. The background was subtracted from the spectra, and the absorption dips visible in the spectra were fitted using a Gaussian, a Lorentzian and a Voigt profile.

The isomer shift of stainless steel was determined at

$$E_{\text{iso}} = (11 \pm 3)\,\text{neV}\,.$$

The effective absorber thickness of stainless steel was calculated to

$$T_{\text{A}} = 6.5 \pm 0.5\,,$$

which could then be used to determine the Debye-Waller factor of the source describing the fraction of recoilless emissions from the source. The Debye-Waller factor was calculated from all three fit functions, resulting in

$$\begin{aligned} f_{\text{S, G}} &= 0.39 \pm 0.03\,, \\ f_{\text{S, L}} &= 0.43 \pm 0.04\,, \\ f_{\text{S, V}} &= 0.43 \pm 0.04\,. \end{aligned}$$

All of these results are compatible with each other within their one sigma uncertainties.

Next, the lifetime of the 14.4 keV state of ^{57}Fe was determined, resulting in

$$\begin{aligned}\tau_{\mathrm{G}} &= (120 \pm 40)\,\mathrm{ns}\,,\\ \tau_{\mathrm{L}} &= (120 \pm 60)\,\mathrm{ns}\,,\\ \tau_{\mathrm{V}} &= (120 \pm 60)\,\mathrm{ns}\end{aligned}$$

for the three fit functions. The literature value of 141 ns [3] is well within one sigma of all of the results. However, the relative errors of 33 % and 50 % are rather large.

The isomer shift of natural iron was determined at

$$\begin{aligned}E_{\mathrm{iso,\,G}} &= (14.8 \pm 1.8)\,\mathrm{neV}\,,\\ E_{\mathrm{iso,\,L}} &= (15 \pm 2)\,\mathrm{neV}\,,\\ E_{\mathrm{iso,\,V}} &= (15 \pm 2)\,\mathrm{neV}\end{aligned}$$

for the three fit functions.

Unfortunately, only the two innermost peaks of the six expected peaks were visible in the data. As a consequence, it was not possible to determine both the magnetic moment μ_e of the excited state and the magnetic field strength B at the nucleus from the measurement. Instead, a literature value for μ_e was used to determine B from the data, resulting in

$$\begin{aligned}B_{\mathrm{G}} &= (34.2 \pm 1.5)\,\mathrm{T}\,,\\ B_{\mathrm{L}} &= (34.0 \pm 1.8)\,\mathrm{T}\,,\\ B_{\mathrm{V}} &= (34 \pm 2)\,\mathrm{T}\,.\end{aligned}$$

All three results are in good agreement with the literature value at a temperature of 300 K of $B_{\mathrm{lit}} = 33.0\,\mathrm{T}$ [3].

5.2. Discussion

The probably largest source of errors in this measurement was the weak Co-source. As the count rate emitted by the source was very low, it would have been necessary to drastically increase the measurement time in order to obtain results with acceptable uncertainties. In the time given for this experiment, it was not possible to see a clear spectrum for either stainless steel or natural iron.

In the spectrum of natural iron, only the two innermost peaks could be identified. Instead, an unclear wavy form of the data was visible. A measurement was performed to investigate a possible velocity-dependent background effect, but did not show a statistically significant shape. Thus, a velocity-dependent background could not be confirmed. Another possible explanation for this wavy form is an error in the execution of the experiment: As the count rate was very low and no spectrum was visible after the first

overnight measurement of stainless steel, the width of the discriminator window was increased drastically to maximize the count rate. However, it is possible that the final window was too large and caused the outer two peaks to each side of the spectrum of natural iron to blend into each other. This might have resulted in the unidentifiable wavy shape, and explains why the shape is roughly symmetric around the isomer shifted 14.4 keV emission line. It is also possible that the shape is caused by contamination of the iron target or quite simply due to statistical fluctuations. However, these possible explanations are unlikely, as the do not explain the symmetry of the wave shape.

It is of note that in the fits using the Voigt profile to both spectra the parameter σ corresponding to the width of the Gaussian contribution is zero with a large uncertainty ($(0 \pm 170)\,\mathrm{mm\,s^{-1}}$ for stainless steel and $(0 \pm 20)\,\mathrm{mm\,s^{-1}}$ for natural iron). This means that the Gaussian part does not contribute to the fit, and indicates that there is not enough data to distinguish between the natural linewidth and the Gaussian broadening effects. The fact that all results calculated from the fits are compatible when comparing different fit functions also shows that not enough data was taken in order to draw a conclusion of which fit function is most suited to evaluate the spectra. The low values for the reduced χ^2 value of the fits (0.64 and 0.4) indicate too large errors. These could have been reduced by increasing the time measured at each point.

Nevertheless, it was possible to determine the lifetime of the 14.4 keV state of ^{57}Fe as well as the magnetic field strength at the nucleus in good agreement with literature values. However, the results for the lifetime have a large relative uncertainty, which is directly caused by the uncertainties stemming from the fit.

A. Appendix

List of Figures

List of Tables

References

[1] Chechev and Kuzmenko. *Co-57 tables*. http://www.lnhb.fr/nuclides/Co-57_tables.pdf, accessed: 2024-04-26. LNE – LNHB/CEA Table de Radionucléides, 2017.

[2] H. Frauenfelder et al. "Elliptical Polarization of ^{57}Fe Gamma Rays". *Physical Review* 126 (1962), pp. 1065–1075.

[3] Brent Fultz. "Mössbauer Spectrometry". *Characterization of Materials* (2011). DOI: 10.1002/0471266965.

[4] J. Heberle. "Linewidth of Mössbauer absorption". *Nuclear Instruments and Methods* 58.1 (1968), pp. 90–92. DOI: 10.1016/0029-554X(68)90033-5.

[5] S. Margulies, P. Debrunner, and H. Frauenfelder. "Transmission and line broadening in the Mössbauer effect. II". *Nuclear Instruments and Methods* 21 (1963), pp. 217–231. DOI: 10.1016/0029-554x(63)90119-8.

[6] S. Margulies and J.R. Ehrman. "Transmission and line broadening of resonance radiation incident on a resonance absorber". *Nuclear Instruments and Methods* 12 (1961), pp. 131–137. DOI: https://doi.org/10.1016/0029-554X(61)90122-7.

[7] Bogdan Povh et al. *Teilchen und Kerne*. Physics and astronomy online library. Springer Berlin Heidelberg, 2014.

[8] National Institute of Standards and Technology. *Composition of IRON*. https://www.physics.nist.gov/cgi-bin/Star/compos.pl?matno=026, accessed: 2024-02-26.

[9] National Institute of Standards and Technology. *Nuclear Magneton*. https://physics.nist.gov/cgi-bin/cuu/Value?munev|search_for=nuclear+magneton, accessed: 2024-04-08.

[10] Y. Stoll. *Background on the Mößbauer-Effect*. 2022.

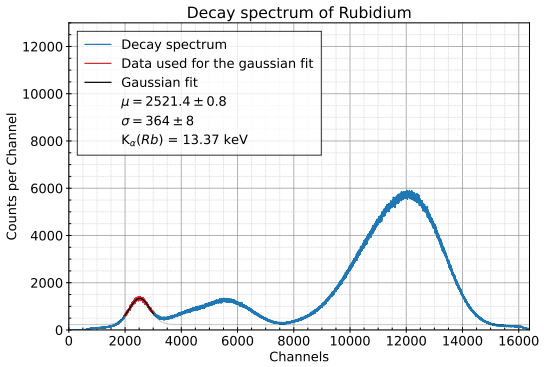
[11] N. J. Stone. "Table of nuclear magnetic dipole and electric quadrupole moments". *Atomic Data and Nuclear Data Tables* 90.1 (2005), pp. 75–176. DOI: https://doi.org/10.1016/j.adt.2005.04.001.

[12] California Institute of Technology. *Experiment 29: The Mossbauer Effect: Hyperfine Splitting*. http://www.sophphx.caltech.edu/Physics_7/Experiment_29.pdf, accessed: 2024-04-09. 2017.

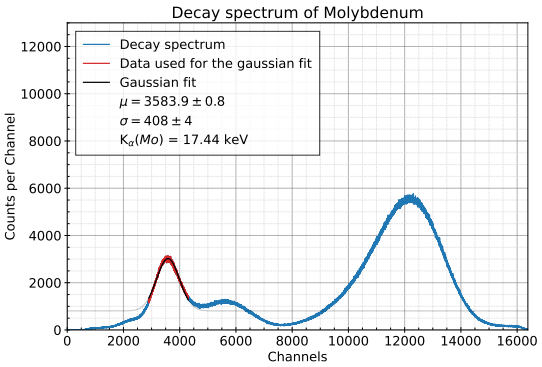
[13] Michael E. Wieser et al. "Atomic weights of the elements 2011 (IUPAC Technical Report)". *Pure and Applied Chemistry* 85.5 (2013), pp. 1047–1078. DOI: 10.1351/PAC-REP-13-03-02.

[14] A. Zwerger et al. *Advanced Lab Courses II, Mößbauer-Effect*. Institut für Mathematik und Physik, Albert Ludwigs Universität, Freiburg im Beisgau, 2017.

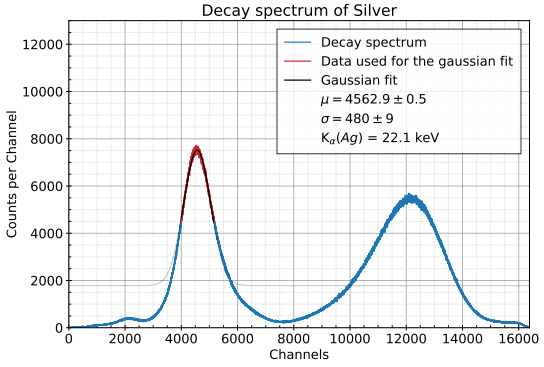
Energy Calibration



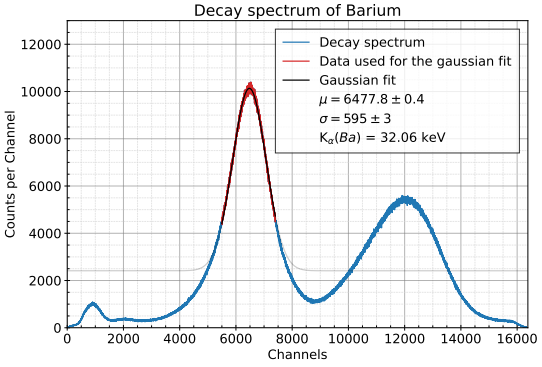
(a) Rubidium



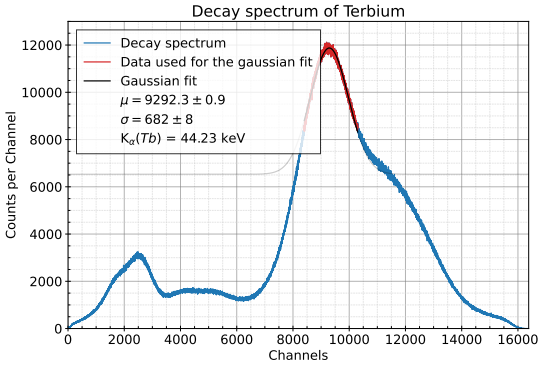
(b) Molybdenum



(c) Silver



(d) Barium



(e) Terbium

Figure 13: Decay spectra of the five different sources used for the energy calibration visible in Fig. 8. The data used for the Gaussian fits of form Eq. (8) is shown in red. The optimum values for μ and σ are shown in the legend as well as the expected energy of each decay.

Source	μ_{Fit}	σ_{Fit}	E_{K_α} [keV]
Rubidium	2521.4 ± 0.8	364 ± 8	13.37
Molybdenum	3583.9 ± 0.8	408 ± 4	17.44
Silver	4562.9 ± 0.5	480 ± 9	22.10
Barium	6477.8 ± 0.4	595 ± 3	32.06
Terbium	9292.3 ± 0.9	682 ± 8	44.23

Table 4: Fit results for the expectation value μ_{Fit} and the standard deviation σ_{Fit} of the Gaussian fits performed in Fig. 13 for all five calibration sources. The expected decay energies taken from [14] are visible as well.

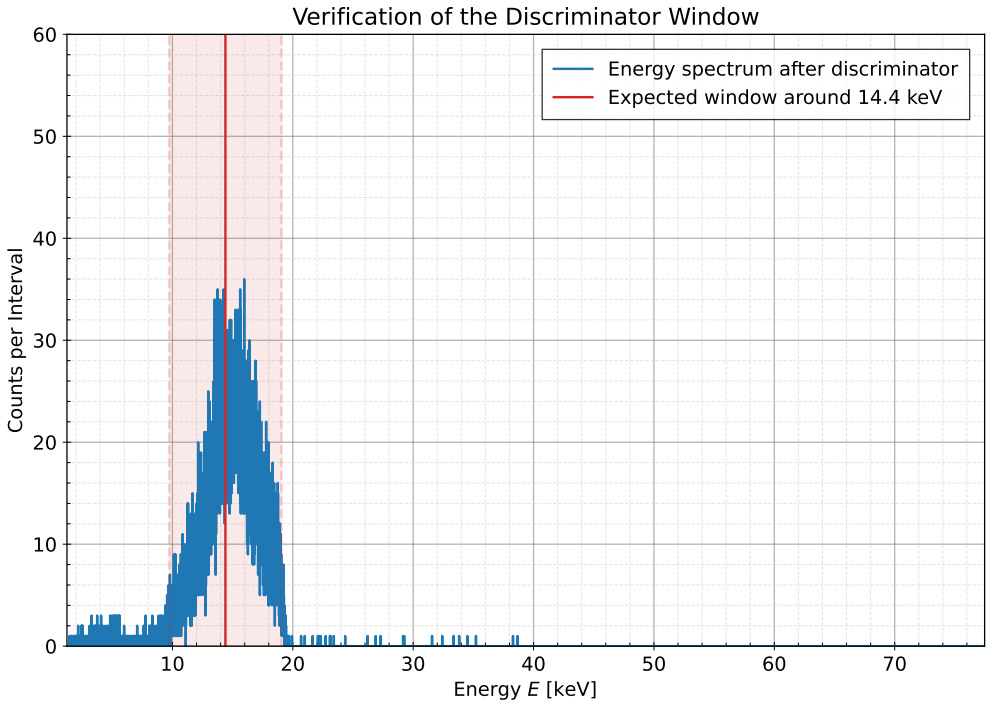


Figure 14: Spectrum of the Rubidium source on the MCA, taken after the discriminator settings were applied according to Section 4.1. The expected energy of 14.4 keV is shown in red, the window chosen on the discriminator in light red. As the data points lie within the expected range, the discriminator calibration was successful.

Transmission of Aluminum

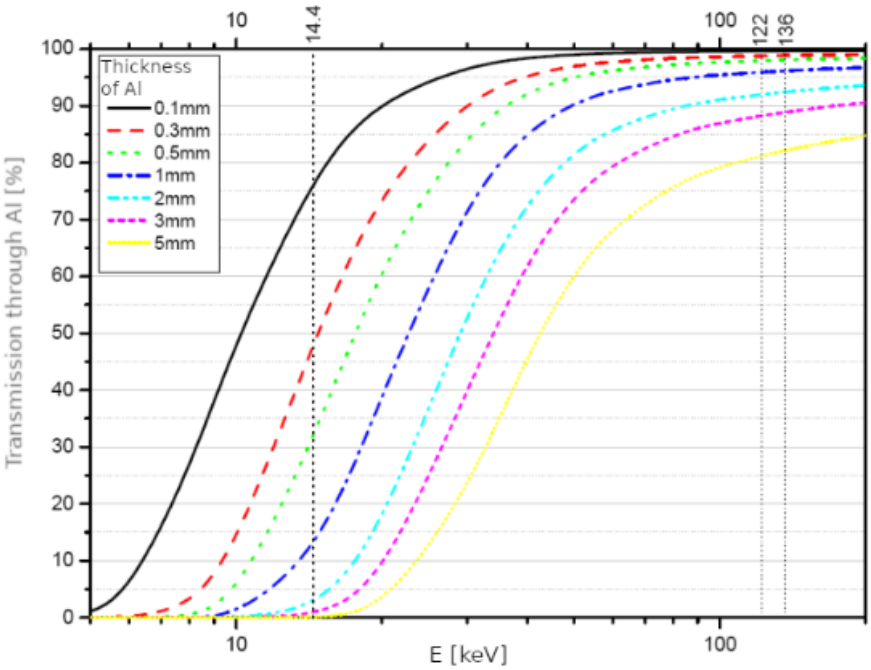


Figure 15: Transmission of radiation at different energies through different thicknesses of aluminum shielding, taken from [14].

Lab Notes

Mößbauer ch 1: yellow, ch 2: blue

Filename Output

L601.PNG Scinti: PA (channel 1) more amplification: We can use larger range of bins
L601.CSV "
L602.PNG "
L602.CSV "
L603.{PNG, CSV} Scinti: AN (channel 2)
L604. " Scinti: PA (ch. 1), AN (ch. 2)
L605 " } same peak "
L606 " "

SCA: Full windows
lower level 0,8, because otherwise there were weird signal

L607. {PNG, CSV} Amplifier uni (ch1) bipolar (ch2), Scinti: Pa.
L608. " Scinti: Pa, Amp.: unipolar → SCA DC INPUT. SCA: Pos out (ch 2)
→ Delay input Delay: ch 1
Adjusted delay: rising slope of SCA on max. of Delay
Scint Pa, Amp. uni → SCA DC In, Pos out → Lin gate: Enable → channel 2
→ Delay → Lin gate input → channel 1

L609. " Lin gate: DC inhibit (example for signals we don't want)
L610. " "
L611. " " signal we like
L612. " Lin Gate: Normal mode

Adjusted delay until normal mode showed middle of peak
SCA: 8,0 delay
Delay amp: 3,5 μs

Final: Amp: Coarse gain: 100 } $\hat{=} \sim 1,45$
gain: 14,8
shaping time: 0,5 μs we can see full peak after linear gate (not cut off)
Timing SCA: lower window: 0,8, upper level: 10,0. Delay: 8,8 (SCA)
Delay amplifier: 3 μs
L613.{...} (ch1) ← csv Set up as of L609, lin. gate: DC inhibit
L614.{...} (ch2)
L615. " (ch1) " lin gate: Norm
L616. (ch2)

Timing SCA lower level 1,6 2,5σ: (2829 ± 403) Bins
upper level 2,0

Stainless steel. Start velocity: 0,05 mm/s, step velocity: 0,05 mm/s, end velocity 1,5 $\frac{mm}{s}$, measure time 120 s

x 4

14.02.2024

~~Acrylic glass~~ ~~(14.2. 8:00 - 14. 8:00)~~

measurement time ~~600s~~. 14.2. 8:00 : 0,01 $\frac{mm}{s}$, 3mm. No shielding. 600 s, 2336 counts

" 8:15 0,2 mm/s 60 mm. Acrylic glass. 600 s, 1963 counts

? thickness: (2,0 ± 0,05) mm (varies depending on where you measure)

~~Background~~:

Stainless steel: 14.2. 8:33 0,2 $\frac{mm}{s}$ 60mm 600s 1183 counts

Stainless steel	Aluminium	0,5 mm	"	"	"	1061 counts	} ! We measured the dip
"	"	1,0 mm	"	"	"	965 counts	
"	"	1,5 mm	"	"	"	940 counts	
no steel	no Aluminium		"	"	"		

Stainless steel	no Aluminium	1 mm/s	100 mm	200 s	401 counts
no steel	no Aluminium	"	"	"	758 counts

Stainless steel	Aluminium	speed	distance	duration	counts	
yes	0 mm	1 $\frac{mm}{s}$	600 mm	600 s	1318	
yes	0,5 mm	"	"	"	1011	
yes	1,0 mm	"	"	"	950	
yes	1,5 mm	"	"	"	952	
yes	2,0 mm	"	"	"	964	
yes	2,5 mm	"	"	"	840	
yes	3,0 mm	"	"	"	798	
yes	4,0 mm	"	"	"	motor died, got it at:	-1 156 153 1 103 104 -1 102 0 1 100 3
"	"	"	"	"	898	
yes	5,0 mm	"	"	"	837	
yes	6,0 mm	"	"	"	818	
yes	7,0 mm	"	"	"	It crashed, remove:	1 103 155 -1 101 138 1 101 126 -1 104 143 1 146 164 -1 144 174
yes	7,0 mm	"	"	"	789	
yes	8,0 mm	"	"	"	681	
yes	9,0 mm	"	"	"	733	
yes	10,0 mm	"	"	"	703	
yes	0,22 mm	"	"	"	1133	
yes	0,44 mm	"	"	"	1118	
yes	0,72 mm	"	"	"	988	
	2,0 mm				944	
	2,5 mm				846	
	3,0 mm				910	

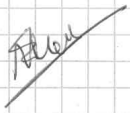
16.02.2024

~~Stainless steel~~ + Aluminium 3mm to check if compton background is indeed independent of velocity
139,9 % ~~absorption~~ at 14,4 keV peak

velocity [$\frac{mm}{s}$]: start: 0,05 step: 0,05 end: 1,5
time: [s] : 120
} did not finish

2 runs

Acrylic glass + aluminium to check for velocity dependent background
velocity [mm/s] start 1 step 1,75 end 8
time [s] : 150



B. Python Code

Energy Calibration

```
# -*- coding: utf-8 -*-

import numpy as np
import pandas as pd
# import matplotlib.pyplot as plt
# import scipy
# import scipy.odr as s_odr
import mymodules.usefultools as mmu
# import mymodules.calculate as mmc
# import mymodules.measure as mmm
# import mymodules.optimize as mmo
import mymodules.functions as mmf
import mymodules.plot as mmp

# verbose = True
# si_format = False
# plot = True
# draft = False
save_images = False
# write_data = False


# %%

targets = ["Rb", "Mo", "Ag", "Ba", "Tb"]
targets_long = ["Rubidium", "Molybdenum", "Silver", "Barium", "Terbium"]
alpha_energies = [13.37, 17.44, 22.1, 32.06, 44.23]


# %%

data = {}
for target in targets:
    data[target] = pd.read_csv(f"../data/calibration/{target.lower()}_800
    .TKA", header=1, names=["counts"])


# %%

legend_loc = [2, 2, 1, 1, 2]

fit_range = [[2000, 3000],
             [2900, 4300],
             [4000, 5150],
             [5500, 7400],
             [8400, 10300]]
# fit_range = [[1500, 2500],
#              [2400, 3400],
#              [3150, 4300],
#              [4300, 6200],
```

```
#               [6900, 8400]]

alpha_bins = []
alpha_bins_err = []
alpha_width = []
alpha_width_err = []

for target_i, target in enumerate(targets):

    # target_i = 4
    # target = targets[target_i]

    data_target = data[target]

    data_target_bins = np.arange(1, len(data_target["counts"]) + 1)
    data_target_counts = data_target["counts"]

    fig, ax = mmp.make_fig(1, 1, grid=True, textsize=16)

    mmp.plot(ax, data_target_bins, data_target_counts, label="Decay
    spectrum")
    mmp.plot(ax,
             data_target_bins[fit_range[target_i][0]:fit_range[target_i
    ][1]],
             data_target_counts[fit_range[target_i][0]:fit_range[target_i
    ][1]],
             label="Data used for the gaussian fit",
             color="tab:red")

    out = mmp.fit(mmf.gauss_poly_0,
                  data_target_bins[fit_range[target_i][0]:fit_range[
    target_i][1]],
                  data_target_counts[fit_range[target_i][0]:fit_range[
    target_i][1]],
                  ax=ax, x_range=data_target_bins, show_results=[1, 2],
                  # kwargs_plot={"linewidth": 4},
                  label="Gaussian fit", color="black")

    alpha_bins.append(out[0][1])
    alpha_bins_err.append(out[1][1])
    alpha_width.append(out[0][2])
    alpha_width_err.append(out[1][2])

    mmp.add_to_legend(ax, rf"K$_\alpha({target})$ = {alpha_energies[
    target_i]} keV")

    ax.set_title(f"Decay spectrum of {targets_long[target_i]}")
    ax.set_xlabel("Channels")
    ax.set_ylabel("Counts per Channel")
    ax.set_xlim(0, len(data_target_bins))
    ax.set_ylim(0, 13000)

    mmp.legend(ax, loc=legend_loc[target_i], textsize=16)
    # break
```

```
    if save_images is True:
        mmp.save_fig(fig, path="../report/figures", name=f"decay_spectrum_{targets_long[target_i].lower()}", extension="pdf")


# %%

mmu.print_to_table(targets_long,
                   mmu.sc_round(alpha_bins, alpha_bins_err, SI=True),
                   mmu.sc_round(alpha_width, alpha_width_err, SI=True),
                   alpha_energies,
                   SI=True,
                   environment=True,
                   header=True,
                   copy=False
                   )


# %%

fig, ax = mmp.make_fig(grid=True)

mmp.plot(ax,
         alpha_energies, alpha_bins,
         y_err=alpha_bins_err,
         config="scatter",
         s=100,
         label=r"K$_\alpha$ channels")

out_energy_calibration = mmp.fit(mmf.poly_1,
                                 alpha_energies, alpha_bins,
                                 conf=True,
                                 show_results=False,
                                 print_results=True,
                                 color="black",
                                 x_range=np.linspace(10, 50, 100),
                                 label="Linear calibration fit\nwith 2-$\\sigma$ confidence interval",
                                 ax=ax)

ax.set_title(r"Energy calibration using the K$_\alpha$ decay energies of known sources")
ax.set_xlabel(r"Energy $E$ of the K$_\alpha$ decays [keV]")
ax.set_ylabel("Channels")

ax.set_xlim(10, 50)

E = 14.4  # keV
bin_14_4 = mmf.poly_1(E, *out_energy_calibration[0])
bin_14_4_err = np.sqrt((E * out_energy_calibration[1][0])**2 + out_energy_calibration[1][1]**2)
factor = 5

# ax.axhline(bin_14_4)
```

```
148 # ax.axhline(bin_14_4 + factor * bin_14_4_err)
149 # ax.axhline(bin_14_4 - factor * bin_14_4_err)
150 ax.axvline(E, color="tab:red", label="Iron decay energy: 14.4 keV")
151
152 mmp.legend(ax, loc=4)
153
154 if save_images:
155     mmp.save_fig(fig, path="../report/figures", name="energy_calibration"
    , extension="pdf")
156
157 print(bin_14_4, bin_14_4_err)
158 print(bin_14_4 - factor * bin_14_4_err)
159 print(bin_14_4 + factor * bin_14_4_err)
160 print(bin_14_4 / len(data_target_bins) * 10 - factor * bin_14_4_err / len
    (data_target_bins) * 10)
161 print(bin_14_4 / len(data_target_bins) * 10 + factor * bin_14_4_err / len
    (data_target_bins) * 10)
162
163
164 # %%
165
166 def energy_from_bins(bins):
167     return (bins - out_energy_calibration[0][1]) / out_energy_calibration
    [0][0]
168
169
170 # energy spectrum after window was applied. The source was the Rb source
    for calibration
171 data_cut = pd.read_csv("../data/calibration/14_4_final_window_applied.TKA
    ", header=1, names=["counts"])
172
173 data_cut_bins = np.arange(1, len(data_cut["counts"]) + 1)
174 data_cut_counts = data_cut["counts"]
175
176 fig, ax = mmp.make_fig(grid=True)
177
178 x = energy_from_bins(data_cut_bins)
179 y = data_cut_counts
180
181 mmp.plot(ax, x, y, label="Energy spectrum after discriminator")
182
183 mmp.plot(ax, 14.4, None, x_err=[[energy_from_bins(bin_14_4 - 1000) -
    14.4], [14.4 - energy_from_bins(bin_14_4 + 1000)]], color="tab:red",
    config="vspan", label="Expected window around 14.4 keV")
184 # ax.axvline(energy_from_bins(bin_14_4 - factor * bin_14_4_err), color="
    tab:red")
185 # ax.axvline(energy_from_bins(bin_14_4 + factor * bin_14_4_err), color="
    tab:red")
186
187 ax.set_title("Verification of the Discriminator Window")
188 ax.set_xlabel(r"Energy $E$ [keV]")
189 ax.set_ylabel("Counts per Interval")
190 ax.set_xlim(np.min(x), np.max(x))
191 ax.set_ylim(0, 60)
```

```
192
193 mmp.legend(ax, loc=1)
194
195 if save_images:
196     mmp.save_fig(fig, path="../report/figures", name="
    spectrum_after_discriminator", extension="pdf")
```

Compton Background

```
1  # -*- coding: utf-8 -*-
2
3  import numpy as np
4  import pandas as pd
5  import matplotlib.pyplot as plt
6  # import scipy
7  # import scipy.odr as s_odr
8  import mymodules.usefultools as mmu
9  # import mymodules.calculate as mmc
10 # import mymodules.measure as mmm
11 # import mymodules.optimize as mmo
12 import mymodules.functions as mmf
13 import mymodules.plot as mmp
14
15 # verbose = True
16 # si_format = False
17 # plot = True
18 # draft = False
19 # save_images = False
20 # write_data = False
21
22
23 # %%
24
25 # sum f two exponential decays withOUT constants
26 def f_exp_decay_sum_of_2(t, A, tau, B, gamma):
27     t = np.array(t)
28     return A * np.exp(-t / tau) + B * np.exp(-t / gamma)
29
30
31 def f_exp_decay_sum_of_2_p0(x, y, tau_lit=1):
32     return np.array([y[0],
33                      tau_lit,
34                      y[0],
35                      tau_lit])
36
37
38 exp_decay_sum_of_2 = mmf.fit_function(
39     f=f_exp_decay_sum_of_2,
40     p0=f_exp_decay_sum_of_2_p0,
41     bounds=([-np.inf, 0, -np.inf, 0], [np.inf, np.inf, np.inf, np.inf]),
42     # f_err=f_exp_decay_err,
43     description="sum of two exponential decays",
44     params=["A", "tau", "B", "gamma"],
```

```
    params_tex=["A", "\\tau", "B", "\\gamma"],
    formula="A exp(-t / tau) + B exp(-t / gamma)",
    formula_tex="A \\cdot \\exp{{(-t / \\tau)}} + B \\cdot \\exp{{(-t / \\gamma)}}",
    docstring='''
usage: y = exp_decay(x, A, tau, B, gamma)
'''
)


# %%

data = pd.read_csv("../data/compton_background/moessbauer_compton_background.txt", header=0, delimiter="\t", decimal=",", names=["velocity", "time", "counts"])
data["time"] = data["time"] / 1000  # time in s

cps = []
cps_err = []
for i in range(int(len(data["time"]) / 6)):
    cps.append(data["counts"][6 * i:6 * i + 6].sum() / data["time"][6 * i:6 * i + 6].sum())
    cps_err.append(np.sqrt(data["counts"][6 * i:6 * i + 6].sum()) / data["time"][6 * i:6 * i + 6].sum())


# %%

absorber = [0, 0.5, 1, 1.5, 2, 2.5, 3, 4, 5, 6, 7, 8, 9, 10, 0.22, 0.44, 0.72,
            2, 2.5, 3]
# counts = [1318, 1011, 950, 952, 964, 840, 798, 898, 837, 818, 789, 681, 733,
#           703, 1133, 1118, 988,
#           944, 846, 910]
# absorber = [0, 0.5, 1, 1.5, 2, 2.5, 3, 4, 5, 6, 7, 8, 9, 10, 0.22, 0.44, 0.72]
# counts = [1318, 1011, 950, 952, 964, 840, 798, 898, 837, 818, 789, 681, 733,
#           703, 1133, 1118, 988]

fig, ax = mmp.make_fig(grid=True)

# fig.set_dpi(100)

x = absorber
y = cps
y_err = cps_err
mmp.plot(ax, x, y, y_err=y_err, config="scatter", label="Measured count rate")

xx = np.linspace(0, 10.5, 100)
out_compton = mmp.fit(exp_decay_sum_of_2,
                      x, y,
```

```
                      y_err=y_err,
                      p0=[900, 30, 300, 1],
                      bounds=True,
                      x_range=xx,
                      show_results=False,
                      label="Fitted sum of two exponential decays",
                      ax=ax)


mmp.plot(ax, xx, mmf.exp_decay(xx, out_compton[0][0], out_compton[0][1]), label="Contribution of the Compton Background", color="tab:red")
mmp.add_to_legend(ax, f"A = ${mmu.sc_round(out_compton[0], out_compton[1], plot=True)[0]}$ cps")
# mmp.plot(ax, xx, mmf.exp_decay(xx, out_compton[0][2], out_compton[0][3]) / mmf.exp_decay(0, out_compton[0][2], out_compton[0][3]) color="tab:red")

ax.set_title("Determination of the Compton Background")
ax.set_xlabel(r"Absorber thickness $d$ of Al [mm]")
ax.set_ylabel("Count rate [cps]")
ax.set_xlim(-0, 10.5)
# ax.set_ylim(1, 5)


mmp.legend(ax, loc=1)

# mmp.save_fig(fig, path="../report/figures", name="compton_background", extension="pdf")

plt.show()


# %%

xxx = np.linspace(0, 4, 9)
mmu.print_to_table(xxx, 1 - mmf.exp_decay(xxx, out_compton[0][2], out_compton[0][3]) / mmf.exp_decay(0, out_compton[0][2], out_compton[0][3]))


# %%

ac = 1963 / 600  # count rate with acrylic glass
nac = 2336 / 600  # count rate without acrylic glass

ac_err = np.sqrt(1963) / 600
nac_err = np.sqrt(2336) / 600

acrylic_absorption = ac / nac
acrylic_absorption_err = np.sqrt((1 / nac * ac_err)**2 + (ac / nac**2 * nac_err)**2)

print(acrylic_absorption, acrylic_absorption_err)

```

```
d_plexi = 0.2  # cm
d_plexi_err = 0.005  # cm
mu_per_rho = 1.101
rho = 1.19

T_theo = np.exp(-mu_per_rho * rho * d_plexi)
T_theo_err = np.sqrt((mu_per_rho * rho * np.exp(-mu_per_rho * rho *
    d_plexi) * d_plexi_err)**2)
print(T_theo, T_theo_err)


# %%

# With acrylic glass absorption because measured with stainless steel
    absorber
compton_cps = out_compton[0][0]
compton_cps_err = out_compton[1][0]

print(compton_cps, compton_cps_err)


# %%

data_background = {"compton_cps": compton_cps,
                   "compton_cps_err": compton_cps_err,
                   "acrylic_absorption": acrylic_absorption,
                   "acrylic_absorption_err": acrylic_absorption_err}

mmu.save_json(data_background, "./data_compton.json")
```

Stainless Steel Analysis

```
# -*- coding: utf-8 -*-

import numpy as np
import pandas as pd
# import matplotlib.pyplot as plt
# import matplotlib as mpl
import scipy
# import scipy.odr as s_odr
import mymodules.usefultools as mmu
# import mymodules.calculate as mmc
# import mymodules.measure as mmm
# import mymodules.optimize as mmo
import mymodules.functions as mmf
import mymodules.plot as mmp

# verbose = True
# si_format = False
# plot = True
# draft = False
save_images = False
# write_data = False
```

```
22
23
24 # %%
25
26 # voigt function
27 def f_voigt(x, A=1, mu=0, sigma=1, gamma=1, C=0):
28     x = np.array(x)
29     return A * scipy.special.voigt_profile((x - mu), sigma, gamma) + C
30
31
32 voigt = mmf.fit_function(
33     f=f_voigt,
34     description="voigt profile",
35     params=["A", "mu", "sigma", "gamma", "C"],
36     params_tex=["A", "\\mu", "\\sigma", "\\gamma", "C"],
37 )
38
39
40 # %%
41
42 data = pd.read_csv("../data/stainless_steel_next_try/
       moessbauer_stainless_steel.txt", header=0, delimiter="\t", decimal=","
       , names=["velocity", "time", "counts"])
43 data["time"] = data["time"] / 1000  # time in s
44
45 print("Measurement time [h]:", data["time"].sum() / 3600)
46 print("Time per point [s]:", data["time"].sum() / len(data["velocity"].
       unique()))
47
48
49 # %%
50
51 data_compton = mmu.read_json("./data_compton.json")
52 data_compton
53
54
55 # %%
56
57 velocities = np.sort(data["velocity"].unique())
58 cps_raw = []
59 cps_raw_err = []
60
61 for velocity in velocities:
62     data_velocity = data[data["velocity"] == velocity]
63     cps_raw.append(data_velocity["counts"].sum() / data_velocity["time"].
       sum())
64     cps_raw_err.append(np.sqrt(data_velocity["counts"].sum()) /
       data_velocity["time"].sum())
65
66 cps_raw, cps_raw_err = mmu.convert_to_array(cps_raw, cps_raw_err)
67
68
69 # %%
70
```

```
71 cps = (cps_raw - data_compton["compton_cps"]) / data_compton["
       acrylic_absorption"]
72 cps_err = np.sqrt((1 / data_compton["acrylic_absorption"] * cps_raw_err)
       **2
73                   + (1 / data_compton["acrylic_absorption"] *
       data_compton["compton_cps_err"])**2
74                   + ((cps_raw - data_compton["compton_cps"]) /
       data_compton["acrylic_absorption"]**2 * data_compton["
       acrylic_absorption_err"])**2)
75
76
77 # %%
78
79 fig, ax = mmp.make_fig(grid=True)
80
81 mmp.plot(ax, velocities, cps, y_err=cps_err, label="Count rate of
       reemission", config="scatter")
82 mmp.add_to_legend(ax, " ")
83 mmp.add_to_legend(ax, " ")
84
85 out_gauss = mmp.fit(mmf.normal,
86                     velocities, cps,
87                     y_err=cps_err,
88                     absolute_sigma=True,
89                     p0=[-0.2, 0.2, 0.2, 0.6],
90                     x_range=np.linspace(-1.6, 1.6, 200),
91                     # show_results=1,
92                     result_units=["cps", "mm s$^{-1}$", "mm s$^{-1}$", "
       cps"],
93                     print_results=True,
94                     # significant=2,
95                     color="darkgreen",
96                     kwargs_plot={"linewidth": 2.5},
97                     ax=ax, label="Gaussian fit")
98
99 out_lorentz = mmp.fit(mmf.lorentzian,
100                       velocities, cps,
101                       y_err=cps_err,
102                       absolute_sigma=True,
103                       p0=[-0.2, 0.2, 0.2, 0.6],
104                       x_range=np.linspace(-1.6, 1.6, 200),
105                       # show_results=1,
106                       print_results=True,
107                       # significant=2,
108                       result_units=["cps", "mm s$^{-1}$", "mm s$^{-1}$",
       "cps"],
109                       ax=ax, color="purple",
110                       kwargs_plot={"linewidth": 2.5},
111                       label="Lorentzian fit")
112
113 out_voigt = mmp.fit(voigt,
114                     velocities, cps,
115                     y_err=cps_err,
116                     absolute_sigma=True,
```

```
                    p0=[-0.2, 0.2, 0.2, 0.2, 0.6],
                    x_range=np.linspace(-1.6, 1.6, 200),
                    ax=ax,
                    # show_results=1,
                    print_results=True,
                    # significant=2,
                    result_units=["cps", "mm s$^{-1}$", "mm s$^{-1}$", "mm s$^{-1}$", "cps"],
                    color="red",
                    kwargs_plot={"linestyle": (0, (1, 3)), "linewidth": 2.5},
                    label="Voigt fit")

out = {"gauss": out_gauss,
       "lorentz": out_lorentz,
       "voigt": out_voigt}

ax.set_title("Reemission spectrum of stainless steel")
ax.set_xlabel(r"Velocity $v$ [mm s$^{-1}$]")
ax.set_ylabel(r"Count rate [cps]")

ax.set_xlim(-1.6, 1.6)
ax.set_ylim(0.24, 0.95)

mmp.legend(ax, loc=1, ncols=2)

if save_images:
    mmp.save_fig(fig, path="../report/figures", name="stainless_steel", extension="pdf")


# %%

def energy_from_speed(E, E_err, v, v_err):
    """
    E in eV, v in m/s
    [Delta E, Delta E err] output in eV
    """

    c = scipy.constants.c
    return np.array([E * v / c, np.sqrt((E / c * v_err)**2 + (v / c * E_err)**2)])


e_charge = scipy.constants.e
E_iron = 14.41295e3  # eV
E_iron_err = 0.00031e3  # eV
for profile in ["gauss", "lorentz", "voigt"]:
    # print(out[profile][0][1], out[profile][1][1])
    print(f"Isomer shift calculated with {profile} profile:")
    print(f"{energy_from_speed(E_iron, E_iron_err, out[profile][0][1]*1e-3, out[profile][1][1]*1e-3)*1e9} neV")


```

```
166 # %%
167
168 f_A = 0.8  # debye waller factor of the absorber
169
170 m_iron = 55.845  # g/mol
171 m_iron_err = 0.002  # g/mol
172 rho_iron = 7.87400  # g/cm^3
173 p = 0.7
174 p_err = 0.05
175 avogadro = scipy.constants.N_A
176 n_A = p * rho_iron * avogadro / m_iron * 1e6  # number of iron atoms per
        m^3
177 n_A_err = rho_iron * avogadro * np.sqrt((1 / m_iron * p_err)**2 + (p /
        m_iron**2 * m_iron_err)**2) * 1e6
178 print("n_A, n_A_err", n_A, n_A_err)
179
180 beta = 0.022  # fraction of 57Fe in the isotope mixture
181
182 h = scipy.constants.h
183 c = scipy.constants.c
184 e_charge = scipy.constants.e
185 E_iron = 14.41295e3 * e_charge  # J
186 E_iron_err = 0.00031e3 * e_charge  # J
187 alpha = 8.55
188 alpha_err = 0.12
189 I_e = 3 / 2
190 I_g = 1 / 2
191 sigma_0 = 1 / (2 * np.pi) * (h * c / E_iron)**2 * (2 * I_e + 1) / (2 *
        I_g + 1) / (1 + alpha)  # calculate
192 sigma_0_err = 1 / (2 * np.pi) * (h * c)**2 * (2 * I_e + 1) / (2 * I_g +
        1) * np.sqrt((2 / ((1 + alpha) * E_iron**3) * E_iron_err)**2 + (1 /
        ((1 + alpha)**2 * E_iron**2) * alpha_err)**2)
193 print("sigma_0, sigma_0_err = ", sigma_0, sigma_0_err)
194
195 d_A = 25e-6  # m, absorber thickness from instructions
196 T_A = f_A * n_A * beta * sigma_0 * d_A  # effective absorber thickness
197 T_A_err = f_A * beta * d_A * np.sqrt((n_A * sigma_0_err)**2 + (n_A_err *
        sigma_0)**2)
198 print("T_A, T_A_err", T_A, T_A_err)
199
200
201 # %%
202
203 f_S_dict = {}
204 print("f_S, f_S_err")
205 for function, profile in zip([mmf.normal, mmf.lorentzian, voigt], ["gauss
        ", "lorentz", "voigt"]):
206     C = out[profile][0][-1]
207     C_err = out[profile][1][-1]
208     N_at_mu = function(out[profile][0][1], *out[profile][0])
209     N_at_mu_err = function(out[profile][0][1] + out[profile][1][1], *out[
        profile][0]) - function(out[profile][0][1], *out[profile][0])
210
```

```
211     f_S = (C - N_at_mu) / C / (1 - np.exp(-0.5 * T_A) * scipy.special.jv
    (0, 1j * 0.5 * T_A)).real
212     f_S_err = np.sqrt(
213         (-N_at_mu / C**2 / (1 - np.exp(-0.5 * T_A) * scipy.special.jv(0,
    1j * 0.5 * T_A)) * C_err)**2 +
214         (-1 / C / (1 - np.exp(-0.5 * T_A) * scipy.special.jv(0, 1j * 0.5
    * T_A)) * N_at_mu_err)**2 +
215         (-(C - N_at_mu) / (2 * C) * np.exp(-0.5 * T_A) * (scipy.special.
    jv(0, 1j * 0.5 * T_A) + 1j * scipy.special.jv(1, 1j * 0.5 * T_A)) / (1
     - np.exp(-0.5 * T_A) * scipy.special.jv(0, 1j * 0.5 * T_A))**2 *
    T_A_err)**2
216     ).real
217     f_S_dict[profile] = (f_S, f_S_err)
218     print(profile, f_S, f_S_err)
219     # print(profile, mmu.sc_round(f_S, f_S_err, SI=True))
220
221
222 # %%
223
224 print("T_S, T_S_err")
225 for profile in ["gauss", "lorentz", "voigt"]:
226     f_S = f_S_dict[profile][0]
227     f_S_err = f_S_dict[profile][1]
228     n_S = n_A
229     n_S_err = n_A_err
230     beta = 1
231     d_S = 100e-10  # m
232     T_S = f_S * n_S * beta * sigma_0 * d_S  # effective absorber
    thickness
233     T_S_err = f_S * beta * d_S * np.sqrt((n_S * sigma_0_err)**2 + (
    n_S_err * sigma_0)**2)
234     # print(profile, T_S, T_S_err)
235     print(profile, mmu.sc_round(T_S, T_S_err, SI=True))
236
237
238 # %%
239
240 W = 2 * (1.01 + 0.145 * T_A - 0.0025 * T_A**2)
241 W_err = abs(2 * (0.145 - 2 * 0.0025 * T_A) * T_A_err)
242 print("W, W_err", mmu.sc_round(W, W_err, SI=True))
243
244
245 # %%
246
247 print("FWHM:")
248
249 Gamma_G = 2 * np.sqrt(2 * np.log(2)) * out["gauss"][0][2]
250 Gamma_G_err = abs(2 * np.sqrt(2 * np.log(2)) * out["gauss"][1][2])
251 Gamma_L = 2 * out["lorentz"][0][2]
252 Gamma_L_err = abs(2 * out["lorentz"][1][2])
253 Gamma_V = 2 * out["voigt"][0][3]
254 Gamma_V_err = abs(2 * out["voigt"][1][3])
255
256 print("Gauss: ", mmu.sc_round(Gamma_G, Gamma_G_err, SI=True), "mm/s")
```

```
257 print("Lorentz: ", mmu.sc_round(Gamma_L, Gamma_L_err, SI=True), "mm/s")
258 print("Voigt: ", mmu.sc_round(Gamma_V, Gamma_V_err, SI=True), "mm/s")
259
260 # %%
261
262 print("Gamma_nat:")
263
264 Gamma_G_nat = Gamma_G / W
265 Gamma_G_nat_err = np.sqrt((1 / W * Gamma_G_err)**2 + (Gamma_G / W**2 *
        W_err)**2)
266 Gamma_L_nat = Gamma_L / W
267 Gamma_L_nat_err = np.sqrt((1 / W * Gamma_L_err)**2 + (Gamma_L / W**2 *
        W_err)**2)
268 Gamma_V_nat = Gamma_V / W
269 Gamma_V_nat_err = np.sqrt((1 / W * Gamma_V_err)**2 + (Gamma_V / W**2 *
        W_err)**2)
270
271 # convert to energies
272 E_iron = 14.41295e3  # eV
273 E_iron_err = 0.00031e3  # eV
274 Gamma_G_nat, Gamma_G_nat_err = energy_from_speed(E_iron, E_iron_err,
        Gamma_G_nat * 1e-3, Gamma_G_nat_err * 1e-3)
275 Gamma_L_nat, Gamma_L_nat_err = energy_from_speed(E_iron, E_iron_err,
        Gamma_L_nat * 1e-3, Gamma_L_nat_err * 1e-3)
276 Gamma_V_nat, Gamma_V_nat_err = energy_from_speed(E_iron, E_iron_err,
        Gamma_V_nat * 1e-3, Gamma_V_nat_err * 1e-3)
277
278 print("Gauss: ", mmu.sc_round(Gamma_G_nat, Gamma_G_nat_err, SI=True), "eV
        ")
279 print("Lorentz: ", mmu.sc_round(Gamma_L_nat, Gamma_L_nat_err, SI=True), "
        eV")
280 print("Voigt: ", mmu.sc_round(Gamma_V_nat, Gamma_V_nat_err, SI=True), "eV
        ")
281
282
283 # %%
284
285 print("Lifetime tau")
286 hbar = scipy.constants.hbar / scipy.constants.e  # in eV
287
288 tau_G = hbar / Gamma_G_nat
289 tau_G_err = abs(hbar / Gamma_G_nat**2 * Gamma_G_nat_err)
290 tau_L = hbar / Gamma_L_nat
291 tau_L_err = abs(hbar / Gamma_L_nat**2 * Gamma_L_nat_err)
292 tau_V = hbar / Gamma_V_nat
293 tau_V_err = abs(hbar / Gamma_V_nat**2 * Gamma_V_nat_err)
294
295 # print("Gauss: ", tau_G, tau_G_err)
296 # print("Lorentz: ", tau_L, tau_L_err)
297 # print("Voigt: ", tau_V, tau_V_err)
298 print("Gauss: ", mmu.sc_round(tau_G, tau_G_err, SI=True))
299 print("Lorentz: ", mmu.sc_round(tau_L, tau_L_err, SI=True))
300 print("Voigt: ", mmu.sc_round(tau_V, tau_V_err, SI=True))
```

Natural Iron Analysis

```
# -*- coding: utf-8 -*-

import numpy as np
import pandas as pd
# import matplotlib.pyplot as plt
# import matplotlib as mpl
import scipy
# import scipy.odr as s_odr
import mymodules.usefultools as mmu
# import mymodules.calculate as mmc
# import mymodules.measure as mmm
# import mymodules.optimize as mmo
import mymodules.functions as mmf
import mymodules.plot as mmp

# verbose = True
# si_format = False
# plot = True
# draft = False
save_images = False
# write_data = False


# %%

# symmetric sum of 2 gaussian functions
def f_sum_2_gauss(x,
                  A1=1, mu1=0, sigma1=1,
                  # A2=1, mu2=0, sigma2=1,
                  # A3=1, mu3=0, sigma3=1,
                  mu0=0, C=0):
    x = np.array(x)
    return A1 / (sigma1 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu0 - mu1) / (sigma1))**2) + \
        A1 / (sigma1 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu0 + mu1) / (sigma1))**2) + \
        C
    # A2 / (sigma2 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu0 - mu2) / (sigma2))**2) + \
    # A3 / (sigma3 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu0 - mu3) / (sigma3))**2) + \
    # A2 / (sigma2 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu0 + mu2) / (sigma2))**2) + \
    # A3 / (sigma3 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu0 + mu3) / (sigma3))**2) + \


sum_2_gauss = mmf.fit_function(
    f=f_sum_2_gauss,
    bounds=([-np.inf, 0, 0.0001,
             # -np.inf, 2, 0.001,
             # -np.inf, 4, 0.001,
```

```
47              -2, 0.5],
48             [0, 2, 10,
49              # 0, 4, 1,
50              # 0, 6, 1,
51              2, 1]),
52     description="sum of 2 gaussian functions with common offset C",
53     params=["A1", "mu1", "sigma1",
54             # "A2", "mu2", "sigma2",
55             # "A3", "mu3", "sigma3",
56             "mu0", "C"],
57 )
58
59
60 # %%
61
62 # symmetric sum of 2 lorentz functions
63 def f_sum_2_lorentz(x,
64                     A1=1, mu1=0, gamma1=1,
65                     # A2=1, mu2=0, gamma2=1,
66                     # A3=1, mu3=0, gamma3=1,
67                     mu0=0, C=0):
68     x = np.array(x)
69     return (A1 * gamma1) / np.pi * 1 / ((x - mu0 - mu1)**2 + gamma1**2) +
       \
70         (A1 * gamma1) / np.pi * 1 / ((x - mu0 + mu1)**2 + gamma1**2) + \
71         C
72     # (A2 * gamma2) / np.pi * 1 / ((x - mu0 - mu2)**2 + gamma2**2) + \
73     # (A3 * gamma3) / np.pi * 1 / ((x - mu0 - mu3)**2 + gamma3**2) + \
74     # (A2 * gamma2) / np.pi * 1 / ((x - mu0 + mu2)**2 + gamma2**2) + \
75     # (A3 * gamma3) / np.pi * 1 / ((x - mu0 + mu3)**2 + gamma3**2) + \
76
77
78 sum_2_lorentz = mmf.fit_function(
79     f=f_sum_2_lorentz,
80     bounds=([-np.inf, 0, 0.001,
81              # -np.inf, 2, 0.001,
82              # -np.inf, 4, 0.001,
83              -2, -np.inf],
84             [0, 2, 1,
85              # 0, 4, 1,
86              # 0, 6, 1,
87              2, np.inf]),
88     description="sum of 2 lorentz functions with common offset C",
89     params=["A1", "mu1", "gamma1",
90             # "A2", "mu2", "gamma2",
91             # "A3", "mu3", "gamma3",
92             "mu0", "C"],
93 )
94
95
96 # %%
97
98 # symmetric sum of 2 voigt profiles
99 def f_sum_2_voigt(x,
```

```
100                     A1=1, mu1=0, sigma1=1, gamma1=1,
101                     # A2=1, mu2=0, sigma2=1, gamma2=1,
102                     # A3=1, mu3=0, sigma3=1, gamma3=1,
103                     mu0=0, C=0):
104     x = np.array(x)
105     return A1 * scipy.special.voigt_profile((x - mu0 - mu1), sigma1,
    gamma1) + \
106         A1 * scipy.special.voigt_profile((x - mu0 + mu1), sigma1, gamma1)
     + \
107         C
108     # A2 * scipy.special.voigt_profile((x - mu0 - mu2), sigma2, gamma2) +
     \
109     # A3 * scipy.special.voigt_profile((x - mu0 - mu3), sigma3, gamma3) +
     \
110     # A2 * scipy.special.voigt_profile((x - mu0 + mu2), sigma2, gamma2) +
     \
111     # A3 * scipy.special.voigt_profile((x - mu0 + mu3), sigma3, gamma3) +
     \
112
113
114 sum_2_voigt = mmf.fit_function(
115     f=f_sum_2_voigt,
116     bounds=([-np.inf, 0, 0.001, 0.001,
117              # -np.inf, 2, 0.001, 0.001,
118              # -np.inf, 4, 0.001, 0.001,
119              -2, -np.inf],
120             [0, 2, 1, 1,
121              # 0, 4, 1, 1,
122              # 0, 6, 1, 1,
123              2, np.inf]),
124     description="sum of 2 voigt profiles with common offset C",
125     params=["A1", "mu1", "sigma1", "gamma1",
126             # "A2", "mu2", "sigma2", "gamma2",
127             # "A3", "mu3", "sigma3", "gamma3",
128             "mu0", "C"],
129 )
130
131 # %%
132
133 # # symmetric sum of 6 gaussian functions
134 # def f_sum_2_gauss(x,
135 #                   A1=1, mu1=0, sigma1=1,
136 #                   A2=1, mu2=0, sigma2=1,
137 #                   A3=1, mu3=0, sigma3=1,
138 #                   A4=1, mu4=0, sigma4=1,
139 #                   A5=1, mu5=0, sigma5=1,
140 #                   A6=1, mu6=0, sigma6=1,
141 #                   C=0):
142 #     x = np.array(x)
143 #     return A1 / (sigma1 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x -
    mu1) / (sigma1))**2) + \
144 #         A2 / (sigma2 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu2)
     / (sigma2))**2) + \
```

```
145 #         A3 / (sigma3 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu3)
      / (sigma3))**2) + \
146 #         A4 / (sigma4 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu4)
      / (sigma4))**2) + \
147 #         A5 / (sigma5 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu5)
      / (sigma5))**2) + \
148 #         A6 / (sigma6 * np.sqrt(2 * np.pi)) * np.exp(-1 / 2 * ((x - mu6)
      / (sigma6))**2) + \
149 #         C
150
151
152 # sum_2_gauss = mmf.fit_function(
153 #     f=f_sum_2_gauss,
154 #     bounds=([-np.inf, -6, 0.1,
155 #              -np.inf, -4, 0.1,
156 #              -np.inf, -2, 0.1,
157 #              -np.inf, 0, 0.1,
158 #              -np.inf, 2, 0.1,
159 #              -np.inf, 4, 0.1,
160 #              -np.inf],
161 #             [0, -4, 1,
162 #              0, -2, 1,
163 #              0, 0, 1,
164 #              0, 2, 1,
165 #              0, 4, 1,
166 #              0, 6, 1,
167 #              np.inf]),
168 #     description="sum of 6 gaussian functions with common offset C",
169 #     params=["A1", "mu1", "sigma1",
170 #             "A2", "mu2", "sigma2",
171 #             "A3", "mu3", "sigma3",
172 #             "A4", "mu4", "sigma4",
173 #             "A5", "mu5", "sigma5",
174 #             "A6", "mu6", "sigma6",
175 #             "C"],
176 # )
177
178
179 # # %%
180
181 # # symmetric sum of 6 lorentz functions
182 # def f_sum_2_lorentz(x,
183 #                     A1=1, mu1=0, gamma1=1,
184 #                     A2=1, mu2=0, gamma2=1,
185 #                     A3=1, mu3=0, gamma3=1,
186 #                     A4=1, mu4=0, gamma4=1,
187 #                     A5=1, mu5=0, gamma5=1,
188 #                     A6=1, mu6=0, gamma6=1,
189 #                     C=0):
190 #     x = np.array(x)
191 #     return (A1 * gamma1) / np.pi * 1 / ((x - mu1)**2 + gamma1**2) + \
192 #         (A2 * gamma2) / np.pi * 1 / ((x - mu2)**2 + gamma2**2) + \
193 #         (A3 * gamma3) / np.pi * 1 / ((x - mu3)**2 + gamma3**2) + \
194 #         (A4 * gamma4) / np.pi * 1 / ((x - mu4)**2 + gamma4**2) + \
```

```
#         (A5 * gamma5) / np.pi * 1 / ((x - mu5)**2 + gamma5**2) + \
#         (A6 * gamma6) / np.pi * 1 / ((x - mu6)**2 + gamma6**2) + \
#         C


# sum_2_lorentz = mmf.fit_function(
#     f=f_sum_2_lorentz,
#     bounds=([-np.inf, -6, 0.1,
#              -np.inf, -4, 0.1,
#              -np.inf, -2, 0.1,
#              -np.inf, 0, 0.1,
#              -np.inf, 2, 0.1,
#              -np.inf, 4, 0.1,
#              -np.inf],
#             [0, -4, 1,
#              0, -2, 1,
#              0, 0, 1,
#              0, 2, 1,
#              0, 4, 1,
#              0, 6, 1,
#              np.inf]),
#     description="sum of 6 lorentz functions with common offset C",
#     params=["A1", "mu1", "gamma1",
#             "A2", "mu2", "gamma2",
#             "A3", "mu3", "gamma3",
#             "A4", "mu4", "gamma4",
#             "A5", "mu5", "gamma5",
#             "A6", "mu6", "gamma6",
#             "C"],
# )


# # %%

# # symmetric sum of 6 voigt profiles
# def f_sum_2_voigt(x,
#                   A1=1, mu1=0, sigma1=1, gamma1=1,
#                   A2=1, mu2=0, sigma2=1, gamma2=1,
#                   A3=1, mu3=0, sigma3=1, gamma3=1,
#                   A4=1, mu4=0, sigma4=1, gamma4=1,
#                   A5=1, mu5=0, sigma5=1, gamma5=1,
#                   A6=1, mu6=0, sigma6=1, gamma6=1,
#                   C=0):
#     x = np.array(x)
#     return A1 * scipy.special.voigt_profile((x - mu1), sigma1, gamma1)
    + \
#         A2 * scipy.special.voigt_profile((x - mu2), sigma2, gamma2) + \
#         A3 * scipy.special.voigt_profile((x - mu3), sigma3, gamma3) + \
#         A4 * scipy.special.voigt_profile((x - mu4), sigma4, gamma4) + \
#         A5 * scipy.special.voigt_profile((x - mu5), sigma5, gamma5) + \
#         A6 * scipy.special.voigt_profile((x - mu6), sigma6, gamma6) + \
#         C


```

```
# sum_2_voigt = mmf.fit_function(
#     f=f_sum_2_voigt,
#     bounds=([-np.inf, -6, 0.1, 0.1,
#              -np.inf, -4, 0.1, 0.1,
#              -np.inf, -2, 0.1, 0.1,
#              -np.inf, 0, 0.1, 0.1,
#              -np.inf, 2, 0.1, 0.1,
#              -np.inf, 4, 0.1, 0.1,
#              -np.inf],
#             [0, -4, 1, 1,
#              0, -1, 1, 1,
#              0, 0, 1, 1,
#              0, 2, 1, 1,
#              0, 4, 1, 1,
#              0, 6, 1, 1,
#              np.inf]),
#     description="sum of 6 voigt profiles with common offset C",
#     params=["A1", "mu1", "sigma1", "gamma1",
#             "A2", "mu2", "sigma2", "gamma2",
#             "A3", "mu3", "sigma3", "gamma3",
#             "A4", "mu4", "sigma4", "gamma4",
#             "A5", "mu5", "sigma5", "gamma5",
#             "A6", "mu6", "sigma6", "gamma6",
#             "C"],
# )


# %%

data = pd.read_csv("../data/iron/moessbauer_iron_final.txt", header=0, delimiter="\t", decimal=",", names=["velocity", "time", "counts"])
data["time"] = data["time"] / 1000  # time in s

data_underground = pd.read_csv("../data/mexican_hat_background/moessbauer_plexiglass_final.txt", header=0, delimiter="\t", decimal=",", names=["velocity", "time", "counts"])
data_underground["time"] = data_underground["time"] / 1000  # time in s

print("Measurement time [h]:", data["time"].sum() / 3600)
print("Time per point [s]:", data["time"].sum() / len(data["velocity"].unique()))


# %%

data_compton = mmu.read_json("./data_compton.json")
data_compton


# %%

velocities = np.sort(data["velocity"].unique())
cps_raw = []
cps_raw_err = []
```

```
298
299 for velocity in velocities:
300     data_velocity = data[data["velocity"] == velocity]
301     cps_raw.append(data_velocity["counts"].sum() / data_velocity["time"].
    sum())
302     cps_raw_err.append(np.sqrt(data_velocity["counts"].sum()) /
    data_velocity["time"].sum())
303
304 cps_raw, cps_raw_err = mmu.convert_to_array(cps_raw, cps_raw_err)
305
306 velocities_underground = np.sort(data_underground["velocity"].unique())
307 cps_underground_raw = []
308 cps_underground_raw_err = []
309
310 for velocity_underground in velocities_underground:
311     data_underground_velocity = data_underground[data_underground["
    velocity"] == velocity_underground]
312     cps_underground_raw.append(data_underground_velocity["counts"].sum()
    / data_underground_velocity["time"].sum())
313     cps_underground_raw_err.append(np.sqrt(data_underground_velocity["
    counts"].sum()) / data_underground_velocity["time"].sum())
314
315 cps_underground_raw, cps_underground_raw_err = mmu.convert_to_array(
    cps_underground_raw, cps_underground_raw_err)
316
317
318 # %%
319
320 cps = (cps_raw - data_compton["compton_cps"]) / data_compton["
    acrylic_absorption"]
321 cps_err = np.sqrt((1 / data_compton["acrylic_absorption"] * cps_raw_err)
    **2
322                   + (1 / data_compton["acrylic_absorption"] *
    data_compton["compton_cps_err"])**2
323                   + ((cps_raw - data_compton["compton_cps"]) /
    data_compton["acrylic_absorption"]**2 * data_compton["
    acrylic_absorption_err"])**2)
324
325 cps_underground = (cps_underground_raw - data_compton["compton_cps"]) /
    data_compton["acrylic_absorption"]
326 cps_underground_err = np.sqrt((1 / data_compton["acrylic_absorption"] *
    cps_underground_raw_err)**2
327                   + (1 / data_compton["acrylic_absorption"] *
    data_compton["compton_cps_err"])**2
328                   + ((cps_underground_raw - data_compton["compton_cps"])
    / data_compton["acrylic_absorption"]**2 * data_compton["
    acrylic_absorption_err"])**2)
329
330
331 # %%
332
333 fig, ax = mmp.make_fig(grid=True)
334
```

```
335 mmp.plot(ax, velocities, cps, y_err=cps_err, label="Count rate of
    reemission", config="scatter")
336 # mmp.plot(ax, velocities_underground, cps_underground, y_err=
    cps_underground_err,
337 #           label="Count rate of the unexplained underground", config="
    scatter", color="tab:red")
338
339
340 velocities_fit = velocities[35:50]
341 cps_fit = cps[35:50]
342 cps_fit_err = cps_err[35:50]
343 mmp.plot(ax, velocities_fit, cps_fit, y_err=cps_fit_err, color="tab:red",
     config="scatter", label="Data used for the fit")
344
345 mmp.add_to_legend(ax, " ")
346
347 xx = np.linspace(-8, 8, 800)
348 # mmp.plot(ax, xx, sum_2_gauss(xx, -0.2, 0.9, 0.1, 0.3, 0.6))
349 out_2_gauss = mmp.fit(sum_2_gauss,
350                       velocities_fit, cps_fit,
351                       y_err=cps_fit_err,
352                       p0=[-0.1, 0.9, 0.2,
353                           0.3, 0.8],
354                       bounds=True,
355                       print_results=True,
356                       x_range=np.linspace(-8.5, 8.5, 800),
357                       ax=ax,
358                       result_units=["cps", "mm s$^{-1}$", "mm s$^{-1}$",
    "mm s$^{-1}$", "cps"],
359                       color="darkgreen",
360                       kwargs_plot={"linewidth": 2.5},
361                       label="Gaussian model")
362
363 # mmp.plot(ax, xx, sum_2_lorentz(xx, -0.2, 0.9, 0.1, 0.3, 0.7))
364 out_2_lorentz = mmp.fit(sum_2_lorentz,
365                         velocities_fit, cps_fit,
366                         y_err=cps_fit_err,
367                         p0=[-0.2, 0.9, 0.1,
368                             0.3, 0.7],
369                         bounds=True,
370                         print_results=True,
371                         x_range=np.linspace(-8.5, 8.5, 400),
372                         ax=ax,
373                         result_units=["cps", "mm s$^{-1}$", "mm s$^{-1}$"
    , "mm s$^{-1}$", "cps"],
374                         color="purple",
375                         kwargs_plot={"linewidth": 2.5},
376                         label="Lorentzian model")
377
378 out_2_voigt = mmp.fit(sum_2_voigt,
379                       velocities_fit, cps_fit,
380                       y_err=cps_fit_err,
381                       p0=[-0.2, 0.9, 0.1, 0.1,
382                           0.3, 0.7],
```

```
                        bounds=True,
                        print_results=True,
                        x_range=np.linspace(-8.5, 8.5, 400),
                        ax=ax,
                        result_units=["cps", "mm s$^{-1}$", "mm s$^{-1}$",
    "mm s$^{-1}$", "mm s$^{-1}$", "cps"],
                        color="red",
                        kwargs_plot={"linestyle": (0, (1, 3)), "linewidth":
     2.5},
                        label="Voigt model")

out = {"gauss": out_2_gauss,
       "lorentz": out_2_lorentz,
       "voigt": out_2_voigt}

ax.set_title("Reemission spectrum of natural iron")
ax.set_xlabel(r"Velocity $v$ [mm s$^{-1}$]")
ax.set_ylabel(r"Count rate [cps]")

ax.set_xlim(-8.5, 8.5)
mmp.legend(ax, loc=1, ncols=2)
ax.set_ylim(0.18, 0.88)

# mmp.legend(ax, loc=1)

if save_images:
    mmp.save_fig(fig, path="../report/figures", name="iron", extension="
    pdf")


# %%

def energy_from_speed(E, E_err, v, v_err):
    """
    E in eV, v in m/s
    [Delta E, Delta E err] output in eV
    """

    c = scipy.constants.c
    return np.array([E * v / c, np.sqrt((E / c * v_err)**2 + (v / c *
    E_err)**2)])


e_charge = scipy.constants.e
E_iron = 14.41295e3  # eV
E_iron_err = 0.00031e3  # eV
for profile in ["gauss", "lorentz", "voigt"]:
    # print(out[profile][0][1], out[profile][1][1])
    print(f"Isomer shift calculated with {profile} profile:")
    print(f"{energy_from_speed(E_iron, E_iron_err, out[profile][0][-2]*1e
    -3, out[profile][1][-2]*1e-3)*1e9} neV")


# %%
```

```
432
433 # lit vals
434 mu_n = 3.15245e-8  # eV / T
435 mu_e = -0.1549 * mu_n
436 mu_e_err = 0.0002 * mu_n
437 mu_g = 0.09044 * mu_n
438 mu_g_err = 0.00007 * mu_n
439 e_iron = 14.41295 * 1e3  # eV
440 e_iron_err = 0.00031 * 1e3  # eV
441
442 speed_gauss = out_2_gauss[0][1] * 1e-3  # this is in m/s. the data was in
        mm/s.
443 speed_gauss_err = out_2_gauss[1][1] * 1e-3
444 speed_lorentz = out_2_lorentz[0][1] * 1e-3  # this is in m/s. the data
        was in mm/s.
445 speed_lorentz_err = out_2_lorentz[1][1] * 1e-3
446 speed_voigt = out_2_voigt[0][1] * 1e-3  # this is in m/s. the data was in
        mm/s.
447 speed_voigt_err = out_2_voigt[1][1] * 1e-3
448 speed_iso_gauss = out_2_gauss[0][3] * 1e-3  # this is in m/s. the data
        was in mm/s.
449 speed_iso_gauss_err = out_2_gauss[1][3] * 1e-3
450
451 # light = scipy.constants.c
452
453 # fit results: Delta E_alpha (distance isomer to first peak) for the
        different
454 # fits in eV:
455 E_gauss = speed_gauss * e_iron / scipy.constants.c
456 E_gauss_err = np.sqrt(
457     (e_iron / scipy.constants.c)**2 * speed_gauss_err**2
458     + (speed_gauss / scipy.constants.c)**2 * e_iron_err**2
459 )
460 E_lorentz = speed_lorentz * e_iron / scipy.constants.c
461 E_lorentz_err = np.sqrt(
462     (e_iron / scipy.constants.c)**2 * speed_lorentz_err**2
463     + (speed_lorentz / scipy.constants.c)**2 * e_iron_err**2
464 )
465 E_voigt = speed_voigt * e_iron / scipy.constants.c
466 E_voigt_err = np.sqrt(
467     (e_iron / scipy.constants.c)**2 * speed_voigt_err**2
468     + (speed_voigt / scipy.constants.c)**2 * e_iron_err**2
469 )
470
471 # isomer shift
472 E_iso_gauss = speed_iso_gauss * e_iron / scipy.constants.c
473 E_iso_gauss_err = np.sqrt(
474     (e_iron / scipy.constants.c)**2 * speed_iso_gauss_err**2
475     + (speed_iso_gauss / scipy.constants.c)**2 * e_iron_err**2
476 )
477 # print(out_2_gauss[0])
478
479 # print(E_gauss, "pm", E_gauss_err)
480 # print(E_lorentz, "pm", E_lorentz_err)
```

```
481 # print(E_voigt, "pm", E_voigt_err)
482
483
484 B_gauss = E_gauss / (1 / 3 * mu_e + mu_g)
485 B_gauss_err = np.sqrt(
486     E_gauss_err**2 / (1 / 3 * mu_e + mu_g)**2 +
487     1 / 9 * E_gauss**2 * mu_e_err**2 / (1 / 3 * mu_e + mu_g)**4
488  +
489     E_gauss**2 * mu_g_err**2 /  (1 / 3 * mu_e + mu_g)**4
490 )
491 B_lorentz = E_lorentz / (1 / 3 * mu_e + mu_g)
492 B_lorentz_err = np.sqrt(
493     E_lorentz_err**2 / (1 / 3 * mu_e + mu_g)**2
494  +
495     1 / 9 * E_lorentz**2 * mu_e_err**2 / (1 / 3 * mu_e + mu_g)**4
496  +
497     E_lorentz**2 * mu_g_err**2 /  (1 / 3 * mu_e + mu_g)**4
498 )
499 B_voigt = E_voigt / (1 / 3 * mu_e + mu_g)
500 B_voigt_err = np.sqrt(
501     E_voigt_err**2 / (1 / 3 * mu_e + mu_g)**2
502  +
503     1 / 9 * E_voigt**2 * mu_e_err**2 / (1 / 3 * mu_e + mu_g)**4
504  +
505     E_voigt**2 * mu_g_err**2 /  (1 / 3 * mu_e + mu_g)**4
506 )
507
508 print("gauss: ", B_gauss, B_gauss_err)
509 print("lorentz: ", B_lorentz, B_lorentz_err)
510 print("voigt: ", B_voigt, B_voigt_err)
```