Contents

1 Theoretical Background

In this section we will introduce all the theoretical background needed to perform the experiment and gain some intuition about the underlying physics and experimental aspects of Mößbauer spectroscopy.

1.1 The Doppler Effect

The ability to adjust the radiated energy emitted by the source, to match the energy levels of the absorber lies at the heart of resonant spectroscopy. In this experiment we use a radioactive source, which emitts photons with the energy $E_\gamma = 14.4\,\text{keV}$. To measure the absorbtion spectrum of the absorber material (Iron and stainless steel) the absorber is moved relatively to the source, such that the radiation emmited by the source gets Doppler shifted for the moving absorber. The Doppler effect shifts the frequency of electromagnetic radiation by

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

where v is the relative speed with which the receiver moves away from the source and c is the speed of light. Expanding eq. (1) to first order in the relative speed one obtains

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

Now using the photon energy $E_\gamma = h\,f$ one obtains

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

where Plancks constant h cancels on both sides. Note however, that v is positive if the receiver moves away from the source and v is negative, when it moves towards the source. So concluding the energy of the photon emitted by the source is shifted by $\Delta E = \frac{v}{c} E_0$ and the sign depends on the direction of propagation of the receiver.

1.2 Gamma Radiation

In this subsection we briefly introduce the mechanisms which lead to gamma radiation in this experiment. First and foremost gamma radiation is just high energetic electromagnetic radiation which arises from the radioactive decay of excited atomic nuclei. The photons, which are used in this experiment, come from a radioactive cobalt source, which decays via the mechanisms shown in fig. 1. Other then the used gamma decay there are also two more gamma decays which contribute in this experiment as background, due to Compton scattering, as discussed in section 1.3.

1.3 Interaction of Electromagnetic Radiation and Matter

For every experiment it is necessary to understand the background processes that contaminate the data. Since in this experiment electromagnetic radiation is used to investigate the energetic

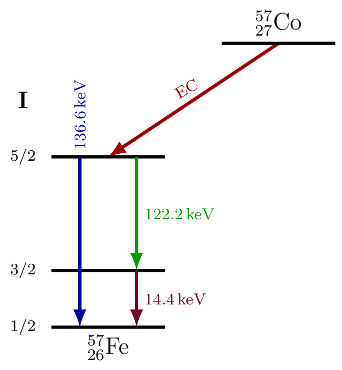


Figure 1: In this figure a simplified decay scheme of ^{57}Co is shown. The cobalt decays via electron capture to an excited state of ^{57}Fe. This exited state decays to the ground state via the radiation of high energy photons.

properties of two materials it is important to understand how this radiation interacts with matter. The three main mechanisms are the photoelectric effect, Compton scattering and pair production. The absorption coefficient is plotted against the photon energy in fig. 2 to illustrate the contribution of the different mechanisms to the total absorption, in dependence of the photon energy.

Photoelectric Effect The photoelectric effect describes the extraction of an electron out of its bound state by a photon. The resulting free electrons are called photoelectrons. The energy of a photoelectron is given by $E_\mathrm{e} = E_\gamma - E_\mathrm{B}$, where E_B is the binding energy of the electron in its orbit. This holds since the photon in in the photoelectric effect get fully absorbed by the shell electron. This is also the reason why this process does not contribute to the underground in this experiment, it only reduces the signal.

Pair Production Pair production describes the scattering event $\gamma + Z \to Z + \mathrm{e}^+ + \mathrm{e}^-$, where Z represents an atom with atomic number Z. This process is due to energy conservation only possible for photons with an energy of $E_\gamma \geq 2 \cdot m_\mathrm{e}c^2 = 1.022\,\mathrm{MeV}c^2$. Above this energy the pair production mechanism gets more and more important as seen in fig. 2. Within the energy regime of the radiation used in this experiment this effect has no influence at all.

Compton Scattering The most dominant background for this experiment is caused by Compton scattering of high energy photons. Which is caused due to the fact that the photon is not absorbed instantly but scattered while changing the frequency. As seen in fig. 1, there are two decay modes in which high energy photons with $E_\gamma > 14.4\,\mathrm{keV}$ get produced. The frequency of these photons can be lowered to $14.4\,\mathrm{keV}$ by repeated Compton scattering.

Attenuation of Gamma Radiation The absorbers used in this experiment are covered with acrylic glass. Therefore, it is of interest to know how much of the radiation is absorbed in these

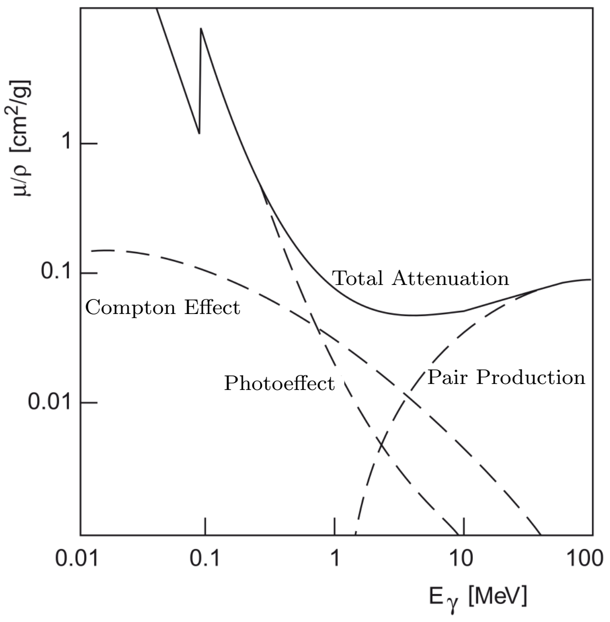


Figure 2: In this picture the photon absorption coefficient μ/ρ is plotted against the photon energy. The three main interaction mechanisms between light and matter are displayed – Photoelectric Effect, Compton Effect and Pair Production. The picture is taken from the textbook of B.Povh [11]

layers of acrylic glass. This attenuation can be quantified by the exponential attenuation law

$$T := \frac{I}{I_0} = \exp\left(-\frac{\mu}{\rho}\cdot\rho d\right), \tag{4}$$

where I is the intensity, I_0 is the incident intensity, ρ is the density of the absorber material and d the thickness of the absorber.

1.4 The Mößbauer Effect

The central phenomenon of this experiment is the Mößbauer effect, which describes the "recoilless emission"[1] of gamma rays from a nucleus confined in a crystal lattice. It is somewhat instructive to first discuss a classical two body decay. So lets assume a resting nucleus of mass m, which is in an energy state with energy E_2, decays to a lower energy state with energy E_1 by emission of a photon with frequency f. The conservation of momentum and energy reads

$$\begin{aligned} 0 &= \frac{hf}{c} + mv, \\ E_2 &= E_1 + hf + \frac{mv^2}{2}. \end{aligned}$$

[1]The term "recoilless" is somewhat misleading, since the crystal as a whole is always recoiling after the emission of a photon. In the literature the Mößbauer effect is also called "photon emission without transfer of energy to internal degrees of freedom of the lattice" [3]

Solving this set of equations for the frequency of the photon one yields

$$hf = (E_2 - E_1) - \frac{h^2 v^2}{2mc^2} =: E_0 - R,$$

where R is the recoil energy of the nucleus. Therefore the photon energy is $E_\gamma < E_0$, because some of the energy from the decay gets absorbed as recoil of the nucleus. If however the excited nucleus is bound within a crystal lattice, there is a non-vanishing probability[2] for the emission of a photon without excitation of a vibrational mode of the crystal lattice (phonon) which would absorb a part of the decay energy E_0. The recoil energy of the whole lattice can be neglected, since $m_{\text{Nucleus}} \ll M_{\text{Lattice}}$. So the defining property which needs to be known to compute the probability for the emission of a photon with $E_\gamma \approx E_0$ is the phonon spectrum of the lattice, to which the nucleus is confined. There are two simple models, which make a prediction about the phonon spectrum. These models are introduced very briefly in the following paragraphs.

Einstein Model In the Einstein model each atom is assumed to be bound in an harmonic potential. Furthermore the harmonic potentials of all nuclei has the same natural frequency ω_{E} (also called eigenfrequency), therefore the atoms vibrate independently which is a strong physical restriction which gets relaxed in the Debye model. The Einstein model allows the description of the heat capacity of the crystal at high temperatures. For a complete discussion of the model see e.g. H. Meyers, Introductory Solid State Physics [8].

Debye Model The Debye model assumes the frequency spectrum to be that of an elastic continuum. It therefore describes atomic vibrations as phonons in a box. In contrast to a free photon gas confined to a box, there is a maximum frequency for phonons since there is only a finite amount of vibrating atoms in the lattice. The Debye model allows the description of the heat capacity at the low temperature and high temperature limits, but deviated from the observed behavior for intermediate temperatures.

1.4.1 Debye-Waller Factor

To quantify the incidence of "recoilless" nuclear transitions the fraction of "recoilless" emissions of photons can be used. This fraction is called the Debye-Waller factor f. The Debye-Waller factor of the source can be computed via

$$f_S = \frac{\dot{N}(\infty) - \dot{N}(0)}{\dot{N}(\infty)} \left(1 - \exp\left(-\frac{1}{2}T_A\right) J_0\left(\frac{1}{2} i T_A\right)\right)^{-1}, \tag{5}$$

where

$$J_0(x) = \frac{1}{\pi} \int_0^\pi \cos(x \sin(\tau))\, \mathrm{d}\tau \tag{6}$$

and T_A is the effective absorber thickness. A detailed discussion about these expressions is found in [6].

[2]The Mößbauer effect is of purely quantum-mechanical nature. For a self-contained discussion based on basic quantum mechanical principles we recommend [3]

1.5 Line Width of Spectral Lines

The natural line width of emission/absorption lines is related to the life-time of the state by

$$\Gamma_{\text{nat.}} = \frac{\hbar}{\tau}, \tag{7}$$

wheren $\gamma_{\text{nat.}}$ is the full width at half maximum of the absorption/emission peak, τ is the lifetime of the state and $\hbar$ is the reduced Planck constant. This can be heuristically justified by the energy-time uncertainty

$$\Delta E \Delta t \geq \hbar. \tag{8}$$

1.6 Isomer Shift

The energy levels of an atomic nucleus are dependent on exterior electric and magnetic fields like the ones generated by the electron shell of the atom. If for example the atom is confined in a lattice, the electron density of the atom may change a little and thus also the energy level of the state of the nucleus. If now source and absorber are slightly different in their chemical composition of the lattice, the energy states of the nuclei in the absorber and source are shifted differently with respect to each other. Therefore, the isomer shift is also called chemical shift. This effect will be observed by a slight shift of the resonances from the expected positions.

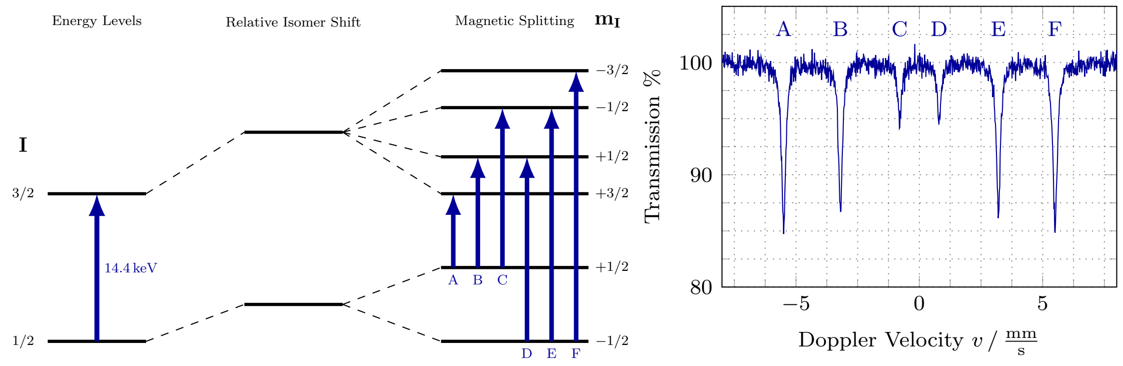


Figure 3: Energy Level diagram and expected Mößbauer spectrum illustrating the magnetic level-splitting in ^{57}Fe nuclei states. Note that the displayed data is not actual measurement data but generated to sketch the signal we expect to see.

1.7 Hyperfine Splitting of Nuclei States

In absence of any exterior fields a nucleus can be described by fermionic particles confined in an effective spherical symmetric single particle potential, like a Wood-Saxon potential. Such an Hamiltonian has degenerate eigenenergies. If now a magnetic field is applied, like the one generated by the electron shell, the angular degeneracy is lifted. This lifting of degeneracy is closely related to the fact that the spherical symmetry of the Hamiltonian is broken, since there is now a preferred direction for the angular momentum given by the quantum number I. So the states which where "$2I + 1$"-fold degenerate are now splitted due to their different magnetic quantum number m_I, which is the z-axis projection of the angular momentum. The energy shift is given by

$$E_{\mathrm{HF}} = \frac{\mu m_I B}{I},$$

where μ is the magnetic moment m_I is the magnetic quntum number assigned to the state, B is the absolute value of the exterior magnetic field and I is the spin quantum number. So the photons which would excite a state specified by (E_1, I_1, m_{I_1}) to a state specified by (E_1, I_1, m_{I_1}) needs to have the energy

$$E_\gamma = (E_2 - E_1) + E_{\mathrm{Iso.}} - \left(\frac{\mu m_{I_2}}{I_2} - \frac{\mu_{I_1}}{I_1} \right) B, \tag{9}$$

where E_1, E_2 are the energies of the degenerate I_1, I_2 states and $E_{\mathrm{Iso.}}$ is the energy of the relative isomer shift between source and absorber. To decide which transitions are possible one finds for dipole order in perturbation theory the selection rules: $\Delta I = \pm 1$ and $\Delta m_I = 0, \pm 1$. The transitions of interest and the expected spectrum in this experiment are displayed in fig. 3.

2 Experimental Setup

In this section we will introduce the setup used in this experiment. The main setup is shown in fig. 4, it consists of a high precision engine, which is able to run at very precise velocities. This engine moves a mount on which the different absorbers can be placed. In this experiment we used an iron and a stainless steal absorber. The ^{57}Co source is fixed in the setup and cannot be moved. Additionally there is a special mount for additional sources, which are used for calibration. At the other end of the setup there is a scintillator detector to detect the gamma radiation. We will now briefly discuss the functionality of a scintillator.

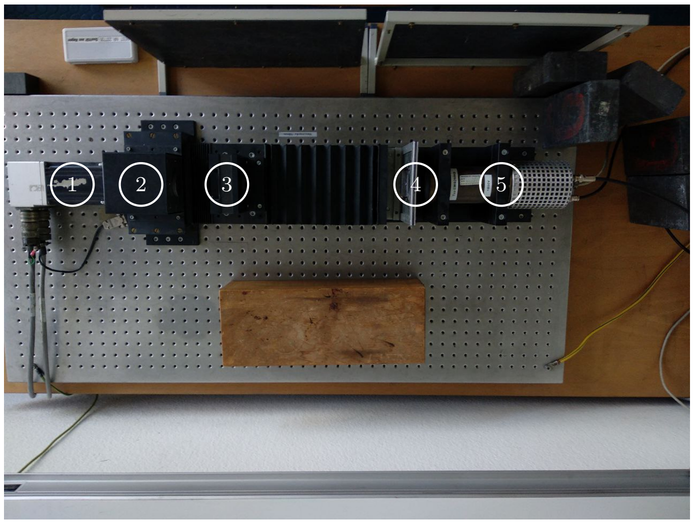


Figure 4: In this figure the actual setup used in this experiment is shown. (1) High precision engine to move the absorber, (2) ^{57}Fe Source, (3) Mount for radio active sources for calibration and the absorber, (4) Mount for Aluminium/Plexiglas shielding, (5) Scintillator detector.

2.1 Scintillator

In this experiment one scintillator is used to detect high energy photons, so called gamma radiation. There are two different types of scintillating materials, organic and inorganic ones. Inorganic scintillators consist of doped crystals. When ionizing radiation hits the scintillator an electron is excited from the valence band into the conduction band. When this excited electron decays back into the valence band it emits photon. Since the crystal is doped, the photon will not be reabsorbed by the scintillator and can leave the material and then be detected. In organic scintillators the detectable photons are produced when molecular states are excited and decay again, usually these photons are in the ultra violet frequency range.

Photomultiplier In the scintillator the high energy photons get absorbed and produce a number of secondary photons with lower energy which can be detected. Therefore a photomultiplier is used.

The photons produced in the scintillator get directed towards a photo cathode by using a light guide. There one or more electrons gets extracted via the photoelectric effect. These electrons are then accelerated towards a dynode where secondary electrons are released. This is iterated multiple time generating more and more electrons until a measurable signal is produced. This process is sketched in fig. 5

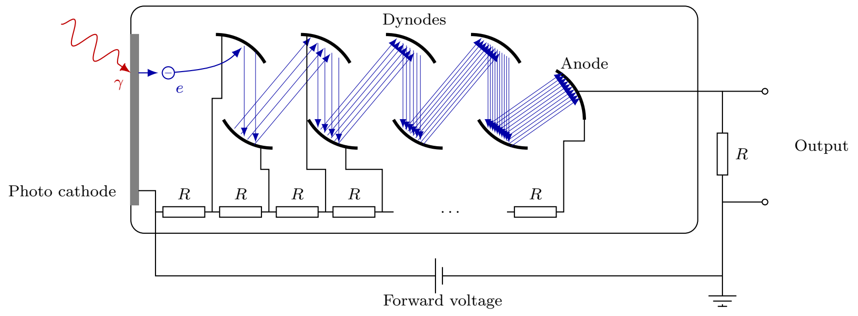


Figure 5: In this figure the functionality of a photomultiplier is sketched.

2.2 NIM Modules

In this experiment several nuclear instrumentation standard (NIM) modules are used for signal processing. As shown in fig. 7 In this experiment we used a multi channel analyzer (MCA), a single channel analyzer (SCA) and a linear gate. The single channel analyzer was used as a discriminator. We set an upper and a lower boundary on the signal, such that the SCA only returns a positive signal if a photon within the set window is detected (The goal is to filter for $E_\gamma \approx 14.4\,\text{keV}$ photons.). This logical signal is put into the linear gate along with the amplified signal from the photomultiplier and then fed into the MCA if the SCA and the amplifier give a signal simultaneously. The wiring is displayed in fig. 6.

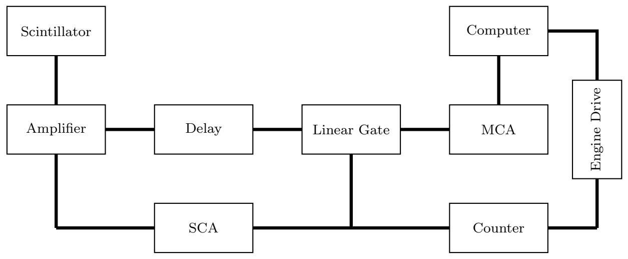


Figure 6: In this figure a schematic setup of the NIM modules is shown to illustrate how we connected the individual modules.

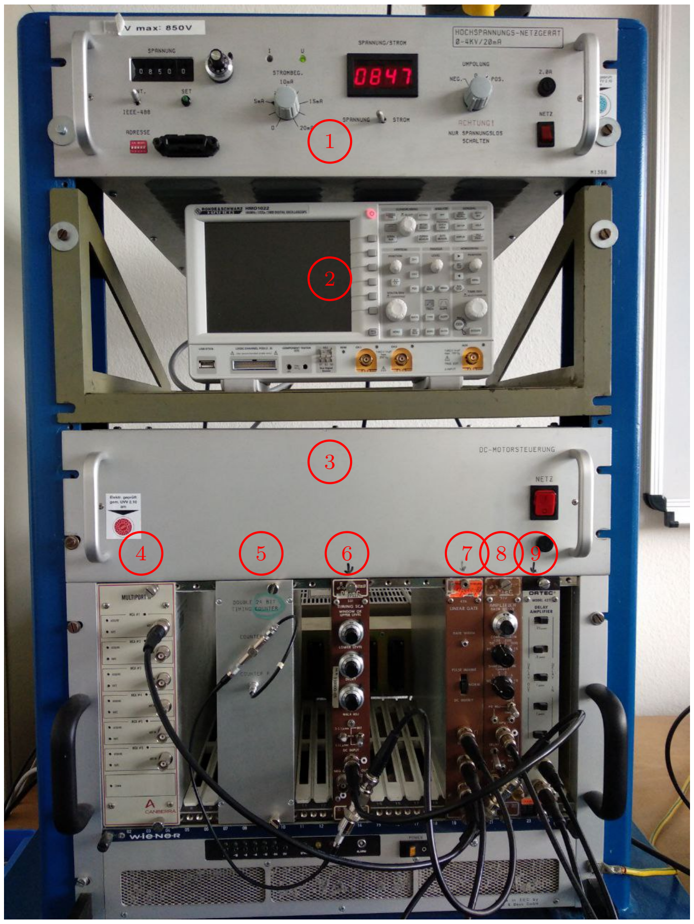
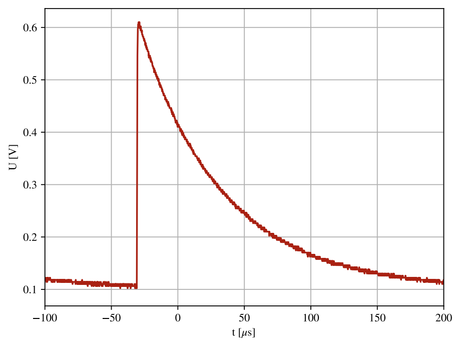


Figure 7: In this figure the electronic devices used in this experiment are shown. (1) High voltage supplier for the photomultiplier, (2) Oscilloscope to show the signals of the NIM devices, (3) Voltage supply for the engine, (4) Multi channel analyzer (MCA), (5) Counter, (6) Timing single channel analyzer (SCA), (7) Linear gate, (8) Amplifier, (9) Delay.

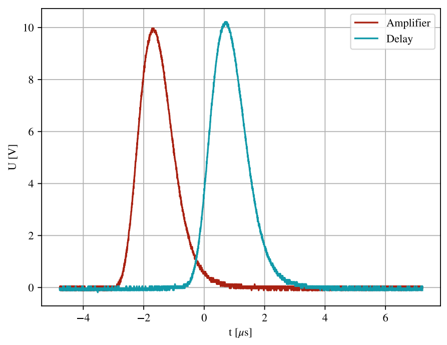
3 Procedure

The goal of the experiment is the measurement of absorption spectra of stainless steel and iron. This is done by moving the absorber in front of a 14.4 keV source, so we can make use of the Doppler effect. As the source does not only emit photons of the right energy the setup has to be calibrated and set to a proper energy window.

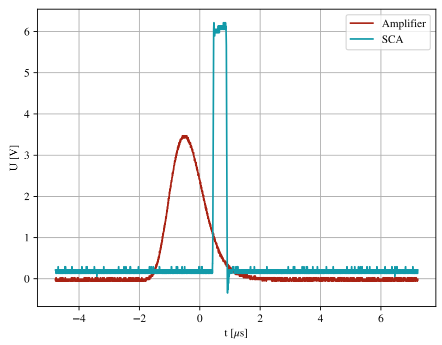
Analysis of the Setup The absorption spectrum is measured by a scintillator and passed through different components to create a computer readable signal with proper settings. While connecting the components following the scheme shown in fig. 6 an oscilloscope was used to check the signal after every step.



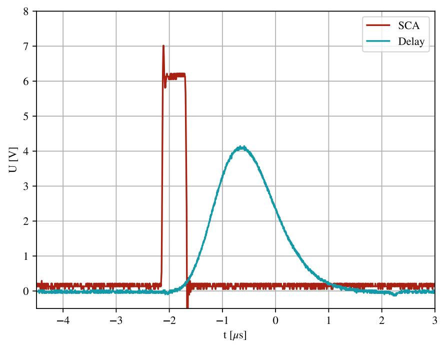
(a) Signal of the preamplifier located behind the photomultiplier.



(b) Amplified signal in comparison to the delayed signal.



(c) The output of the amplifier triggers a logical yes in the SCA.



(d) The delayed signal that passed the linear gate that was opened due to the logical yes of the SCA.

Figure 8: Path of the signal through the used setup.

In fig. 8 the signal after the steps is shown. The exponential decay in the preamplifier originates from the discharge of a capacitor. Afterwards the signal gets amplified and shaped to a Gaussian and is splitted into two signals. On signal is delayed with a delay unit and we can see, that the signals are of the same shape but shown an offset. The other output of the amplifier is lead to a SCA and triggers a logical yes. The signal of the SCA is connected to a linear gate, which opens for a given logical yes and lets the delayed amplifier signal pass.

Calibration of the MCA The signal that passes the described signal is lead to a MCA and the different energies are assigned to bins. As for the later measurements we are only interested in detecting photons with 14.4 keV we need to be able to determine the bin that corresponds to that energy. This is done by calibrating the MCA with help of five known spectra lines that were measured, so we could determine the bin in the MCA that corresponds to the known energy of the measured spectrum.

This calibration measurement was done twice. First with a small Corse Gain which gives a good overview of the spectra and a second one with a higher Corse Gain, that was used for the later measurements.

With help of these calibrations the SCA-discriminator window was set so that only photons in the desired energy range pass, we found an upper bound of 2.10 and a lower bound of 1.10 for a Corse Gain of 100.

Background To determine the count rate that originates by Compton scattering of the absorber the count rate for different thicknesses of aluminium were measured. The sledge with the absorber was at rest during this measurement and the stainless steel absorber was used, as for the later calculations this is the probe for which the absolute count rate is of great interest. Nevertheless this can also be used for the iron absorber as the background should be the same.

Furthermore the attenuation of gamma radiation by acrylic glass was determined by measuring the count rate without an absorber, but acrylic glass inserted in the thickness of the acrylic glass that wraps the absorber.

Mößbauer Spectroscopy After the calibration and the background measurements the spectra of stainless steel and iron were measured. The absorber was placed on the sledge and a `LabView` software was used to set velocity ranges and start and stop the measurement. The measurement times, as well as the clustering density of the velocities were chosen as a compromise between accuracy and time.

4 Analysis

4.1 Energy Calibration

For the measurements of the iron and the stainless steel spectra a sensitive energy window needs to be determined. In this analysis the calibration with a Corse Gain set to 100 is shown, but it was only used for error estimation on the chosen window as it was measured in the end of the experiment. On the first day of the experiment a calibration for a Corse Gain of 20 was recorded and the window of the experiment was set by eye with help of that calibration.

To find a window the Multi Channel Analyser is calibrated, so the position of the 14.4 keV peak can be determined. For this calibration five different probes were measured and the channel that corresponds to the K_α decay peak with the energies listed in the instruction [1] was determine by fitting a Gaussian function of the form

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

A is the amplitude, μ the position of the maximum, σ the variance and C the offset. The fits are shown in fig. 15 and the fit parameters that lead to the linear regression can be found in table 3.

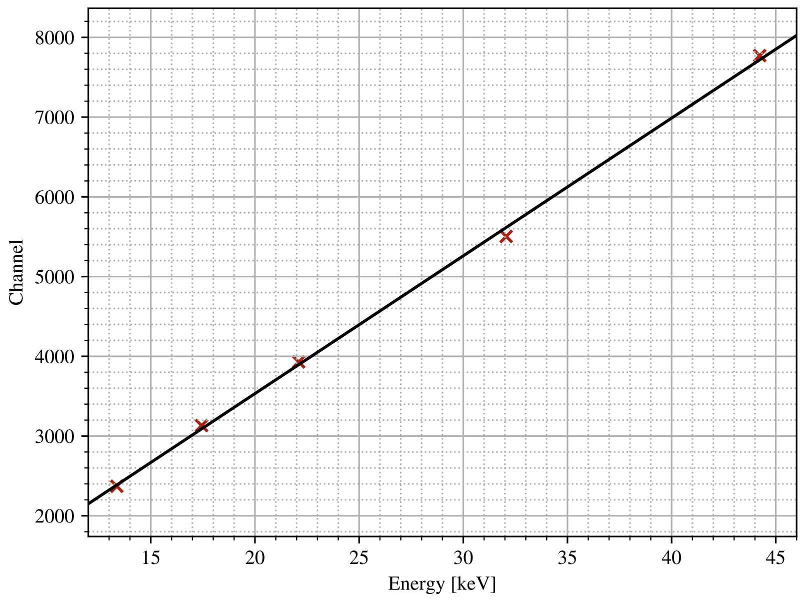


Figure 9: Linear regression to determine the relation between channel and energy. The shown points correspond to the five different probes that were used for calibration (see fig. 15)

The found positions were plotted against the corresponding energy and a linear regression was performed which yields

$$f(E) = (173 \pm 3)\,\text{Channel/keV} \cdot E - (77 \pm 89)\,\text{Channel}. \tag{10}$$

The fit looks as expected, the offset normally should be zero but this is fulfilled within the uncertainty which is rather big due to the very few data points. This fit can now be used to determine the channel for 14.4 keV would fill and we find $\text{Channel}_{14.4} \approx 2414 \pm 46$.

Anyways this calculated channel was not used to determine the window, as the amplification was changed after the first calibration and it was not certain if there would be enough time for a second calibration. To determine the window the K_α-line of Rubidium was used. With help of the first calibration it was possible to determine the position of that peak and it was possible to follow that peak to the higher Corse Gain. Therefore for the new amplification a window was chosen with regard to that peak, as it is rather close to the Cobalt peak we later measure. Afterwards the Cobalt-spectrum for this window was looked at and the window was adjusted a little. This results in the peak that is shown in fig. 10. Later, after the second calibration was performed the energy the peak corresponds to was calculated with the inverse of eq. (10) and we found $E_{\text{Co}} = (13.99 \pm 0.02)\,\text{keV}$, so the window was chosen around the right peak, but we note that the window is much wider than it needs to be to satisfy the uncertainty of the calibration, so this might lead to higher noise in the later measurements. Nevertheless it is also not recommended to choose a really narrow window, as this might cut of too much data and yield bad statistics.

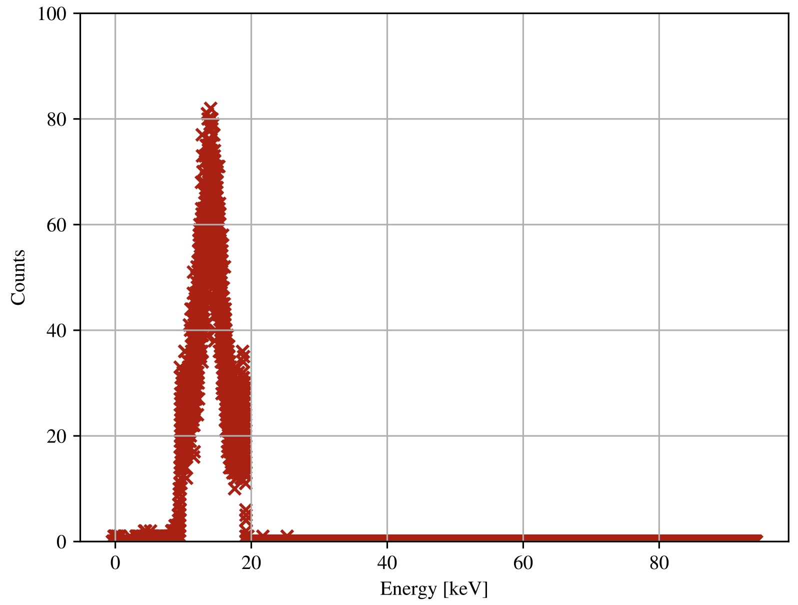


Figure 10: Measured Cobalt spectrum after the window was set. The energies are calculated with help of the second calibration.

4.2 Compton Background

With the set window of the SCA the Compton background was measured. The window just counts photons with an energy around the searched 14.4 keV, but due to the Compton effect the higher Cobalt energies 136 keV and 122 keV can be decreased and measured in the window. As these photons do not correspond to the absorption that should be measured this background needs to be determine and subtracted. The background was determined by measuring the count rates for different thicknesses of aluminium in front of the scintillator for the stainless steel absorber. The stainless steel absorber was chosen as the absolute count rate is of interest for determining the Debye-Waller factor which is done for stainless steel and not of great interest for the iron absorber as for that case we are only interested in the position of the peaks. The aluminium shielding will absorb the 14.4 keV photons and for a thickness of about 1 mm around 90 % of the photons are absorbed. The higher energy photons are not effected this much by the aluminium and around 95 % of the photons can pass a thickness of 1 mm. Due to the different reactions on increasing aluminium thickness a double exponential function can be observed when plotting the count rated against the aluminium thickness.

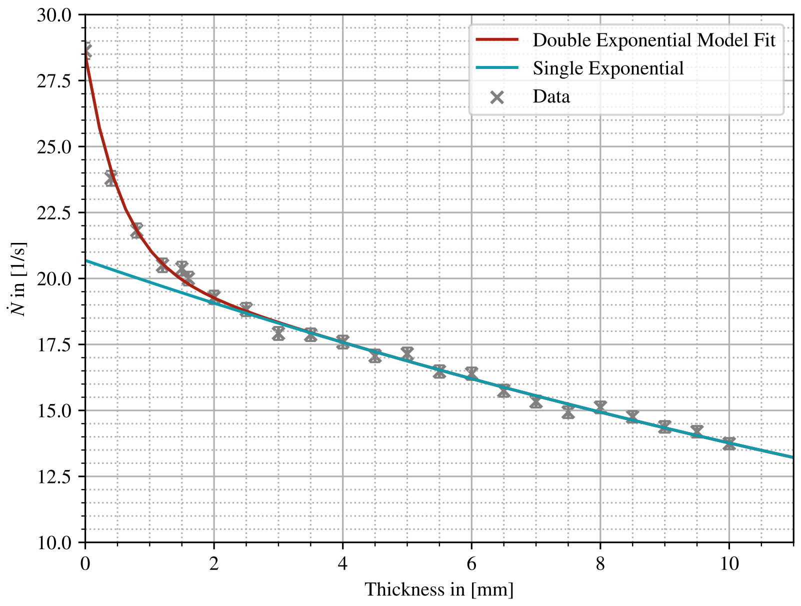


Figure 11: Measurement of the count rate for different thicknesses of aluminium and the stainless steel absorber. The fit of a double exponential function is shown, as well as the slower decaying part that is extrapolated to zero. The count rate of the slower part at zero corresponds to the Compton offset.

As the data taken by the MCP are absolute counts the count rate is determine by

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

where for the error on the count rate a normal Poisson error was applied. The taken data is shown in fig. 11 and a double exponential was fitted to the data

$$\dot{N}(d) = A\exp(-\mu d) + B\exp(-\nu d).$$

The slower decaying part of this double exponential corresponds to the Compton part and is shown separately and extrapolated to an absorber thickness of zero. With the extrapolation to zero the count rate of the Compton background can be determine to

$$\dot{N}_{\mathrm{Compton}} = B = (20.68 \pm 0.17)\,\mathrm{Counts/s}.$$

4.3 Acrylic Glass

Both absorbers are fixed between two acrylic glass plates that also absorb gamma radiation. Thus the absolute count rate is reduced, so to determine the Debye-Waller factor this dampening needs to be determined, so the measurement can be cleared by this background. This was done by measuring the count rate with two acrylic glass plates, but without an absorber, as well as a clean measurement of the count rate without anything inserted. The absorption in acrylic glass follows the Beer-Lambert law and is depended on the thickness of the glass $d = (2.00 \pm 0.02)\,\mathrm{mm}$, the density $\rho_{\mathrm{a.g.}} = 1.19\,\frac{\mathrm{g}}{\mathrm{cm}^3}$ and the attenuation coefficient $\mu/\rho(E = 14.4\,\mathrm{keV}) \approx 1.101\,\frac{\mathrm{cm}^2}{\mathrm{g}}$ [1] that has been extrapolated by the given data. The effect of this dampening can be calculated using these material constants by

$$\dot{N}(d) = \dot{N}_0 \exp\left(-\frac{\mu}{\rho}\rho_{\mathrm{a.g.}} d\right) = \dot{N}_0 R(d).$$

$R(d)$ is the dampening factor that can be calculated analytically with the given formula as well as with the measured count rates by

$$R_{\mathrm{exp.}} = \frac{\dot{N}(d)}{\dot{N}_0},$$

where N_0 is the count rate without absorber of acrylic glass. The experimental measurement gives $R_{\mathrm{exp.}} = 0.837 \pm 0.007$ and the analytical calculation gives $R_{\mathrm{ana.}} = 0.77 \pm 0.04$, where both errors were calculated with Gaussian error propagation. The values do not completely match but are of the same magnitude, for the later calculations the experimental dampening factor is used as this is expected to match the experimental behaviour and all uncertainties are well defined and not dependend from extrapolating some given data to the needed energy.

4.4 Measurements of Stainless Steel

The measurements of the stainless steel absorber were done in a velocity range from circa $-2\,\frac{\text{mm}}{\text{s}}$ to $2\,\frac{\text{mm}}{\text{s}}$ and with varying measurement times. For each measurement the count rate $\dot{N}_{\text{meas.}}$ was calculated with its error, again a Poisson error was used. Afterwards the count rate was cleared by the Compton background and the absorption due to the acrylic glass. First the rate is cleared by the Compton background so only the rate created by the photons of interest, so the 14.4 keV photons, remain. Afterwards this new rate is weighed with the damping factor due to the acrylic glass, as this factor has just been measured or calculated for this energy. Thus the new cleared count rate is given by

$$\dot{N} = \frac{\dot{N}_{\text{meas.}} - \dot{N}_{\text{Compton}}}{R_{\text{exp.}}}. \tag{11}$$

The error on this was propagated with Gaussian error propagation and the measured count rate without its background is shown in the plot fig. 12.

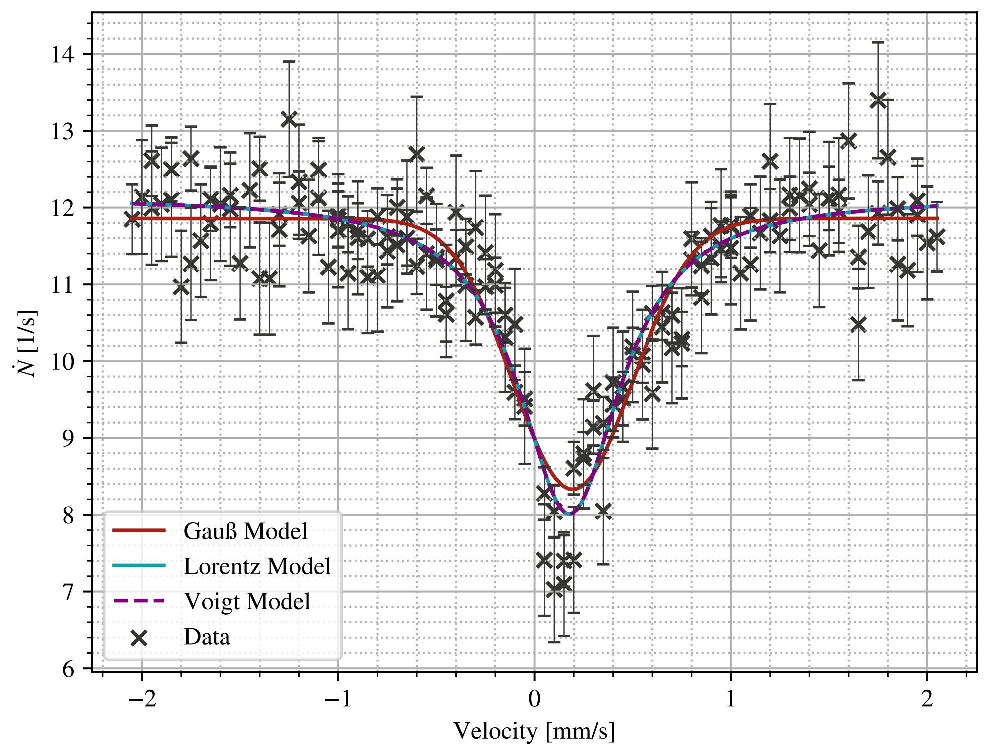


Figure 12: Absorption spectrum of stainless steel. Three different models were fitted to the data.

The measured rate (see fig. 12) shows the expected behaviour. We can see one dip in the count rate that is roughly located around zero and has the shape of a Lorentzian or a Gauss function. For an absorption spectrum we would expect a negative Lorentzian, so a function of the form

$$\dot{N}_{\mathrm{L}}(v) = \frac{A\gamma}{\pi}\frac{1}{(v-v_0)^2+\gamma^2} + C$$

is fitted to the data. Here A is the amplitude (here expected to be negative), γ gives the width of the peak, v_0 describes the position of the peak and C is the offset.

Statistical effects in the measurements like changes in the temperature, a varying sledge velocity or variations in the detection follow a Gauss distribution. This can be used to check whether the fluctuation has a higher effect on the measurement than the absorption we actually want to measure by also fitting a Gauss function and checking whether this function satisfies the data more than the expected Lorentzian. The fit function is given by

$$\dot{N}_{\mathrm{G}} = \frac{A}{\sqrt{2\pi}\sigma}\exp\left(\frac{(v-v_0)^2}{2\sigma^2}\right) + C.$$

here σ corresponds to the standard deviation and the other parameters fulfill the same purpose as in the Lorentzian.

Last a combination of these two things can be taken into account. This makes sense, as the statistical effects effect every point in the measurement and even though they might not be high enough to completely disturbed the measurement it might be nice to take it into account in the fit function. This combination of both attributes is given by the convolution of a Gauss and a Lorentzian and called a Voigt profile and given by

$$\dot{N}_{\mathrm{V}} = \int_{-\infty}^{\infty} G(v';\sigma)L(v-v';\gamma)\mathrm{d}v'.$$

As this Voigt profile does not have a nice analytical expression the function `voigt_profile` from the Python package `scipy.special` was used.

Table 1: Fit parameters for all three fit functions for the measurement of stainless steel.

	Gauss fit	Lorentz fit	Voigt profile
v_0[mm / s]	0.194 ± 0.012	0.177 ± 0.010	0.177 ± 0.010
σ[mm / s]	0.303 ± 0.013		0 ± 53
γ[mm / s]		0.319 ± 0.019	0.32 ± 0.03
A[1 / s]	-3.5 ± 0.6	-4.1 ± 0.3	-4.1 ± 0.3
C[1 / s]	11.86 ± 0.06	12.14 ± 0.07	12.14 ± 0.08

The fit parameters are shown in table 1 and a visualisation of the fits is given in fig. 12. We can see that all the fitted functions describe the data quite well, but the Lorentzian model achieves a better description of the depth of the data. To find a description for the fit quality the reduced χ^2 was calculated and we find $\chi^2_{\mathrm{G}} = 0.94$ and $\chi^2_{\mathrm{L}} = \chi^2_{\mathrm{V}} = 0.82$. For all fits this value is quite low, this is most probably due to the rather high errors. Nevertheless the agreement between data and fit seems okay but to decide for one or another fit a hypothesis test with one model against the other would need to be performed. Furthermore data with lower errors, so longer measurement times is desired.

Conversion from Velocity to Energy The Mößbauer effect can be used to determine small shifts in energies. To do so the measured velocities have to be connected to energies and this is done with help of the Doppler shift and we find

$$E(v) = E_0 \frac{v}{c}, \tag{12}$$

where E_0 is the initial photon energy $E_0 = 14.4\,\mathrm{keV}$ and c is the speed of light. The error on this conversion is the error on the velocity scaled with the factor E_0/c.

4.4.1 Isomeric Shift

The isomeric shift is a small shift in the energy of the ground state or the excited state and shows in the measurements as a velocity offset of the measured dip. The isomeric shift depends on the electronic charge distribution around the nucleus that can be different in emitter- and absorber material. Thus the emitted energy does not perfectly match the required absorption energy which can be compensated by the moving sledge. To calculate the isomeric shift it is therefore possible to use the determined position of the peak, which has an offset of zero and convert this velocity to an energy via eq. (12). For the three different fits this yields

$$\begin{aligned} E_{\text{iso, G}} &= (9.3 \pm 0.6)\,\mathrm{neV} \\ E_{\text{iso, L/V}} &= (8.5 \pm 0.5)\,\mathrm{neV} \end{aligned}$$

4.4.2 Effective Absorber Thickness

For the calculation of the Debye-Waller factor the effective absorber thickness T_A is needed. It can be calculated with help of the material parameters of the stainless steel absorber [1] via

$$T_A = f_A n_A \beta \sigma_0 d_A. \tag{13}$$

The needed parameters are (taken from [1]):

$$\begin{aligned} \text{absorber thickness} &: d_A = 25\,\mu\mathrm{m} \\ \text{number of iron atoms per m}^3 &: n_A \\ \text{proportion of Fe}^{56}\text{ iron} &: \beta = 0.022 \\ \text{cross section} &: \sigma_0 \\ \text{Debye-Waller factor of the absorber} &: f_A = 0.8 \end{aligned}$$

The cross section can be calculated as given in [6] with

$$\sigma_0 = \frac{1}{2\pi}\left(\frac{hc}{E_{\text{iron}}}\right)^2 \left(\frac{2I_e + 1}{2_g + 1}\right) \frac{1}{1+\alpha},$$

where $E_{\text{iron}} = 14.4\,\mathrm{keV}$. $I_e = 3/2$ and $I_g = 1/2$ are the contributing spin states and $\alpha = 8.58 \pm 0.18$ [2] the conversion coefficient. The calculation of the cross section yields

$$\sigma_0 = (2.46 \pm 0.05) \cdot 10^{-22}\,\mathrm{m}^2.$$

The number of iron atoms is calculated with the Avogadro constant N_A, the density of iron $\rho_{\mathrm{Fe}} = 7.89 \cdot 10^6\,\frac{\mathrm{g}}{\mathrm{m}^3}$ [9], the molar mass $m_{\mathrm{Fe}} = 55.8\,\frac{\mathrm{g}}{\mathrm{mol}}$ and the proportion of iron that is used in the stainless steel absorber $p = 0.70 \pm 0.05$ [1] via

$$n_A = \frac{N_A \rho_{\mathrm{Fe}}}{M_{\mathrm{Fe}}} p = (5.9 \pm 0.4) \cdot 10^{28}\,\frac{1}{\mathrm{m}^3}.$$

Combining these results and inserting in eq. (13) gives

$$T_A = 6.4 \pm 0.5.$$

During all the steps the errors have been calculated with Gaussian error propagation.

4.4.3 Debye-Waller Factor

With the calculation of the effective absorber thickness and the results of the fits the Debye-Waller factor of the source can be calculated. The Debye-Waller factor is the fraction of recoilless emitted or absorbed photons and can be calculated by [6]

$$f_S = \frac{\dot{N}(\infty) - \dot{N}(0)}{\dot{N}(\infty)} \left(1 - \exp\left(-\frac{1}{2}T_A\right) J_0\left(\frac{1}{2} i T_A\right)\right)^{-1}.$$

J_0 is the Bessel-function of zeroth order and the count rates are given by the fit parameters. $\dot{N}(\infty)$ is the rate without absorption, so it is given by the offset C of our fit and the difference $\dot{N}(\infty) - \dot{N}(0)$ is the amplitude A of the fit, as we are not actually interested in the depth at zero, but at the minimum, so at zero plus isomer shift. With this the equation can be rewritten and we can also use the fit parameters and their errors to calculate an error for the Debye-Waller factor:

$$f_S = \frac{A}{C} \left(1 - \exp\left(-\frac{1}{2}T_A\right) J_0\left(\frac{1}{2} i T_A\right)\right)^{-1}$$

and

$$s_{f_s} = \sqrt{\left(\frac{\partial f_S}{\partial C}\right)^2 s_C^2 + \left(\frac{\partial f_S}{\partial A}\right)^2 s_A^2 + \left(\frac{\partial f_S}{\partial T_A}\right)^2 s_{T_A}^2 + \left(\frac{\partial f_S}{\partial C}\right)\left(\frac{\partial f_S}{\partial A}\right) s_{AC} * *2},$$

with

$$\left(\frac{\partial f_S}{\partial A}\right) = \frac{f_S}{A}$$
$$\left(\frac{\partial f_S}{\partial C}\right) = -\frac{f_S}{C}$$
$$\left(\frac{\partial f_S}{\partial T_A}\right) = -\frac{A}{C} \cdot \frac{\left(\frac{1}{2}\exp\left(-\frac{1}{2}T_A\right) J_0\left(\frac{1}{2}iT_A\right) + \frac{1}{2} i \exp\left(-\frac{1}{2}T_A\right) J_1\left(\frac{1}{2}iT_A\right)\right)}{\left(1 - \exp\left(-\frac{1}{2}T_A\right) J_0\left(\frac{1}{2}iT_A\right)\right)^2}.$$

The errors on the fit parameters s_A, s_C and s_{AC} are taken from the covariance matrix as the root of the corresponding diagonal entry for the uncorrelated uncertainties and the square root of the correlated entry for s_{AC}. For the three different fits this calculation of the Debye-Waller factor of the source results in

$$f_{S,\mathrm{G}} = 0.39 \pm 0.02$$
$$f_{S,\mathrm{L}} = 0.44 \pm 0.02$$
$$f_{S,\mathrm{V}} = 0.44 \pm 0.04.$$

4.4.4 Lifetime of the 14.4 keV State in Fe^{57}

The natural line width Γ of a state is connected to its lifetime by $\tau = \hbar/\Gamma$ but this relation can not be directly applied in this experiment. The finite size of the emitter and the absorber cause an overlap between emission and absorption spectra, so the line width is increased and a correction factor has to be introduced as shown in [5, 7].

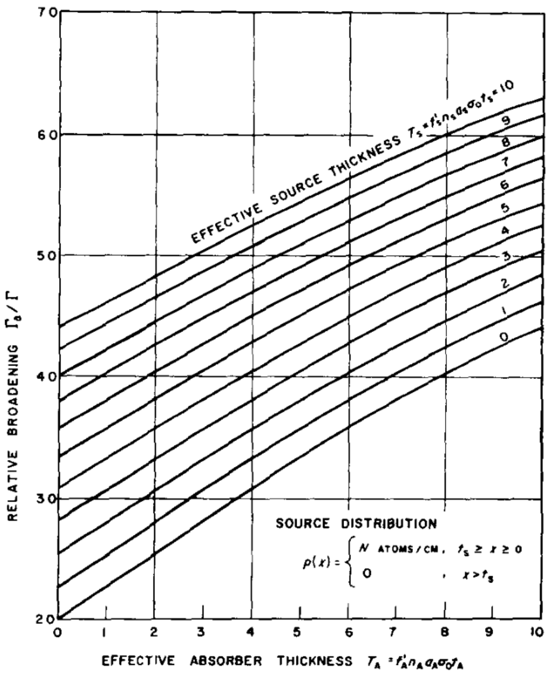


Figure 13: Relative line broadening of the Mößbauer absorption due to the absorber and source thickness. The values on the vertical axis range from 2 to 7, but are noted with a space as a separator.

Taking a look at fig. 13 we can see that for effective absorber and source thickness approaching zero the relative broadening approaches 2, so the observed line width is twice the natural line width of the Mößbauer spectrum, so the line width of the source adds up with the line width of the absorption spectrum. The effective absorber thickness has been calculated in section 4.4.2 and we found $T_A = 6.4 \pm 0.5$ for a thickness of $d_A = 25\,\mu\mathrm{m}$. In [1] an estimation for the thickness of the source is given with $d_S \approx 100\,\text{Å}$, so if the other parameters needed to calculate the effective source thickness are of the same magnitude, the effective source thickness will be very small and can be approximates by $T_S \approx 0$. For this case the formula of Visscher [5] can be used to calculate a correction factor

$$\begin{aligned} W &= 2(1.01 + 0.145T_A - 0.0025T_A^2), \qquad \text{for } 4 \leq T_A \leq 10 \\ s_W &= 2(0.145T_A - 0.005T_A)s_{T_A}. \end{aligned}$$

This calculation yields

$$W = 3.68 \pm 0.10.$$

The corrected line width can now be calculated by

$$\Gamma_{\text{corr.}} = \frac{\Gamma_0}{W}$$

and this corrected line width can be inserted in the known formula $\tau = \hbar/\Gamma_{\text{corr.}}$. For the Gauss fit the line width has to be calculated out of the fit parameters and is given by the Full-Width-Half-Maximum, so $\Gamma_\text{G} = FWHM_\text{G} = 2\sqrt{2\ln(2)}\sigma$ and the error scales with the same factor. For the Lorentz fit the line width is given by the width of the fit, so γ, for the Voigt profile only the width of the Lorenz part has to be used. The observed values are

$$\begin{aligned} \tau^{57}_{\text{Fe, G}} &= (71 \pm 4)\,\text{ns} \\ \tau^{57}_{\text{Fe, L}} &= (158 \pm 11)\,\text{ns} \\ \tau^{57}_{\text{Fe, V}} &= (158 \pm 16)\,\text{ns}, \end{aligned}$$

and the errors on these values were propagated with Gauss through the whole procedure.

4.5 Measurements of Iron

Besides measuring the lifetime or the Debye-Waller factor the Mößbauer effect can also be used to investigate hyperfine structure splitting, as the resolution of the data is rather fine. To do so the stainless steel absorber was replaced with an iron absorber and measurements for different velocities and different measurement times were performed. The rates and the errors were calculated same as for the stainless steel absorber and the data was cleared by its background, again using eq. (11).

The taken data is shown in fig. 14 and it shows the expected behaviour with six dips roughly symmetrized around a small offset of zero. To determine the position of the peaks a Gauss and a Lorentz both constructed as a sum of six single Gauss or Lorentz functions were fitted to the data, the fit parameters are shown in table 4. Again due to the high noise and errors in the data the reduced χ^2 is rather low, and we find $\chi^2_\text{G} \approx \chi^2_\text{L} = 0.82$. As the positions of the peaks do not vary significantly for the two models only the calculation with the Lorentz parameters is shown here.

The velocities were the dips are located can be converted to energies with the Doppler shift. These energies then indicate the shift due to the hyperfine splitting and are shown in table 2.

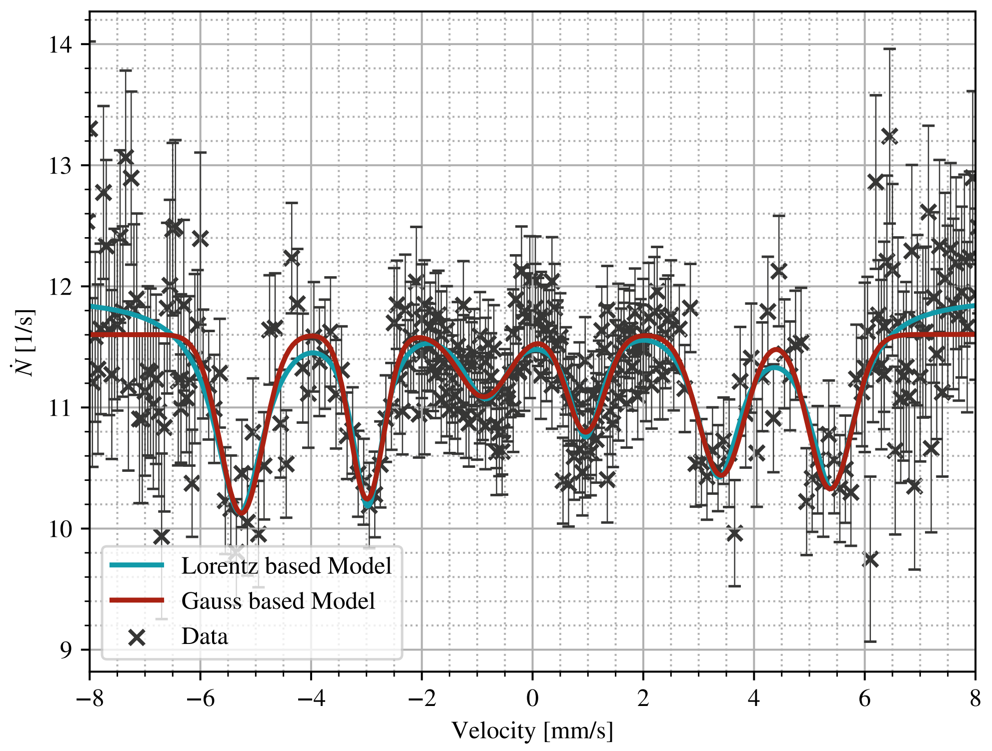


Figure 14: Absorption spectrum of iron. Two models were fitted to the data.

Table 2: Location of the dips as taken from the fit parameters in terms of a velocity and converted to energies.

Dip	ΔE_{HFS}[mm/s]	ΔE_{HFS}[neV]
1	-5.29 ± 0.06	-254 ± 3
2	-2.97 ± 0.04	-143 ± 2
3	-0.85 ± 0.08	-41 ± 4
4	0.96 ± 0.05	46 ± 2
5	3.36 ± 0.07	162 ± 3
6	5.41 ± 0.07	260 ± 3

4.5.1 Isomeric shift

Again the energies are shifted from a symmetry around zero by an offset. The energies that are observed due to the hyperfine splitting are symmetrical, so if there was not an isomeric shift the sum of positions of a dip and its counterpart on the other side would be zero. For an existing isomeric shift this sum gives the shift, so the sum of the three dips with their corresponding other were calculated:

$$\begin{aligned} E_{\text{iso},16} &= E_1 + E_6 \\ E_{\text{iso},25} &= E_2 + E_5 \\ E_{\text{iso},34} &= E_3 + E_4. \end{aligned}$$

All three shifts should give the same, so the mean of the three isomeric shifts is calculated and we obtain

$$\bar{E}_{\text{iso}} = (10 \pm 3)\,\text{neV}.$$

The error on this value was propagated from the error on the fit.

4.5.2 Magnetic Moment and Magnetic Field

When the energies are corrected by the isomeric shift they can be connected to the hyperfine splitting and we find

$$\begin{aligned} E_1 - \bar{E}_{\text{iso}} &= (\mu_e - \mu_g)B_z \\ E_2 - \bar{E}_{\text{iso}} &= (\frac{1}{3}\mu_e - \mu_g)B_z \\ E_3 - \bar{E}_{\text{iso}} &= (-\frac{1}{3}\mu_e - \mu_g)B_z \\ E_4 - \bar{E}_{\text{iso}} &= -(-\frac{1}{3}\mu_e - \mu_g)B_z \\ E_5 - \bar{E}_{\text{iso}} &= -(\frac{1}{3}\mu_e - \mu_g)B_z \\ E_6 - \bar{E}_{\text{iso}} &= -(\mu_e - \mu_g)B_z \end{aligned}$$

and by subtracting the corresponding energies a system of equations can be found

$$\begin{aligned} E_{16} = E_1 - E_6 &= 2(\mu_e - \mu_g)B_z \\ E_{25} = E_2 - E_4 &= 2(\frac{1}{3}\mu_e - \mu_g)B_z \\ E_{34} = E_3 - E_5 &= 2(-\frac{1}{3}\mu_e - \mu_g)B_z, \end{aligned}$$

that can be solved for the magnetic field B_z and the magnetic moment μ_e. Solving the system of equations gives

$$\begin{aligned} B_z &= -\frac{E_{34} - E_{25}}{4\mu_g} \\ \mu_e &= \frac{3\mu_g(E_{34} - E_{25})}{E_{34} + E_{25}} \end{aligned}$$

and the error is calculated with Gaussian error propagation what gives

$$s_{B_z} = \sqrt{\left(\frac{1}{4\mu_g}\right)^2 s_{E_{34}}^2 + \left(\frac{1}{4\mu_g}\right)^2 s_{E_{25}}^2 + \left(\frac{E_{34} - E_{25}}{4\mu_g^2}\right)^2 s_{\mu_g}^2}$$

$$s_{\mu_e} = \sqrt{\left(\frac{6\mu_g E_{34}}{(E_{34} - E_{25})^2}\right)^2 s_{E_{25}}^2 + \left(\frac{6\mu_g E_{25}}{(E_{34} - E_{25})^2}\right)^2 s_{E_{34}}^2 + \left(\frac{3(E_{34} - E_{25})}{E_{34} + E_{25}}\right) s_{\mu_g}^2},$$

and the errors on E_{34} and E_{25} are calculated straight out of the fit. μ_g is the magnetic moment of the ground state of Fe^{57} and given by $\mu_g = 0.090\,44 \pm 0.000\,07\mu_N$ [12] with $\mu_N = 3.152\,45 \cdot 10^{-8}\,\frac{\text{eV}}{\text{T}}$ the nuclear momentum [10].

Inserting the measured energies in the equations gives

$$B_z = (34 \pm 2)\,\text{T}$$

$$\mu_e = (-4.75 \pm 0.15)\,\frac{\text{neV}}{\text{T}} = (-0.151 \pm 0.005)\mu_N.$$

5 Summary and Discussion

Calibration First of all the MCA was calibrated and a channel-energy-relation was determine with help of five known absorbers. The relation is linear and a fit gives

$$f(E) = (173 \pm 3)\,\text{Channel/keV} \cdot E - (77 \pm 89)\,\text{Channel}.$$

With help of a first calibration the SCA window was set and checking the result with help of the second calibration shows the maximum of the Cobalt peak at $E_{\text{Co}} = (13.99 \pm 0.02)\,\text{keV}$, but also shows a rather big energy window. Therefore the setup of the window is satisfactory and the energy of interest is going to pass through, but it will have to be considered as a reason for noise in the data.

Background Measurements For the fixed energy window the Compton background was determine for the stainless steel absorber with help of Aluminium shielding. Extrapolation of the slower decaying part of a double exponential fit gives the offset

$$\dot{N}_{\text{Compton}} = B = (20.68 \pm 0.17)\,\text{Counts/s}.$$

Adding to that the attenuation of acrylic glass was determined, as both absorbers are inserted in two layers of acrylic glass. This has been done with theoretical calculations using material constants and with help of a measurement. The calculations and measurement give

$$R_{\text{exp.}} = 0.837 \pm 0.007$$
$$R_{\text{ana.}} = 0.77 \pm 0.04.$$

These two background measurements now were used to clear the following data from its background, for the dampening factor the experimental value was used. The reasoning for this decision is, that even though the error is higher, this value should represent the experiment with higher certainty, as the same setup was used to determine this value that is going to be used for the later measurements.

5.1 Stainless Steel

The spectrum of stainless steel has been recorded for different velocities and measuring times and a Poisson error has been applied on the count rates. The data was cleared by its background by subtracting the Compton background and taking the dampening into account. Afterwards three different fits, a Gaussian, a Lorentzian and a Voigt-fit were applied. Because of the high error of the data and the rather high noise it was not possible to determine the best model for the data with a simple χ^2-test and one would need to perform a hypothesis test with one model against the other to quantify the models. As it was not possible to determine the best model with the done calculations the different parameters that were calculated for stainless steel were calculated for all models.

Isomeric Shift The isomeric shift was calculated out of the offset of the data and the velocity offset was converted to an energy with help of the Doppler effect. The calculated isomeric shifts are

$$E_{\text{iso, G}} = (9.3 \pm 0.6)\,\text{neV}$$
$$E_{\text{iso, L/V}} = (8.5 \pm 0.5)\,\text{neV}.$$

Effective Absorber Thickness The effective absorber thickness, needed to calculate the Debye-Waller factor was determine with material parameters and we found

$$T_A = 6.4 \pm 0.5.$$

Debye-Waller Factor The Debye-Waller factor describes the fraction of recoilless emitted or absorbed photons and was calculated out of the fit parameters for offset and amplitude. The calculated factors are

$$f_{S,\mathrm{G}} = 0.39 \pm 0.02$$
$$f_{S,\mathrm{L}} = 0.44 \pm 0.02$$
$$f_{S,\mathrm{V}} = 0.44 \pm 0.04.$$

Lifetime The lifetime of the 14.4 keV state in Fe^{57} was determine out of the fitted line width given by the $FWHM$ of the peak. To take the absorber thickness into account the formula of Visscher was used to calculate a correction factor to the line width. The calculation with help of the fit parameters and the theoretically calculated effective absorber thickness gives

$$\tau^{57}_{\mathrm{Fe,\ G}} = (71 \pm 4)\,\mathrm{ns}$$
$$\tau^{57}_{\mathrm{Fe,\ L}} = (158 \pm 11)\,\mathrm{ns}$$
$$\tau^{57}_{\mathrm{Fe,\ V}} = (158 \pm 16)\,\mathrm{ns}.$$

The literature value found in [4] gives $\tau^{57}_{\mathrm{Fe}} = 141\,\mathrm{ns}$, so the Lorentzian and the Voigt-fit give a sensitive estimation for the lifetime, but the Gaussian model underestimates the lifetime. This was expected, as the lifetime depends on the width of the function and one difference between the Gaussian and Lorentzian model is, that the Lorentzian model gives a sharper peak, so the width is affected by the choice of the model. Anyways much more data with longer measurement times is needed so it is possible to decide for one model.

5.2 Iron

The absorption spectrum of iron was measured with the same setup, but for a wider velocity range. Again the data was cleared by the background and the error was determined as an Poisson error. To determine the positions of the dips a sum of Gaussian functions and a sum of Lorentzians were fitted to the data. Again the quality of the model could not be determined with help of a χ^2-test, but as we are only interested in the position of the peaks for the following calculations and both fits show the same behaviour in this context it was decided to only show the calculations with help of the Lorentzian here. This decision also makes sense, as the Lorentzian is the expected model from the physical point of view.

Isomeric Shift The positions of the six dips were determine in terms of velocities as the fit parameters and converted to energies with help of the Doppler shift. As the absorption spectrum shows a symmetry around zero apart form the isomeric shift it can be calculated by summing up the mirrored peaks. For higher certainty the mean of the three values was determine and we obtain

$$\bar{E}_{\mathrm{iso}} = (10 \pm 3)\,\mathrm{neV}.$$

Magnetic Moment and Magnetic Field By relating the energies to the magnetic fields and the magnetic moment of the excited and ground state, the magnetic field and the magnetic moment can be calculated out of the fit parameters. Solving a system of equations and inserting the fit parameters gives

$$\begin{aligned} B_z &= (34 \pm 2)\,\mathrm{T} \\ \mu_e &= (-4.75 \pm 0.15)\,\frac{\mathrm{neV}}{\mathrm{T}} = (-0.151 \pm 0.005)\mu_N. \end{aligned}$$

Literature values for these two measurements are $B = 33\,\mathrm{T}$ for the magnetic field at $300\,\mathrm{K}$ [4] and $\mu_e = -0.1549 \pm 0.0002\mu_N$ [10]. Both of these values lie within 1σ of our measurements, which also shows that the choice of the model is not of huge relevance for this measurement, as the calculation only depends on the position of the peak and both models yield comparable results for that parameter.

5.3 Estimation of Errors

During the whole experiment and analysis the estimation of errors is rather unpleasant and some further measurements are desired in order to be ceartain about the results. First of all a smaller energy window in the SCA would be of interest, this might yield better results as there is less background but also can lead to longer measurement times as interesting events might be accidentally cut off.

Furthermore it is of interest to perform all measurements, so background and the absorption spectra with longer measurement times, as the error in the used plots results from a combination of different quite high counting errors. The smaller errors might also lead to a decision which model fits the data better, as for high errors the difference between a Gaussian and a Lorentzian model is to small.

Apart from that it would be nice to take more data, so measure for more different velocities, but also perform the same measurement more often, so possible statistic effects cancel out.

References

[1] A. Zwerger and S. Winkelmann and M. Köhli and J. Wollrath. “Mößbauer-Effekt” (2017).

[2] V.P. Chechev and N.K. Kuzemenko. “Table of Radionucléides” (2017).

[3] Leonard Eyges. “Physics of the Mössbauer Effect”. *American Journal of Physics* 33.10 (1965), pp. 790–802. DOI: `10.1119/1.1970986`.

[4] Brent Fultz. “Mössbauer Spectrometry”. *Characterization of Materials* (2011).

[5] J. Heberle. “Linewidth of Mössbauer absorption”. *Nuclear Instruments and Methods* 58 (1968), pp. 90–92.

[6] 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`.

[7] 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: `10.1016/0029-554X(61)90122-7`.

[8] H.P. Meyers and H.P. Myers. *Introductory Solid State Physics, Second Edition.* Taylor & Francis, 1997.

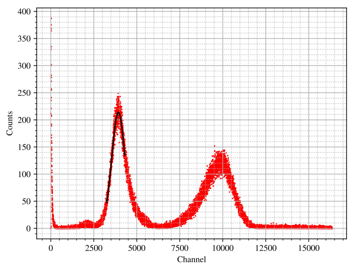
[9] “Physical Measurement Labatory” (). URL: `https://physics.nist.gov/cgi-bin/Star/compos.pl?matno=026`.

[10] “Physical Measurement Labatory” (). URL: `https://physics.nist.gov/cgi-bin/cuu/Value?munev%7Csearch_for=nuclear+magneton`.

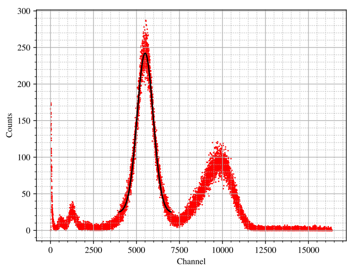
[11] Bogdan Povh et al. *Teilchen und Kerne.* Physics and astronomy online library. Springer Berlin Heidelberg, 2014.

[12] N.J. Stone. “Table of Nuclear Magnetic Dipole end Electric Quadrupole Moments” ().

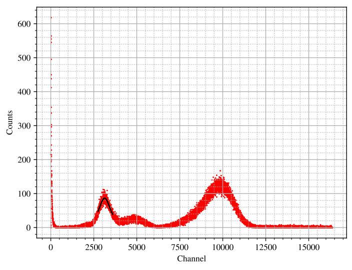
A Analysis



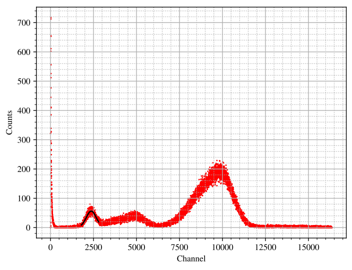
(a) Spectrum of Ag



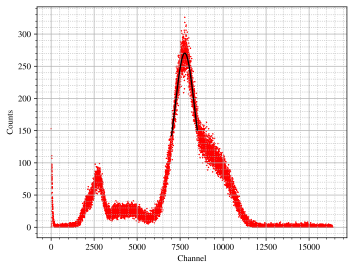
(b) Spectrum of Ba



(c) Spectrum of Mo



(d) Spectrum of Rb



(e) Spectrum of Tb

Figure 15: Spectra of different radioactive sources used to calibrate the MCA

Table 3: Calibration energy of the different measured probes with the channel they fill in the MCP. This data was taken with the later used Corse Gain of 100.

Probe	Energy [keV]	Channel
Ba	32.06	5503 ± 1
Ag	22.1	3929 ± 2
Mo	17.44	3132 ± 3
Rb	13.37	2371 ± 2
Tb	44.23	7771 ± 2

Table 4: Fit parameters of the Lorentz fit to the data of the iron spectrum. The offset is the same for all dips, as this parameter was not fitted seperately.

Dip	v_0[mm / s]	γ[mm / s]	A[1 / s]	C[1 / s]
1	-5.29 ± 0.06	0.55 ± 0.11	-3.0 ± 0.6	11.93 ± 0.10
2	-2.97 ± 0.04	0.35 ± 0.07	-1.7 ± 0.3	11.93 ± 0.10
3	-0.85 ± 0.08	0.57 ± 0.16	-1.3 ± 0.4	11.93 ± 0.10
4	0.96 ± 0.05	0.40 ± 0.08	-1.2 ± 0.3	11.93 ± 0.10
5	3.36 ± 0.07	0.51 ± 0.12	-2.2 ± 0.5	11.93 ± 0.10
6	5.41 ± 0.07	0.51 ± 0.12	-2.4 ± 0.5	11.93 ± 0.10

B Python-Code

Calibration

```python
"""
In this module the MCP calibration is performed.
"""

from matplotlib import pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

# This handles the used fonts in the plot to make it more or less consistent
# with the standard latex font.
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams['font.family'] = 'STIXGeneral'
plt.rcParams['mathtext.rm'] = 'Bitstream Vera Sans'
plt.rcParams['mathtext.it'] = 'Bitstream Vera Sans:italic'
plt.rcParams['mathtext.bf'] = 'Bitstream Vera Sans:bold'


def gauss(x: float, mu: float, sigma: float, amp: float, off: float) -> float:
    """
    Gauss function used to fit data.

    x: x-Position
    mu: Mean
    sigma: Standard Deviation
    amp: Amplitude
    off: Offset on y-Axis
    """
    return amp * np.exp(-(x - mu)**2/(2 * sigma**2)) + off


def linear(x: float, m: float, c: float) -> float:
    """
    Linear function used to fit data.

    x: x-Value
    m: slope
    c: Offset
    """
    return m * x + c


# Name of data
PROBES = ["ba", "ag", "mo", "rb", "tb"]

# Container for raw data
CHANNELS = [[] for _ in range(len(PROBES))]
COUNTS = [[] for _ in range(len(PROBES))]

MUS = []
SIGMAS = []
S_MUS = []
S_SIGMAS = []

INIT_BA = [5500, 500, 270, 0]
INIT_AG = [4000, 500, 3000, 0]
INIT_MO = [3100, 300, 100, 0]
INIT_RB = [2400, 200, 70, 0]
INIT_TB = [7700, 200, 300, 0]
INITS = [INIT_BA, INIT_AG, INIT_MO, INIT_RB, INIT_TB]

LIM_BA = (4000, 7000)
LIM_AG = (3250, 4300)
LIM_MO = (2750, 3600)
LIM_RB = (1800, 2800)
LIM_TB = (7000, 8500)
LIMITS = [LIM_BA, LIM_AG, LIM_MO, LIM_RB, LIM_TB]

GAUSS_X = []
GAUSS_Y = []

# Read data
for i in range(len(PROBES)):
    with open("./cal_at100/" + PROBES[i] + ".TKA", "r") as data:
        lines = data.read().split("\n")
        for j, line in enumerate(lines[1:-1]):
            COUNTS[i].append(float(line))
            CHANNELS[i].append(j)

    popt, pcov = curve_fit(gauss, CHANNELS[i][LIMITS[i][0]:LIMITS[i][1]],
                           COUNTS[i][LIMITS[i][0]:LIMITS[i][1]], p0=INITS[i])

    MUS.append(popt[0])
```

```python
    SIGMAS.append(popt[1])

    S_MUS.append(np.sqrt(pcov[0][0]))
    S_SIGMAS.append(np.sqrt(pcov[1][1]))

    xx = np.linspace(CHANNELS[i][LIMITS[i][0]], CHANNELS[i][LIMITS[i][1]], 100)
    yy = gauss(xx, *popt)
    GAUSS_X.append(xx)
    GAUSS_Y.append(yy)


for i in range(5):
    plt.scatter(CHANNELS[i], COUNTS[i], color="red", marker="x", s=0.5)
    plt.plot(GAUSS_X[i], GAUSS_Y[i], color="black", label="Gauss")
    plt.minorticks_on()
    plt.grid(which="minor", linestyle=":")
    plt.grid(which="major", linestyle="-")
    plt.xlabel("Channel")
    plt.ylabel("Counts")
    plt.savefig("calibration" + str(PROBES[i]) + ".pdf")
    plt.clf()


ENERGIES = [32.06, 22.10, 17.44, 13.37, 44.23]  # keV

# Linear regression for calibration
popt, pcov = curve_fit(linear, ENERGIES, MUS)

xx = np.linspace(min(ENERGIES) - 2, max(ENERGIES) + 2, 100)
yy = linear(xx, *popt)

print("energies = " + str(ENERGIES))
print("mus = " + str(MUS))
print("s_mus = " + str(S_MUS))

plt.scatter(ENERGIES, MUS, color="#A92112", marker="x")
plt.plot(xx, yy, color="black")
plt.minorticks_on()
plt.errorbar(ENERGIES, MUS, yerr=S_MUS, fmt="none",
             capsize=2.5, ecolor="#817F7F")
plt.grid(which="minor", linestyle=":")
plt.grid(which="major", linestyle="-")
plt.xlim(12, 46)
plt.xlabel("Energy [keV]")
plt.ylabel("Channel")
plt.show()
# plt.savefig("linreg_calibration.pdf")

print("Steigung: m = ", str(popt[0]), " channel/keV")
print("Steigung: m = ", str(np.sqrt(pcov[0][0])), " channel/keV")
print("Offset  : c = ", str(popt[1]), " channel")
print("Offset  : c = ", str(np.sqrt(pcov[1][1])), " channel")

# Save fit parameter to file
# with open("cal.txt", "w") as cal:
#     calpar = "Slope [channel/keV]\t Offset [channel]\n" +\
#             str(popt[0]) + "\t" + str(popt[1])
#     cal.write(calpar)
```

Compton Background

```python
"""
This module contains code to extract the compton background from
a aluminum shielding measurement.
"""

from matplotlib import pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
import os

# This handles the used fonts in the plot to make it more or less consistent
# with the standard latex font.
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams['font.family'] = 'STIXGeneral'
plt.rcParams['mathtext.rm'] = 'Bitstream Vera Sans'
plt.rcParams['mathtext.it'] = 'Bitstream Vera Sans:italic'
plt.rcParams['mathtext.bf'] = 'Bitstream Vera Sans:bold'


def double_exp(x: float, A: float, alpha: float,
               B: float, beta: float) -> float:
    """
    Function to perform a model fit:
        f(x) = A exp(alpha * x) + B exp(beta * x)
    """
    return A * np.exp(alpha * x) + B * np.exp(beta * x)


def exp(x: float, A: float, alpha: float) -> float:
    """
    Function to plot exponential fit:
    f(x) = A exp(alpha * x)
    """
    return A * np.exp(alpha * x)


# Get data names NOTE: Beware if the directory structure changes!
TMP = os.popen("ls").read()
DATA_NAMES = TMP[0:-1].split("\n")[0:-2]

# Initialize container for data
DATA = [list() for _ in range(len(DATA_NAMES))]
TIMES = list()

for i, name in enumerate(DATA_NAMES):
    with open(name, "r") as doc:
        lines = doc.read().split("\n")
        TIMES.append(float(lines[0]))
        for line in lines[2:-1]:
            DATA[i].append(float(line))

CPERSEC = [sum(data)/time for data, time in zip(DATA, TIMES)]
S_CPERSEC = [np.sqrt(sum(data))/time for data, time in zip(DATA, TIMES)]
THICKNESS = [float(name.split("mm")[0]) for name in DATA_NAMES]

popt, pcov = curve_fit(double_exp, THICKNESS, CPERSEC, sigma=S_CPERSEC)
xx = np.linspace(min(THICKNESS) - 1, max(THICKNESS) + 1, 60)
```

```python
yy = double_exp(xx, *popt)

yy_singl_exp = exp(xx, popt[2], popt[3])

plt.scatter(THICKNESS, CPERSEC, marker="x", color="#817F7F", label="Data")
plt.errorbar(THICKNESS, CPERSEC, yerr=S_CPERSEC, fmt="none",
             capsize=2.5, ecolor="#817F7F")
plt.plot(xx, yy, color="#A92112", label="Double Exponential Model Fit")
plt.plot(xx, yy_singl_exp, color="#129AA9", label="Single Exponential")
plt.minorticks_on()
plt.grid(which="minor", linestyle=":")
plt.grid(which="major", linestyle="-")
plt.xlabel("Thickness in [mm]")
plt.ylabel(r"$\dot{N}$ in [1/s]")
plt.xlim(0, 11)
plt.ylim(10, 30)
plt.legend()
plt.savefig("compton.pdf")

print("Fit values \n ------------------------------------")
print("A = " + str(popt[0]) + " +- " + str(np.sqrt(pcov[0][0])))
print("alpha = " + str(popt[1]) + " +- " + str(np.sqrt(pcov[1][1])))
print("B = " + str(popt[2]) + " +- " + str(np.sqrt(pcov[2][2])))
print("beta = " + str(popt[3]) + " +- " + str(np.sqrt(pcov[3][3])))
```

Acrylic Glass

```python
"""
Program used to calculate the dampening of acrylic glass
"""

import numpy as np

def read_data(filename: str) -> list:
    with open(filename, "r") as data:
        list = []
        raw_data = data.read()
        lines = raw_data.split("\n")
        for line in lines[:-1]:
            list.append(float(line))
        return list


# calculate dampening with experimental values

PLEXI_DATA = read_data("plexi_without_ssteel.TKA")
CLEAN_DATA = read_data("../compton_backgound/0.0mm.TKA")
CLEAN_DATA = read_data("../calibration/cobalt_data/cobalt_first_wincut.TKA")

PLEXI_TIME = PLEXI_DATA[0]
CLEAN_TIME = CLEAN_DATA[0]

PLEXI_COUNTS = np.sum(PLEXI_DATA[2::])
CLEAN_COUNTS = np.sum(CLEAN_DATA[2::])

PLEXI_RATE = PLEXI_COUNTS / PLEXI_TIME
CLEAN_RATE = CLEAN_COUNTS / CLEAN_TIME
```

```

s_plexi_rate = np.sqrt(PLEXI_COUNTS) / PLEXI_TIME
s_clean_rate = np.sqrt(CLEAN_COUNTS) / CLEAN_TIME

damp = PLEXI_RATE / CLEAN_RATE

s_damp = np.sqrt((s_plexi_rate / CLEAN_RATE)**2 + (PLEXI_RATE * s_clean_rate /
                 (CLEAN_RATE**2))**2)



# calculate theoretical dampening

D_PLEXI = 0.2  # cm
S_D = 0.0002  # cm
RHO_PLEXI = 1.19  # g / (cm)**3
ATTEN_COEFF = 1.1  # (cm)**2 / g
S_ATTEN = 0.2  # (cm)**2 / g

r_calc = np.exp(- ATTEN_COEFF * RHO_PLEXI * D_PLEXI)
s_r = r_calc * np.sqrt((RHO_PLEXI * D_PLEXI * S_ATTEN)**2 +
                       (ATTEN_COEFF * RHO_PLEXI * S_D)**2)

print("experimental dampening factor r = " + str(damp) + " +- " + str(s_damp))
print("theoretical dampening factor r = " + str(r_calc) + " +- " + str(s_r))
```

Stainless Steel

```
"""
This module is used to analyse the data measured using the
stainless steel absorber.
"""

from matplotlib import pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from scipy.special import voigt_profile
from scipy.special import jv

# This handles the used fonts in the plot to make it more or less consistent
# with the standard latex font.
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams['font.family'] = 'STIXGeneral'
plt.rcParams['mathtext.rm'] = 'Bitstream Vera Sans'
plt.rcParams['mathtext.it'] = 'Bitstream Vera Sans:italic'
plt.rcParams['mathtext.bf'] = 'Bitstream Vera Sans:bold'


def gauss(x: float, mu: float, sigma: float, amp: float, off: float) -> float:
    """
    Gauss function used to fit data.

    x: x-Position
    mu: Mean
    sigma: Standard Deviation
    amp: Amplitude
    off: Offset on y-Axis
    """
    return (amp / (np.sqrt(2*np.pi) * sigma)) *\
        np.exp(-(x - mu)**2/(2 * sigma**2)) + off


def lorentz(x: float, x0: float, gamma: float, A: float, C: float) -> float:
    """
    Lorentz function used to fit data.

    x: x-Value
    x0: Position of Maximum
    gamma: Width of the Peak
    A: Amplitude
    C: Off-set
    """
    return (A * gamma / np.pi) * ((x - x0)**2 + gamma**2)**(-1) + C


def voigt(x: float, x0: float, gamma: float,
          s0: float, A: float, C: float) -> float:
    """
    Voigt function wrapper used to fit data.

    x: x-Value
    x0: Position of Maximum
    gamma: Width of the lorentz part
    s0: Width of the gaussian part
    A: Amplitude
    C: Off-set
    """
    return A * voigt_profile((x - x0), s0, gamma) + C


# def chi_square_g(x_data: list, y_data: list, popt):
#     chi = 0
#     for x, y in zip(x_data, y_data):
#         chi += (y - gauss(x, *popt))**2 / gauss(x, *popt)
#     return chi / 4

def chi_square_g(x_data: list, y_data: list, popt, yerr):
    chi = 0
    for x, y, err in zip(x_data, y_data, yerr):
        chi += (y - gauss(x, *popt))**2 / err**2
    return chi / (len(x_data) - 4)


def t_gauss(x_data, y_data, popt):
    abw = []
    for x, y in zip(x_data, y_data):
        abw.append(y - gauss(x, *popt))
    mean = np.mean(abw)
    std = np.std(abw)
    return mean / std

def t_lorentz(x_data, y_data, popt):
    abw = []
    for x, y in zip(x_data, y_data):
        abw.append(y - lorentz(x, *popt))
```

```python
    mean = np.mean(abw)
    std = np.std(abw)
    return mean / std

def t_voigt(x_data, y_data, popt):
    abw = []
    for x, y in zip(x_data, y_data):
        abw.append(y - voigt(x, *popt))
    mean = np.mean(abw)
    std = np.std(abw)
    return mean / std

# def chi_square_l(x_data: list, y_data: list, popt):
#     chi = 0
#     for x, y in zip(x_data, y_data):
#         chi += (y - lorentz(x, *popt))**2 / lorentz(x, *popt)
#     return chi / 4

def chi_square_l(x_data: list, y_data: list, popt, yerr):
    chi = 0
    for x, y, err in zip(x_data, y_data, yerr):
        chi += (y - lorentz(x, *popt))**2 / err**2
    return chi / (len(x_data) - 4)


# def chi_square_v(x_data: list, y_data: list, popt):
#     chi = 0
#     for x, y in zip(x_data, y_data):
#         chi += (y - voigt(x, *popt))**2 / voigt(x, *popt)
#     return chi / 5


def chi_square_v(x_data: list, y_data: list, popt, yerr):
    chi = 0
    for x, y, err in zip(x_data, y_data, yerr):
        chi += (y - voigt(x, *popt))**2 / err**2
    return chi / (len(x_data) - 5)

COUNT_OFFSET = 20.684847  # 1/s
OFFSET_ERR = 0.170814  # 1/s

R_EXP = 0.837
S_R = 0.008

# Get raw data
VEL = list()
COUNTS = list()
TIME = list()
for i in ["1", "2", "3", "4"]:
    with open("ssteel_" + i + ".txt", "r") as doc:
        lines = doc.read().split("\n")
        for line in lines[0:-1]:
            entries = line.split("\t")
            VEL.append(float(entries[0].replace(',', '.')))
            COUNTS.append(float(entries[2].replace(',', '.')))
            TIME.append(float(entries[1].replace(',', '.')))

# Calculate counting rates NOTE: Go from ms to s
```

```
RATES = [(c/(t/1000) - COUNT_OFFSET) / R_EXP for c, t in zip(COUNTS, TIME)]
# Calculate counting rate errors
S_RATES_BETW = [np.sqrt((np.sqrt(c)/(t/1000))**2 + OFFSET_ERR**2)for c, t in
                zip(COUNTS, TIME)]
S_RATES = [np.sqrt((s_x / R_EXP)**2 + (x * S_R / R_EXP**2)**2) for x, s_x in
           zip(RATES, S_RATES_BETW)]

# Sort the data by velocity
DATA = list(zip(VEL, RATES, S_RATES))
DATA = sorted(DATA, key=lambda tup: tup[0])

# Split the data
VEL = [d[0] for d in DATA]
RATES = [d[1] for d in DATA]
S_RATES = [d[2] for d in DATA]

# Gauss Fit
INITS_G = [0.15, 0.2, -4, 31]
popt_g, pcov_g = curve_fit(gauss, VEL, RATES, p0=INITS_G, sigma=S_RATES)
xx_g = np.linspace(min(VEL), max(VEL), 200)
yy_g = gauss(xx_g, *popt_g)

# Lorentz Fit
INITS_L = [0.15, 0.2, -4, 31]
popt_l, pcov_l = curve_fit(lorentz, VEL, RATES, p0=INITS_L, sigma=S_RATES)
xx_l = np.linspace(min(VEL), max(VEL), 200)
yy_l = lorentz(xx_l, *popt_l)

# Voigt Fit
INITS_V = [0.15, 0.1, 0.1, -4, 31]
popt_v, pcov_v = curve_fit(voigt, VEL, RATES, p0=INITS_V, sigma=S_RATES)
xx_v = np.linspace(min(VEL), max(VEL), 200)
yy_v = voigt(xx_v, *popt_v)

# plt.plot(VEL, RATES)
plt.plot(xx_g, yy_g, label="Gauss Model", color="#A92112")
plt.plot(xx_l, yy_l, label="Lorentz Model", color="#129AA9")
plt.plot(xx_v, yy_v, label="Voigt Model", color="purple", linestyle="dashed")
plt.scatter(VEL, RATES, marker="x", label="Data", color="#363635")
plt.errorbar(VEL, RATES, yerr=S_RATES, fmt="none",
             capsize=2.5, elinewidth=0.5, ecolor="#363635")

plt.ylabel(r"$\dot{N}$ [1/s]")
plt.xlabel("Velocity [mm/s]")
plt.minorticks_on()
plt.grid(which="minor", linestyle=":")
plt.grid(which="major", linestyle="-")
plt.legend(loc=3)
plt.savefig("steel.pdf")
# plt.show()

print("Gauss fit: \n ---------------------------------")
print("mean = " + str(popt_g[0]) + " +- " + str(np.sqrt(pcov_g[0][0])))
print("sigma = " + str(popt_g[1]) + " +- " + str(np.sqrt(pcov_g[1][1])))
print("amp = " + str(popt_g[2]) + " +- " + str(np.sqrt(pcov_g[2][2])))
print("offset = " + str(popt_g[3]) + " +- " + str(np.sqrt(pcov_g[3][3])))
print("chi = " + str(chi_square_g(VEL, RATES, popt_g, S_RATES)))
print("t = " + str(t_gauss(VEL, RATES, popt_g)))
```

```python
print("")
print("Lorentz fit: \n ---------------------------------")
print("mean = " + str(popt_l[0]) + " +- " + str(np.sqrt(pcov_l[0][0])))
print("sigma = " + str(popt_l[1]) + " +- " + str(np.sqrt(pcov_l[1][1])))
print("amp = " + str(popt_l[2]) + " +- " + str(np.sqrt(pcov_l[2][2])))
print("offset = " + str(popt_l[3]) + " +- " + str(np.sqrt(pcov_l[3][3])))
print("chi = " + str(chi_square_l(VEL, RATES, popt_l, S_RATES)))
print("t = " + str(t_lorentz(VEL, RATES, popt_l)))
print("")
print("Voigt fit: \n ---------------------------------")
print("pos of max lorentz = " + str(popt_v[0]) + " +- " +
      str(np.sqrt(pcov_v[0][0])))
print("width lorentz = " + str(popt_v[1]) + " +- " +
      str(np.sqrt(pcov_v[1][1])))
print("width gauss = " + str(popt_v[2]) + " +- " + str(np.sqrt(pcov_v[2][2])))
print("amp = " + str(popt_v[3]) + " +- " + str(np.sqrt(pcov_v[3][3])))
print("offset = " + str(popt_v[4]) + " +- " + str(np.sqrt(pcov_v[4][4])))
print("chi = " + str(chi_square_v(VEL, RATES, popt_v, S_RATES)))
print("t = " + str(t_voigt(VEL, RATES, popt_v)))


# calculate isomeric shift

def v_to_E(v: float) -> float:
    """
    Takes sledge velocity and return energy
    """
    E_GAMMA = 14.4e3  # eV
    SPEED_OF_LIGHT = 2.99792e8  # m / s
    return E_GAMMA * v*10**(-3) / SPEED_OF_LIGHT


print("==========================================")
print("isomeric shift [eV] \n------------------------------")
print("gauss model: E_iso = " + str(v_to_E(popt_g[0])) + " +- " +
      str(v_to_E(np.sqrt(pcov_g[0][0]))))
print("lorentz model: E_iso = " + str(v_to_E(popt_l[0])) + " +- " +
      str(v_to_E(np.sqrt(pcov_l[0][0]))))
print("voigt model: E_iso = " + str(v_to_E(popt_v[0])) + " +- " +
      str(v_to_E(np.sqrt(pcov_v[0][0]))))

# effective absorber thickness

d_a = 25e-6  # m
beta = 0.022
f_a = 0.8
rho_fe = 7.87e6  # g / (m)**3
m_fe = 55.8  # g / mol
p_ssteel = 0.7
n_A = 6.022e23  # 1 / mol
SPEED_OF_LIGHT = 2.99792e8  # m / s
E_GAMMA = 14.4e3  # eV
h = 4.136e-15  # ev s
alpha = 8.58
s_alpha = 0.18

number_of_iron = n_A * rho_fe * p_ssteel / m_fe
s_noi = n_A * rho_fe * 0.05 / m_fe
```

```python

cross_section = (1 / (2 * np.pi)) * (h * SPEED_OF_LIGHT / E_GAMMA)**2 *\
                (4 / 2) * (1 / (1 + alpha))

s_cs = cross_section * (s_alpha / (1 + alpha)**2)

T_A = f_a * number_of_iron * beta * cross_section * d_a
s_TA = np.sqrt((f_a * s_noi * beta * cross_section * d_a)**2 +
               (f_a * number_of_iron * beta * s_cs * d_a)**2)


print("\n effective absorber thickness \n --------------------------")
print("number of iron = " + str(number_of_iron) + " +- " + str(s_noi))
print("cross section = " + str(cross_section) + " +- " + str(s_cs))
print("absorber thickness = " + str(T_A) + " +- " + str(s_TA))

# debye-waller factor


def debye_waller(A: float, C: float):
    """
    Calculates the fucking debye waller factor for given shit.
    A: Amplitude of Fit (Lorenz / Gauss)
    C: Offset of Fit
    """
    return (- A / C) * (1 / (1 - np.exp(- 0.5 * T_A) * jv(0, 0.5j * T_A)))

def debye_error(A: float, C: float, s_A: float, s_C: float, s_AC: float):
    """
    Calculate a beatiful error....
    Kill me.
    """
    dfda = debye_waller(A, C) / A
    dfdc = debye_waller(A, C) / C
    dfdt = (0.5 * np.exp(- 0.5 * T_A) * jv(0, 0.5j * T_A) + 0.5j *
             np.exp(- 0.5 * T_A) * jv(1, 0.5j * T_A)) /\
            (1 - np.exp(- 0.5 * T_A) * jv(0, 0.5j * T_A))**2
    return np.sqrt((dfda * s_A)**2 + (dfdc * s_C)**2 + (dfdt * s_TA)**2 + 2 *
                   dfda * dfdc * s_AC)


print("\n=============================")
print("Debyeeee waller lorentz = " + str(debye_waller(popt_l[2], popt_l[3])) +
      " +- " + str(debye_error(popt_l[2], popt_l[3], np.sqrt(pcov_l[2][2]),
                              np.sqrt(pcov_l[3][3]), pcov_l[2][3])))
print("Debyeeee waller gauss = " + str(debye_waller(popt_g[2], popt_g[3])) +
      " +- " + str(debye_error(popt_g[2], popt_g[3], np.sqrt(pcov_g[2][2]),
                              np.sqrt(pcov_g[3][3]), pcov_g[2][3])))
print("Debyeeee waller voigt = " + str(debye_waller(popt_v[3], popt_v[4])) +
      " +- " + str(debye_error(popt_v[3], popt_v[4], np.sqrt(pcov_v[3][3]),
                              np.sqrt(pcov_v[4][4]), pcov_v[3][4])))


# calculate lifetime

def lifetime(sigma: float):
    correction = 2 * (1.01 + 0.145 * T_A - 0.0025 * T_A**2)
    return v_to_E(sigma) / correction
```

```python


def lifetime_err(sigma: float, sigma_err: float):
    correction = 2 * (1.01 + 0.145 * T_A - 0.0025 * T_A**2)
    s_c = 2 * (0.145 - 0.005 * T_A) * s_TA
    return np.sqrt((v_to_E(sigma) * s_c / correction**2)**2 +
                   (v_to_E(sigma_err) / correction)**2)


def lifetime_seconds(lifetime: float):
    return h / (2 * np.pi * lifetime)


def lifetime_sec_err(life: float, lifeerror: float):
    return (h * lifeerror) / (2 * np.pi * life**2)


print("\n===========================")

print("correction factor = " + str(2 * (1.01 + 0.145 * T_A - 0.0025 * T_A**2))
      + " +- " + str(2 * (0.145 - 0.005 * T_A) * s_TA))
print("lifetime gauss = " + str(lifetime(2*np.sqrt(2*np.log(2))*popt_g[1])) + " +- "
     +
      str(lifetime_err(2*np.sqrt(2*np.log(2))*popt_g[1], 2*np.sqrt(2*np.log(2))*np.
    sqrt(pcov_g[1][1]))) + "eV")
print("lifetime gauss = " + str(lifetime_seconds(lifetime(2*np.sqrt(2*np.log(2))*
    popt_g[1]))) + " +- " +
      str(lifetime_sec_err((lifetime(2*np.sqrt(2*np.log(2))*popt_g[1])),
          lifetime_err(2*np.sqrt(2*np.log(2))*popt_g[1], 2*np.sqrt(2*np.log(2))*np.
    sqrt(pcov_g[1][1])))) + "s")
print("lifetime lorentz = " + str(lifetime(popt_l[1])) + " +- " +
      str(lifetime_err(popt_l[1], np.sqrt(pcov_l[1][1]))) + "eV")
print("lifetime lorentz = " + str(1e9 * lifetime_seconds(lifetime(popt_l[1]))) + "
    +- " +
      str(1e9 * lifetime_sec_err((lifetime(popt_l[1])),
          lifetime_err(popt_l[1], np.sqrt(pcov_l[1][1])))) + "ns")
print("lifetime voigt = " + str(lifetime(popt_v[1])) + " +- " +
      str(lifetime_err(popt_v[1], np.sqrt(pcov_v[1][1]))) + "eV")
print("lifetime voigt = " + str(1e9 * lifetime_seconds(lifetime(popt_v[1]))) + " +-
    " +
      str(1e9 * lifetime_sec_err((lifetime(popt_v[1])),
          lifetime_err(popt_v[1], np.sqrt(pcov_v[1][1])))) + "ns")

print(len(VEL))
```

Iron

```python
"""
This module is used to analyse the hyperfine spectrum of iron. Two different
models for peak-position-estimation are fitted to the data.
"""

from matplotlib import pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from scipy.special import voigt_profile

```

```
SINGLE_FIT = False

# This handles the used fonts in the plot to make it more or less consistent
# with the standard latex font.
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams['font.family'] = 'STIXGeneral'
plt.rcParams['mathtext.rm'] = 'Bitstream Vera Sans'
plt.rcParams['mathtext.it'] = 'Bitstream Vera Sans:italic'
plt.rcParams['mathtext.bf'] = 'Bitstream Vera Sans:bold'


def gauss(x: float, mu: float, sigma: float, amp: float, off: float) -> float:
    """
    Gauss function used to fit data.

    x: x-Position
    mu: Mean
    sigma: Standard Deviation
    amp: Amplitude
    off: Offset on y-Axis
    """
    return (amp / (np.sqrt(2*np.pi) * sigma)) *\
        np.exp(-(x - mu)**2/(2 * sigma**2)) + off


def monster_gauss(x,
                  mu1, mu2, mu3, mu4, mu5, mu6,
                  s1, s2, s3, s4, s5, s6,
                  a1, a2, a3, a4, a5, a6,
                  off):
    """
    Six added gaussian distributions to fit to the data.
    """
    output = gauss(x, mu1, s1, a1, 0) +\
        gauss(x, mu2, s2, a2, 0) +\
        gauss(x, mu3, s3, a3, 0) +\
        gauss(x, mu4, s4, a4, 0) +\
        gauss(x, mu5, s5, a5, 0) +\
        gauss(x, mu6, s6, a6, 0) + off
    return output


def lorentz(x: float, x0: float, gamma: float, A: float, C: float) -> float:
    """
    Lorentz function used to fit data.

    x: x-Value
    x0: Position of Maximum
    gamma: Width of the Peak
    A: Amplitude
    C: Off-set
    """
    return (A * gamma / np.pi) * ((x - x0)**2 + gamma**2)**(-1) + C


def monster_lorentz(x,
                    mu1, mu2, mu3, mu4, mu5, mu6,
                    s1, s2, s3, s4, s5, s6,
                    a1, a2, a3, a4, a5, a6,
                    off):
    """
    Six added lorentz distributions to fit to the data.
    """
    output = lorentz(x, mu1, s1, a1, 0) +\
        lorentz(x, mu2, s2, a2, 0) +\
        lorentz(x, mu3, s3, a3, 0) +\
        lorentz(x, mu4, s4, a4, 0) +\
        lorentz(x, mu5, s5, a5, 0) +\
        lorentz(x, mu6, s6, a6, 0) + off
    return output

def chi_square_g(x_data: list, y_data: list, popt, yerr):
    chi = 0
    for x, y, err in zip(x_data, y_data, yerr):
        chi += (y - monster_gauss(x, *popt))**2 / err**2
    return chi / (len(x_data) - 4)


def chi_square_l(x_data: list, y_data: list, popt, yerr):
    chi = 0
    for x, y, err in zip(x_data, y_data, yerr):
        chi += (y - monster_lorentz(x, *popt))**2 / err**2
    return chi / (len(x_data) - 4)

COUNT_OFFSET = 20.684847  # 1/s
OFFSET_ERR = 0.170814  # 1/s

R_EXP = 0.837
S_R = 0.008

# Get raw data
VEL = list()
COUNTS = list()
TIME = list()
for i in ["1", "2", "3", "4", "5"]:
    with open("iron_" + i + ".txt", "r") as doc:
        lines = doc.read().split("\n")
        for line in lines[0:-1]:
            entries = line.split("\t")
            VEL.append(float(entries[0].replace(',', '.')))
            COUNTS.append(float(entries[2].replace(',', '.')))
            TIME.append(float(entries[1].replace(',', '.')))

# Calculate counting rates NOTE: Go from ms to s
RATES = [(c/(t/1000) - COUNT_OFFSET) / R_EXP for c, t in zip(COUNTS, TIME)]
# Calculate counting rate errors
S_RATES_BETW = [np.sqrt((np.sqrt(c)/(t/1000))**2 + OFFSET_ERR**2)for c, t in
                zip(COUNTS, TIME)]
S_RATES = [np.sqrt((s_x / R_EXP)**2 + (x * S_R / R_EXP**2)**2) for x, s_x in
           zip(RATES, S_RATES_BETW)]

# Sort the data by velocity
DATA = list(zip(VEL, RATES, S_RATES))
DATA = sorted(DATA, key=lambda tup: tup[0])

# Split the data
```

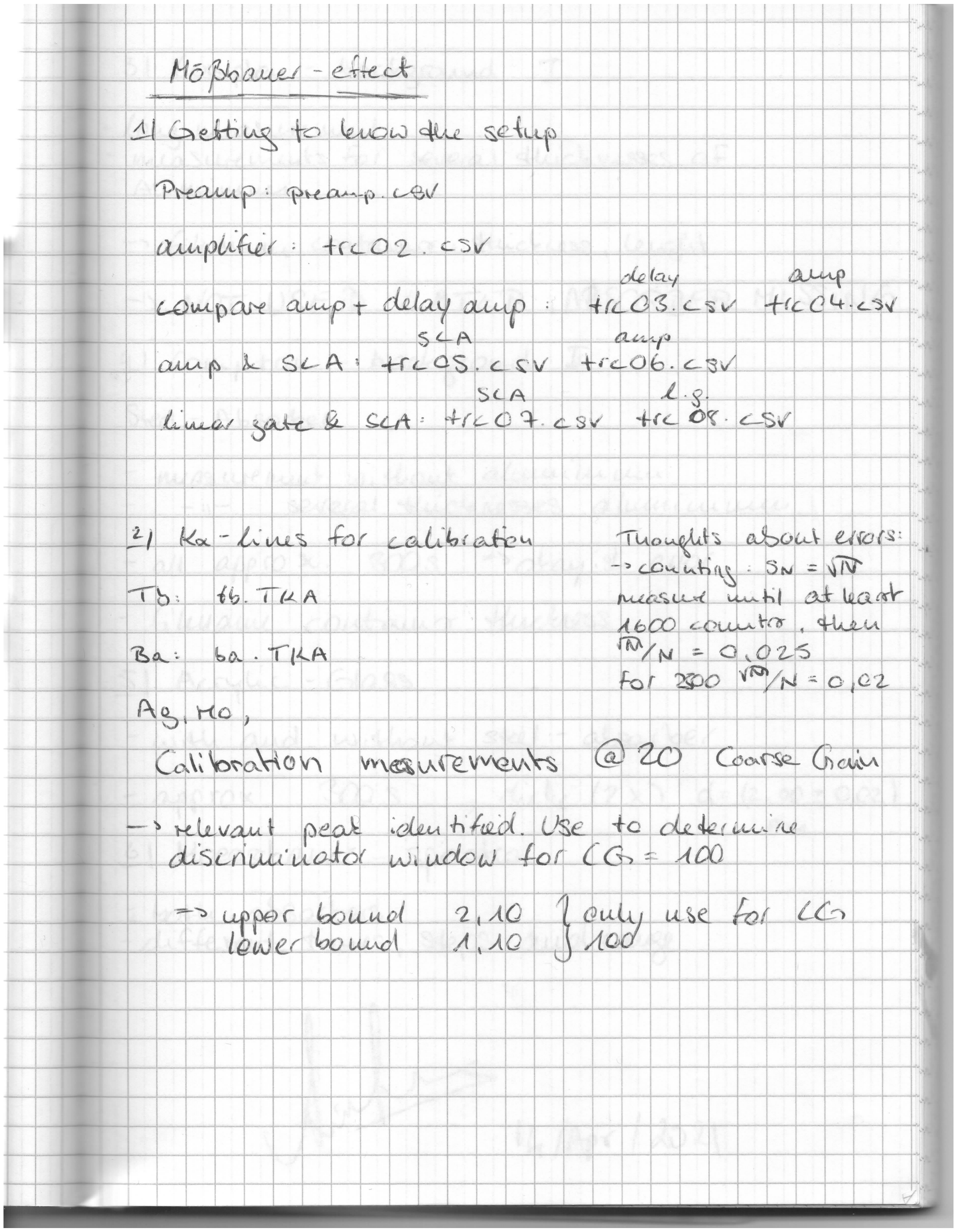
```
VEL = [d[0] for d in DATA]
RATES = [d[1] for d in DATA]
S_RATES = [d[2] for d in DATA]

# Get size of data
# print("Number of total data points: ", len(VEL))

# LET THE FITTING BEGIN \(0_o)/
# ============================

# PEAK 1
INIT1 = [-5.25, 0.5, -2, 30.25]
# PEAK 2
INIT2 = [-3, 0.2, -1, 30.25]
# PEAK 3
INIT3 = [-0.6, 0.2, -1, 30.25]
# PEAK 4
INIT4 = [0.6, 0.2, -1, 30.25]
# PEAK 5
INIT5 = [3, 0.2, -1, 30.25]
# PEAK 6
INIT6 = [5.25, 0.5, -2, 30.25]
INITS = [INIT1, INIT2, INIT3, INIT4, INIT5, INIT6]

if SINGLE_FIT:
    # Initialize Container
    POPT_G = []
    PCOV_G = []
    XX_G = []
    YY_G = []
    for i in range(6):
        # Perform model fit
        popt_g, pcov_g = curve_fit(gauss, VEL, RATES,
                                   p0=INITS[i], sigma=S_RATES)
        POPT_G.append(popt_g)
        PCOV_G.append(popt_g)

        # Save plot data
        xx = np.linspace(min(VEL), max(VEL), 100)
        XX_G.append(xx)
        YY_G.append(gauss(xx, *popt_g))

# Monster model fit
# =================
MONSTER_INIT = [i[0] for i in INITS] +\
        [i[1] for i in INITS] +\
        [i[3] for i in INITS] + [30]

popt_gm, pcov_gm = curve_fit(monster_gauss, VEL, RATES,
                             p0=MONSTER_INIT, sigma=S_RATES)
popt_lm, pcov_lm = curve_fit(monster_lorentz, VEL, RATES,
                             p0=MONSTER_INIT, sigma=S_RATES)

xx_gm = np.linspace(min(VEL) - 2, max(VEL) + 2, 950)
yy_gm = monster_gauss(xx_gm, *popt_gm)

xx_lm = np.linspace(min(VEL) - 2, max(VEL) + 2, 950)
yy_lm = monster_lorentz(xx_lm, *popt_lm)
```

```python

# Plot single gauss models
if SINGLE_FIT:
    for i in range(6):
        plt.plot(XX_G[i], YY_G[i])

# Plot data
plt.scatter(VEL, RATES, marker="x", color="#363635", label="Data")
plt.errorbar(VEL, RATES, yerr=S_RATES, fmt="none",
             capsize=2.5, elinewidth=0.5, ecolor="#363635")

# Plot monster lorentz model
plt.plot(xx_lm, yy_lm, color="#129AA9",
         label="Lorentz based Model", linewidth=2)

# Plot monster gauss model
plt.plot(xx_gm, yy_gm, color="#A92112",
         label="Gauss based Model", linewidth=2)

plt.minorticks_on()
plt.grid(which="minor", linestyle=":")
plt.grid(which="major", linestyle="-")
plt.ylabel(r"$\dot{N}$ [1/s]")
plt.xlabel("Velocity [mm/s]")
plt.xlim(-8, 8)
plt.legend(loc=3)
plt.savefig("iron.pdf")
# plt.show()

# Start calculating isomeric shift


def v_to_E(v: float) -> float:
    """
    Takes sledge velocity and return energy
    """
    E_GAMMA = 14.4e3  # eV
    SPEED_OF_LIGHT = 2.99792e8  # m / s
    return E_GAMMA * v*10**(-3) / SPEED_OF_LIGHT


print("Position of the dips\n===========================")
for i in [0, 1, 2, 3, 4, 5]:
    print("Gauss Dip " + str(i + 1) + ": (" + str(popt_gm[i]) + " +- "
          + str(np.sqrt(pcov_gm[i][i])) + ") mm / s")
    print("Gauss Dip " + str(i + 1) + ": (" + str(v_to_E(popt_gm[i] * 1e9)) + " +- "
          + str(v_to_E(np.sqrt(pcov_gm[i][i]) * 1e9)) + ") neV")
print("\n===============================\n")
for i in [0, 1, 2, 3, 4, 5]:
    print("Lorentz Dip " + str(i + 1) + ": (" + str(popt_lm[i + 18]) + " +- "
          + str(np.sqrt(pcov_lm[i + 18][i + 18])) + ") mm / s")
    print("Lorentz Dip " + str(i + 1) + ": (" + str(v_to_E(popt_lm[i] * 1e9)) + " +-
     "
          + str(v_to_E(np.sqrt(pcov_lm[i][i]) * 1e9)) + ") neV")

ENERGIES_G = [v_to_E(popt_gm[i]) for i in [0, 1, 2, 3, 4, 5]]
S_ENERGIES_G = [v_to_E(np.sqrt(pcov_gm[i][i])) for i in [0, 1, 2, 3, 4, 5]]
ENERGIES_L = [v_to_E(popt_lm[i]) for i in [0, 1, 2, 3, 4, 5]]
```

```
S_ENERGIES_L = [v_to_E(np.sqrt(pcov_lm[i][i])) for i in [0, 1, 2, 3, 4, 5]]

print("chi = " + str(chi_square_g(VEL, RATES, popt_gm, S_RATES)))
print("chi = " + str(chi_square_l(VEL, RATES, popt_lm, S_RATES)))
# Isoshifts

ISO_16_G = ENERGIES_G[0] + ENERGIES_G[5]
ISO_25_G = ENERGIES_G[1] + ENERGIES_G[4]
ISO_34_G = ENERGIES_G[2] + ENERGIES_G[3]
ISO_16_L = ENERGIES_L[0] + ENERGIES_L[5]
ISO_25_L = ENERGIES_L[1] + ENERGIES_L[4]
ISO_34_L = ENERGIES_L[2] + ENERGIES_L[3]


S_ISO_16_G = np.sqrt(S_ENERGIES_G[0]**2 + S_ENERGIES_G[5]**2)
S_ISO_25_G = np.sqrt(S_ENERGIES_G[1]**2 + S_ENERGIES_G[4]**2)
S_ISO_34_G = np.sqrt(S_ENERGIES_G[2]**2 + S_ENERGIES_G[3]**2)
S_ISO_16_L = np.sqrt(S_ENERGIES_L[0]**2 + S_ENERGIES_L[5]**2)
S_ISO_25_L = np.sqrt(S_ENERGIES_L[1]**2 + S_ENERGIES_L[4]**2)
S_ISO_34_L = np.sqrt(S_ENERGIES_L[2]**2 + S_ENERGIES_L[3]**2)

MEAN_ISO_G = (ISO_16_G + ISO_25_G + ISO_34_G) / 3
MEAN_ISO_L = (ISO_16_L + ISO_25_L + ISO_34_L) / 3

S_MEAN_ISO_G = np.sqrt(S_ISO_16_G**2 + S_ISO_25_G**2 + S_ISO_34_G**2) / 3
S_MEAN_ISO_L = np.sqrt(S_ISO_16_L**2 + S_ISO_25_L**2 + S_ISO_34_L**2) / 3

print("\n Isomeric shifts \n=================================")
print("Gauss : " + str(MEAN_ISO_G * 1e9) + " +- " + str(S_MEAN_ISO_G * 1e9) +
      " neV")
print("Lorentz : " + str(MEAN_ISO_L * 1e9) + " +- " + str(S_MEAN_ISO_L * 1e9) +
      " neV")

# calc B field
E_16_G = ENERGIES_G[0] - ENERGIES_G[5]
E_25_G = ENERGIES_G[1] - ENERGIES_G[4]
E_34_G = ENERGIES_G[2] - ENERGIES_G[3]
E_16_L = ENERGIES_L[0] - ENERGIES_L[5]
E_25_L = ENERGIES_L[1] - ENERGIES_L[4]
E_34_L = ENERGIES_L[2] - ENERGIES_L[3]

MU_N = 3.15245e-8
MU_G = 0.09044 * 3.15245e-8  # eV / T
S_MU_G = 0.00007 * 3.15245e-8  # eV / T

B_GAUSS = - (E_34_G + E_25_G) / (4 * MU_G)
B_LORENZ = - (E_34_L + E_25_L) / (4 * MU_G)

# calculate B error

dbd34 = B_GAUSS / E_34_G
dbd25 = B_GAUSS / E_25_G
dbdmu = B_GAUSS / MU_G

dbd34_L = B_LORENZ / E_34_L
dbd25_L = B_LORENZ / E_25_L
dbdmu_L = B_LORENZ / MU_G

```

```
S_B_GAUSS = np.sqrt(dbd34**2 * S_ISO_34_G**2 + dbd25**2 * S_ISO_25_G**2 +
                    dbdmu**2 * S_MU_G**2)

S_B_LORENZ = np.sqrt(dbd34_L**2 * S_ISO_34_L**2 + dbd25_L**2 * S_ISO_25_L**2 +
                     dbdmu_L**2 * S_MU_G**2)

# calculate magnetic moment

MU_E_GAUSS = (3 * MU_G * (E_34_G - E_25_G)) / (E_34_G + E_25_G)
dmud34 = (6 * MU_G * E_25_G) / (E_34_G + E_25_G)**2
dmud25 = - (6 * MU_G * E_34_G) / (E_34_G + E_25_G)**2
dmudmu = MU_E_GAUSS / MU_G

S_MU_E_GAUSS = np.sqrt(dmud34**2 * S_ISO_34_G**2 + dmud25**2 * S_ISO_25_G**2 +
                       dmudmu**2 * S_MU_G**2)

MU_E_LORENZ = (3 * MU_G * (E_34_L - E_25_L)) / (E_34_L + E_25_L)
dmud34_L = (6 * MU_G * E_25_L) / (E_34_L + E_25_L)**2
dmud25_L = - (6 * MU_G * E_34_L) / (E_34_L + E_25_L)**2
dmudmu_L = MU_E_LORENZ / MU_G

S_MU_E_LORENZ = np.sqrt(dmud34_L**2 * S_ISO_34_L**2 +
                        dmud25_L**2 * S_ISO_25_L**2 +
                        dmudmu_L**2 * S_MU_G**2)

print("\n Magnetic field and moment\n==========================")
print("Gauss")
print(" B = (" + str(B_GAUSS) + " +- " + str(S_B_GAUSS) + ") T")
print(" MU_E = (" + str(MU_E_GAUSS) + " +- " + str(S_MU_E_GAUSS) + ") eV / T")
print(" MU_E in MU_N = (" + str(MU_E_GAUSS / MU_N) + " +- "
      + str(S_MU_E_GAUSS / MU_N) + ") MU_N")
print("Lorentz")
print(" B = (" + str(B_LORENZ) + " +- " + str(S_B_LORENZ) + ") T")
print(" MU_E = (" + str(MU_E_LORENZ) + " +- " + str(S_MU_E_LORENZ)
      + ") eV / T")
print(" MU_E in MU_N = (" + str(MU_E_LORENZ / MU_N) + " +- "
      + str(S_MU_E_LORENZ / MU_N) + ") MU_N")
```

C Lab Notes



Mößbauer - effect

1) Getting to know the setup

Preamp: preamp.csv

amplifier: trc02.csv

compare amp + delay amp: $\overset{\text{delay}}{\text{trc03.csv}}$ $\overset{\text{amp}}{\text{trc04.csv}}$

amp & SCA: $\overset{\text{SCA}}{\text{trc05.csv}}$ $\overset{\text{amp}}{\text{trc06.csv}}$

linear gate & SCA: $\overset{\text{SCA}}{\text{trc07.csv}}$ $\overset{\text{l.g.}}{\text{trc08.csv}}$

2) Kα-lines for calibration

Tb: tb.TKA

Ba: ba.TKA

Ag, Mo,

Thoughts about errors:
→ counting: $s_N = \sqrt{N}$
measure until at least 1600 counts, then $\sqrt{N}/N = 0{,}025$
for 2500 $\sqrt{N}/N = 0{,}02$

Calibration measurements @ 20 Coarse Gain

→ relevant peak identified. Use to determine discriminator window for CG = 100

⇒ upper bound 2,10 } only use for CG
lower bound 1,10 } 100

3| Compton - background I

- long measurement
- measurements for several thicknesses of Aluminium

-> filename contains: thickness, lenght

-> NOT USED LATER, ABSORBER MISSING

4| Compton - backgroud II

Steel - Absorber

- measurement without aluminium
- -"- several thicknesses aluminium

- all approx. 300s -> okayish error

- filename contains thickness

5| Acrylic - Glass

- with and without steel - absorber

- approx. 300s, dicke (2x) d = (2,00 ± 0,02) mm

6| Moessbauer - spectra

- both absorbers
- different times, steps and range

[illegible]

14/Apr/2021