Contents

1. Introduction

In the early 20^{th} century emission and re-absorption of X-rays in gases had been observed. The origin of the radiation are transitions of the orbital electrons. But the resonant absorption of gamma-radiation of nuclear transitions could not be measured. It was later found that the reason for this is the recoil of the nucleus due to the emission of the high energetic photon due to which the photon looses energy and can not excite an atom with the same transition. This was only solved when Rudolf Mößbauer studied resonance absorption in solids for his PhD in 1958. Since the transition lines get sharper at low temperatures Mößbauer expected to measure even less resonant absorption for cooled samples. To his surprise the probability of resonant absorption was increased compared to the samples above room temperature. This was then explained by the reduced recoil at low temperatures where crystals get stiffer and lattice vibrations are reduced. This effect is named after Mößbauer who received the Nobel prize in physics for this discovery in 1961.
In this experiment the Mößbauer effect is used to measure spectra of nuclear transition. This kind of spectroscopy is called Mößbauer spectroscopy which is performed here with the 14.4 keV transition of excited ^{57}Fe. The resonant transition line of stainless steel is measured and the hyperfine structure of natural iron is determined. These spectra allow the determination of the isomeric shifts of the absorber and the Debye-Waller factor of the source. Furthermore a lower limit for the lifetime of the 14.4 keV state is found.

2. Theoretical Considerations

2.1. Gamma Radiation

The half life of ^{57}Co is (272.11 ± 0.26) d [4]. The decay is induced via electron capture. In this process an electron from an inner atomic shell (K or L) is captured by the nucleus and a proton decays into a neutron and an electron neutrino:

$$\mathrm{p} + \mathrm{e}^- \longrightarrow \mathrm{n} + \nu_\mathrm{e}.$$

In the shell the electron leaves a hole which is filled by an electron from an outer shell which leads to the emission of X-ray photons or the Auger-Meitner effect. In the Auger-Meitner effect the energy which is lost by the electron filling the hole is transferred to another electron. With the additional energy this electron can get emitted. The decay product of ^{57}Co is ^{57}Fe. The whole level scheme of the decay is shown in Figure 1.

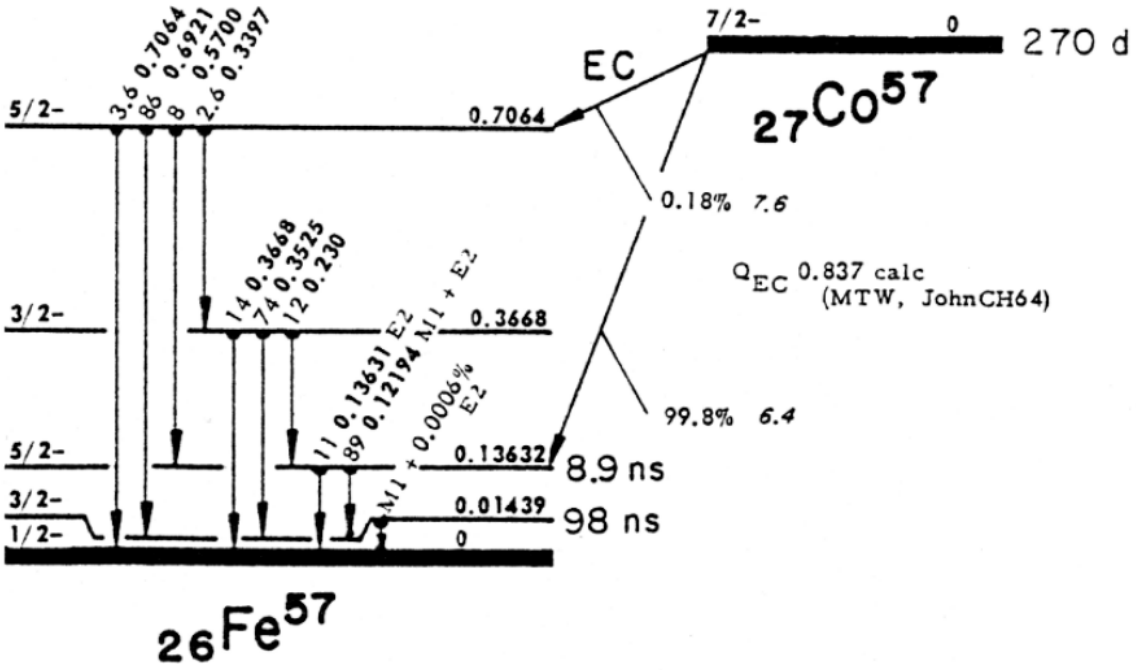


Fig. 1: Decay of ^{57}Co into excited states of ^{57}Fe. The transition which is of special interest in this experiment is from the first excited state at 14.39 keV to the ground state with an half life time of 98 ns [1].

Figure 1 displays that ^{57}Co decays into excited states of ^{57}Fe. The most important decay channel, with a probability of 99.8 %, leads to the second excited level which has an energy of 136.32 keV and a half life time of 8.9 ns. This decays further, either directly or via the first excited state to the ground state. The first excited state has an energy of approximately 14.4 keV and a half life time of 98 ns and therefore a mean life time of $\tau = 141$ ns. The decay from the first excited state to the ground state is used in this experiment for the Mößbauer spectroscopy.

2.2. Interaction of Electromagnetic Radiation with Matter

There are three main processes, which are responsible for the interaction between electromagnetic radiation and matter [5]: the photoelectric effect, the Compton effect and pair production. The energy ranges in which they occur and dominate are shown in Figure 2.

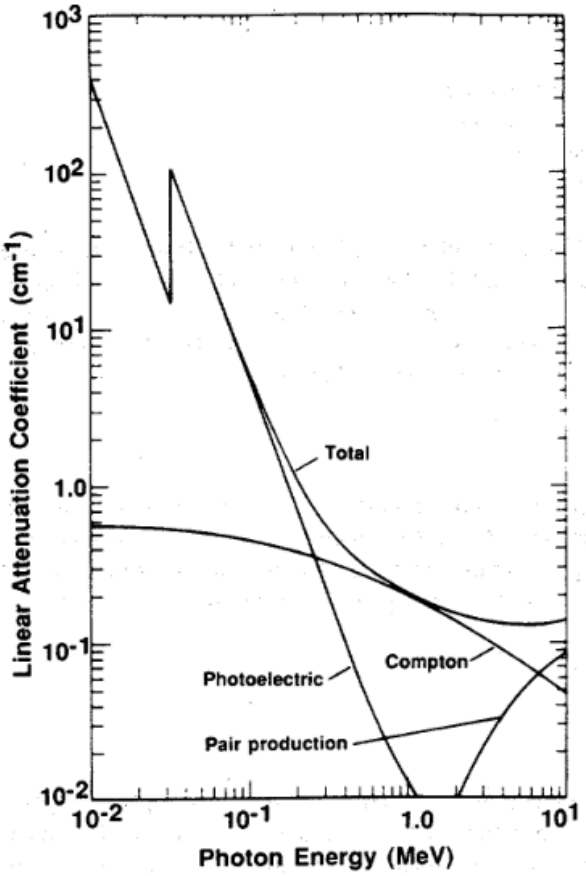


Fig. 2: Energy ranges of interaction processes of photons with matter taken from [6]. At low energies the photoelectric effect is dominant in the absorption process of photons. At larger energies the Compton effect gets more prominent until the pair production takes over.

2.2.1. Photoelectric Effect

In the photoelectric effect, which was first explained by Albert Einstein in 1905 [7], a photon transfers its total energy $E_\gamma = \hbar\omega$ to a shell electron. The energy of the excited electron is

$$E_e = E_\gamma - E_\mathrm{B},$$

with the binding energy E_B. If the energy of the photon is larger than the binding energy, the electron is expelled from its orbit. The resulting hole is filled by an electron from an outer shell under emission of a characteristic radiation or an Auger-Meitner electron.

2.2.2. Compton Effect

The Compton effect describes the inelastic scattering of a photon with a free or weak bound electron [8]. The photon transfers a part of its energy to the electron which

changes the movement directions of both photon and electron and thus also their energy. In this experiment the Compton effect occurs for photons with 122 keV, 136.32 keV and 14.4 keV, since these are mainly produced in the used source (see Figure 1). The photons with higher energies are shifted down by Compton scattering into the range of the 14.4 keV photons, which distorts the measured rates at that energy. Due to the energy dependent attenuation of photons in materials like aluminium [1], a double exponential decay in the counting rate is expected, when gradually shielding a detector with aluminium. This behaviour will be used to determine the Compton background in this experiment (see Section 4.4).

2.2.3. Pair Production

Photons with at least two times the resting energy of an electron can lead to pair production, the creation of electron-positron pairs, in the field of a nucleus. This process only occurs for photons with energies larger than 1022 MeV, which is two times the resting mass of an electron or a positron (511 keV). Since positrons are meta-stable particles they annihilate again with an electron under the emission of at least two photons.

2.2.4. Attenuation of Gamma Radiation by Acrylic Glass

In this experiment the interaction of gamma radiation with acrylic glass is of special interest. How well the glass transmits the radiation can be quantified by either the transmission factor T or the mass-attenuation coefficient μ/ρ. They are connected as stated in [9] by

$$T = \exp\left\{-\frac{\mu}{\rho}\rho\, d\right\}, \tag{1}$$

with the density of acrylic glass ρ and its thickness d. By rearranging this the mass-attenuation coefficient is gained

$$\frac{\mu}{\rho} = -\ln(T)\frac{1}{\rho\, d}. \tag{2}$$

2.3. Linewidth of Gamma Radiation

All emission and absorption lines of nuclear transitions have a natural line width, the full width at half maximum of

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

with the lifetime τ and the reduced Planck constant $\hbar$. This follows from Heisenberg's uncertainty relation which states that energy and time of a quantum object can only be determined up to some uncertainty

$$\Delta E \Delta t \geq \hbar.$$

For the analyzed transition of ^{57}Fe with an energy of 14.4 keV, this leads to a relative linewidth of $\Gamma_{\text{nat}}/E_\gamma \approx 3 \cdot 10^{-13}$ [10]. This small size makes it difficult to measure resonance absorption (see Section 2.4).

2.4. Nuclear Resonance Absorption

A photon which is emitted by a nuclear transition with energy E_0 can be reabsorbed by another nucleus, which is excited in this process. This is called nuclear resonance absorption. Photons which are emitted by free atoms do not hold the whole energy E_0, since the nucleus receives a recoil energy. Thus E_0 is reduced by

$$\Delta E = \frac{E_\gamma^2}{2mc^2} - E_\gamma \frac{v_\mathrm{t}}{c}, \tag{3}$$

with the photon energy E_γ, the mass m of the atom and v_t the thermal velocity. The first term is the recoil energy E_r. The second term with $v_\mathrm{t} = p_\mathrm{t}/m$ results from the thermal movement with the momentum in the direction of the emission p_t and describes the energy shift due to the Doppler effect.
For the 14.4 keV transition in ^{57}Fe the recoil energy is $E_\mathrm{r} \approx 2 \cdot 10^{-3}$ eV. This is several orders of magnitude larger than the natural linewidth of the transition with $\Gamma_\mathrm{nat} = 4.7 \cdot 10^{-9}$ eV [10]. If the distribution of v_t is broad enough the recoil energy is compensated, with some probability. By cooling the atoms the distribution of v_r is shifted to smaller velocities and thus the probability to achieve resonance absorption is reduced.

2.5. Mößbauer Effect

The recoilless emission and absorption of gamma radiation by nuclei is called Mößbauer effect. This can only occur in atoms which are bound in solids, since their mass m is big enough to reduce the recoil energy drastically. Since the number of atoms in a lattice is in the range of 10^{23}, the energy, which is gained by each atom, can be neglected in comparison to the natural linewidth of nuclear transitions. For 1 mol of ^{57}Fe the recoil energy for the absorption/emission of a 14.4 keV-photon is approximately $3 \cdot 10^{-27}$ eV and thus negligible compared to the natural linewidth $\Gamma_\mathrm{nat} \approx 4.7$ neV.
This recoilless emission of photons is used in the experiment for the Mößbauer spectroscopy. By using Equation 3 and the excitation energy of the nucleus E_0 with $\Delta E = E_0 - E_\gamma$ and with no recoil energy,

$$E_0 = E_\gamma \left(1 - \frac{v}{c}\right) \tag{4}$$

is obtained, the classical limit $v \ll c$ of the Doppler effect. With this equation E_0 is determined by moving the absorber with a velocity v towards the source or away from it. By using different velocities the intensity of the transmitted light changes. This gives the profile of the gamma-lines. For the 14.4 keV−line the Mößbauer effect allows a resolution of 1 in 10^{12} which is approximately the size of one sheet of paper on the distance between the earth and the sun [11].

2.5.1. Excitation of Phonons

The small recoil can be absorbed by the crystal as lattice vibrations, since the structure of the lattice becomes less rigid at increasing temperatures. Only at 0 K the atoms would form a stiff lattice. For small deviations from the resting position of the lattice points the interaction potential can be approximated as harmonic. Under a quantum mechanical view such a system with N atoms can only occupy discrete total energies

$$E_n = 3N\hbar\omega \left(\langle n \rangle + \frac{1}{2}\right),$$

by so called phonons (for more detail see i.e., [12] or [13]). With no further assumptions the phonon spectrum can only be approximated for very high and very low temperatures. In order to determine the spectrum more easily some models exist. The most prominent are the models by Einstein and Debye which will be outlined in the following paragraphs.

In the Einstein model the atoms are assumed to oscillate all with the same frequency ω_E. Due to this single photons have an energy of $E = \hbar\omega_\mathrm{E}$ which is also the only allowed recoil energy.

In the Debye model the frequency is proportional to the crystals momentum

$$\omega_\mathrm{s} = v_\mathrm{s} k,$$

with the speed of sound v_s and the wave number k. This leads to a continuous spectrum up to the Debye frequency ω_D. Via thermal energy a temperature can be associated to this frequency

$$\Theta_\mathrm{D} = \frac{\hbar\omega_\mathrm{D}}{k_\mathrm{B}},$$

with the Boltzmann constant k_B. This temperature is in the order of $10^2\,\mathrm{K}$.

2.5.2. Debye-Waller Factor

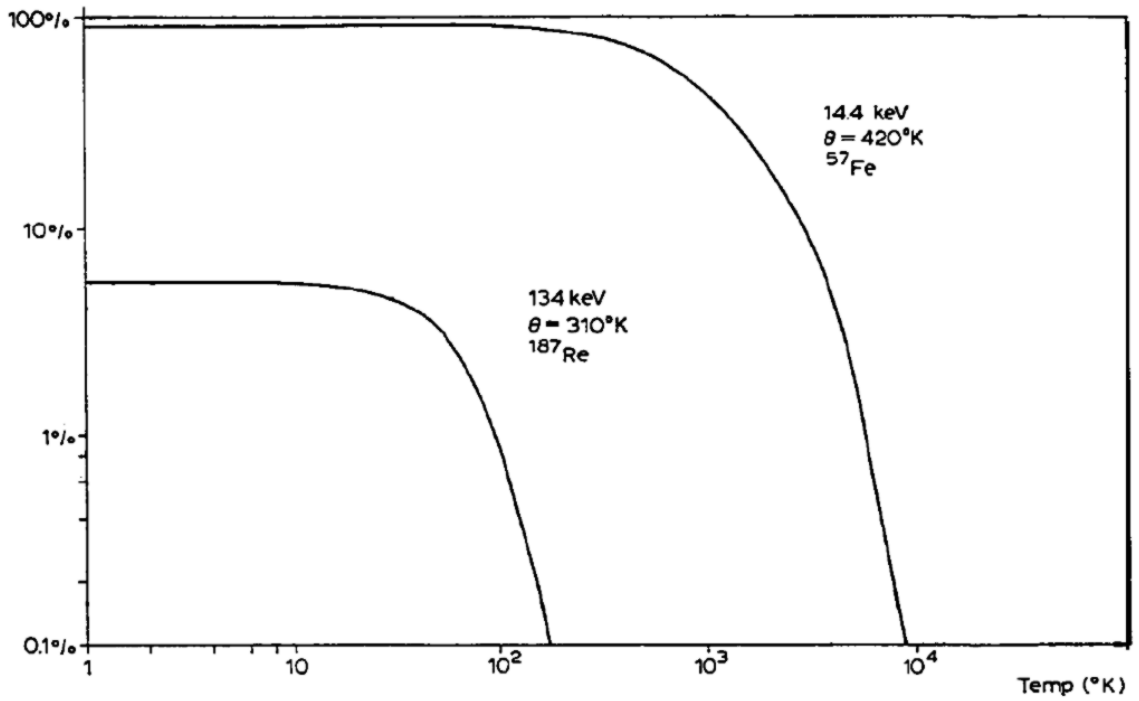
The fraction of recoilless nuclear transitions is called the Debye-Waller factor f. This quantity indicates the relative amount of photons, which are emitted from the nucleus with no recoil.
In the Debye model, this fraction of recoilless nuclear transitions can be expressed by

$$f = \exp\left\{-\frac{3E_\mathrm{r}}{2k_\mathrm{B}\Theta_\mathrm{D}}\left(1 + \frac{4T^2}{\Theta_\mathrm{D}^2}\int_0^{\Theta_\mathrm{D}/T}\frac{x\,dx}{e^x - 1}\right)\right\},$$

as described in [10]. If $T \leq \Theta_\mathrm{D}$, the integral can be approximated to

$$f \approx \exp\left\{-\frac{E_\mathrm{r}}{k_\mathrm{B}\Theta_\mathrm{D}}\left(\frac{3}{2} + \frac{\pi^2 T^2}{\Theta_\mathrm{D}^2}\right)\right\}.$$

This function is illustrated in Figure 3 for two transitions in dependence of the temperature T, with set Debye temperatures Θ_D. The 134 keV transition of ^{187}Re and the 14.4 keV transition of ^{57}Fe, which is utilized in this experiment are displayed.
The relevance of the 14.4 keV transition in ^{57}Fe for Mößbauer spectroscopy can be seen, as ^{57}Fe shows a recoilless transition probability of 91 % [10] at room temperature. Using this transition as a source in a Mößbauer spectrometer ensures that no complex cooling is required, making the spectrometer smaller, lighter and cheaper. Because of this a ^{57}Co source was used for the MIMOS II Mößbauer spectrometers for the Mars Exploration Rovers Spirit and Opportunity, for close-up investigations of the martian surface. The whole unit weights only around 500 g [14].



Fractions of recoil-free nuclear transitions (Debye-Waller factors) in ^{57}Fe and ^{187}Re, shown as functions of the temperature.

Fig. 3: Debye-Waller factor for the 134 keV transition of ^{187}Re and the 14.4 keV transition of ^{57}Fe taken from [1].

2.6. Isomeric Shift

The exact positions of the energy levels of a nucleus depend on the surrounding charge distribution. In a solid the charge distribution around a nucleus is mainly the result of the electrons in the shell of the nucleus.
If the charge distributions of the source and the absorber are identical, the spectrum is distributed symmetrically around $v = 0\,\mathrm{m\,s^{-1}}$ in a Mößbauer spectrum. If different materials are used with different charge distributions, the whole spectrum is shifted to a velocity $v \neq 0$.
Additionally, the first excited state of ^{57}Fe has a different spin configuration than the ground state which also leads to an isomeric shift at the transitions.

2.7. Hyperfine Splitting

A magnetic field in vicinity to the nucleus, which can be induced by the movement of electrons in an atom, lifts degeneracies of nuclear states. The resulting energy level structure is called hyperfine structure. The state with nuclear spin I is split into several lines which are shifted by the energy

$$E = -\frac{\mu m_I}{I} B,$$

where μ is the nuclear magnetic moment of the state, m_I the magnetic quantum number and B the current magnetic field. Exemplary, Figure 4 shows the hyperfine structure of ^{57}Fe. In the figure the ground state is labeled with I and the excited state with I^*.

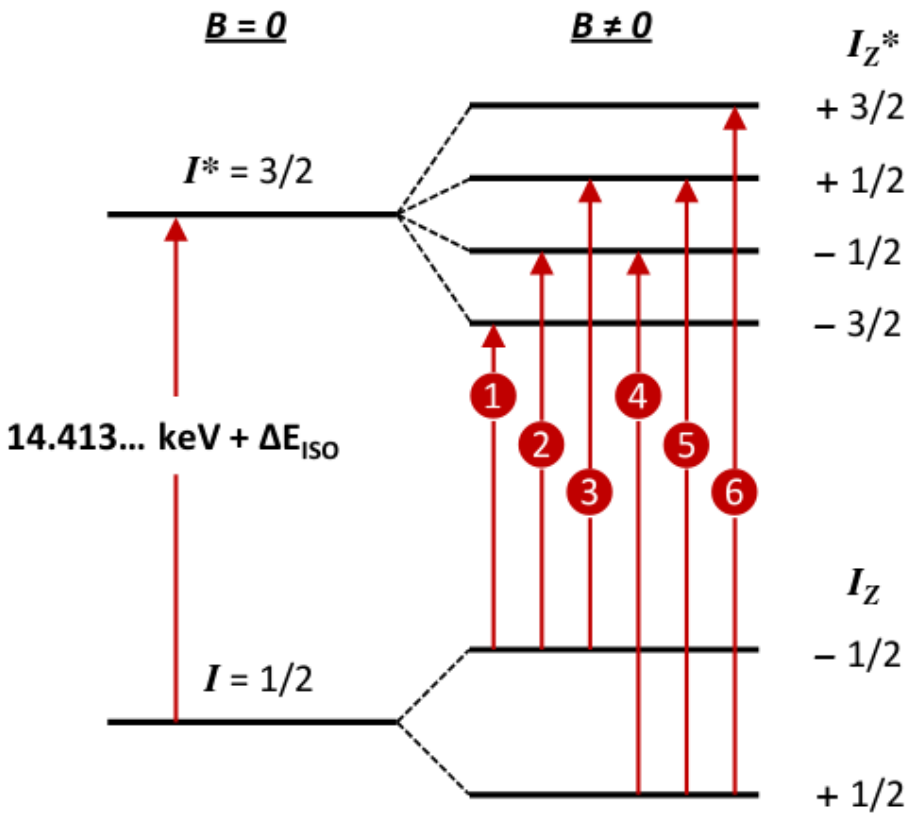


Fig. 4: Hyperfine structure of ^{57}Fe from [15]. Shown is the ground state $I = 1/2$, which is split to $m_{I_z} = \pm 1/2$, and the first excited state $I^* = 3/2$, which splits into $m_{I_z^*} = \pm 3/2,\ \pm 1/2$, if a magnetic field is present near the nucleus.

The ground state $I_\mathrm{g} = 1/2$ is split into the two energy levels, corresponding to the magnetic quantum numbers $m_{I_\mathrm{g}} = \pm 1/2$. The first excited state $I_\mathrm{e} = 3/2$ splits into $m_{I_\mathrm{e}} = \pm 3/2,\ \pm 1/2$. From the selection rules $\Delta m_I = 0, \pm 1$, six allowed transitions and thus six absorption lines follow. In this experiment such splitting will be observed in the measurement with a natural iron absorber. The Mößbauer spectrum of a stainless steel absorber only shows one line since here the spin correlation time τ is small enough to satisfy $\tau A/\hbar \ll 1$, with the hyperfine coupling constant A which gives the spectral line spacing of a nucleus [16].
The transition energies are shifted compared to those of a free nucleus by

$$\Delta E = E_\mathrm{Iso} + \left(\frac{\mu_\mathrm{g} m_{I_\mathrm{g}}}{I_\mathrm{g}} - \frac{\mu_\mathrm{e} m_{I_\mathrm{e}}}{I_\mathrm{e}} \right) \cdot B,$$

with the isomeric shift E_Iso, $\mu_\mathrm{g/e}$ the nuclear magnetic moments and $I_\mathrm{g/e}$ the nuclear spins with their magnetic quantum numbers $m_{I_\mathrm{g/I_e}}$ for the ground and excited state. The hyperfine splitting is

$$E = \left(\frac{\mu_\mathrm{g} m_{I_\mathrm{g}}}{I_\mathrm{g}} - \frac{\mu_\mathrm{e} m_{I_\mathrm{e}}}{I_\mathrm{e}} \right) \cdot B. \tag{5}$$

Table 1 lists the allowed hyperfine transitions of ^{57}Fe. It also shows the corresponding quantum numbers and E/B, determined by Equation 5.

Trans.	m_{I_g}	m_{I_e}	E/B
1	-1/2	-3/2	$\mu_\mathrm{e} - \mu_\mathrm{g}$
2	-1/2	-1/2	$\frac{1}{3}\,\mu_\mathrm{e} - \mu_\mathrm{g}$
3	-1/2	1/2	$-\frac{1}{3}\,\mu_\mathrm{e} - \mu_\mathrm{g}$
4	1/2	-1/2	$\frac{1}{3}\,\mu_\mathrm{e} + \mu_\mathrm{g}$
5	1/2	1/2	$-\frac{1}{3}\,\mu_\mathrm{e} + \mu_\mathrm{g}$
6	1/2	3/2	$-\mu_\mathrm{e} + \mu_\mathrm{g}$

Tab. 1: Allowed hyperfine transitions of ^{57}Fe. m_{I_g} is the magnetic quantum number of the ground state and m_{I_e} of the excited state. The final column lists Equation 5 rearranged and evaluated for $I_\mathrm{g} = 1/2$, $I_\mathrm{e} = 3/2$ and the indicated m_I.

2.8. Gaussian, Lorentz and Voigt Functions

By fitting a Gaussian, a Lorentz and the convolution of both, also called a Voigt function onto a Mößbauer spectrum, different properties can be obtained. Choosing one over the other has different reasons. From the theory of atomic decay, a Lorentz (also called Cauchy) function is expected, due to it being the solution to a damped harmonic oscillator differential equation. Different effects induce additional homogeneous and inhomogeneous broadening of the linewidth like statistical fluctuations of the velocity of the sledge or temperature dependent lattice vibrations.
A Voigt function is a convolution of a Gaussian and a Lorentz function

$$f_\mathrm{Voigt}(x) = (G * L)(x) = \int G(\tau) L(x-\tau)\, d\tau,$$

but since this integral cannot be solved analytically, numerous different numerical approaches are possible. For example a superposition with a shaping parameter is called a pseudo-Voigt profile and was commonly used in the beginning of the age of computers [17]. Nowadays higher performing computers are available to the masses and new methods have been developed. One of the most common ways to obtain a Voigt function is by evaluating the real part of the Faddeeva function $\mathcal{F}$. In the analysis of this experiment, the following functions or sixfold versions of them are used as fit functions

$$\begin{aligned}
f_\mathrm{Gaussian}(x) &= -\frac{A}{\sqrt{2\pi}\,\sigma}\,\exp\left\{-0.5\left(\frac{x-\mu}{\sigma}\right)^2\right\} + B,\\
f_\mathrm{Lorentz}(x) &= -\frac{A}{\pi}\frac{\gamma}{(x-\mu)^2+\gamma^2} + B,\\
f_\mathrm{Voigt}(x) &= -\frac{A}{\sqrt{2\pi}\,\sigma}\,\mathrm{Re}\left\{\mathcal{F}\left(\frac{x-\mu+i\gamma}{\sqrt{2}\,\sigma}\right)\right\} + B.
\end{aligned}$$

All three function have a position parameter μ, which indicates the position of a peak, an pseudo-amplitude factor A and an offset parameter B. Note that A is not the actual amplitude of the function, but a factor either divided by π or $\sqrt{2\pi}\sigma$. Additionally the functions have width parameters σ, γ or both in the case of the Voigt function. Since the Voigt function has two width parameters, it is difficult to correctly distribute the actual width of a peak. Physically the parameter σ originating from the Gaussian function describes all processes, which cause a homogeneous linewidth broadening. The parameter

γ mostly describes the actual physical decay width, but also includes all inhomogeneous broadening effects. Without intrinsic information about the broadening effects, a fitting algorithm cannot accurately attribute the total width to the two width parameters.

3. Setup and Conduction of the Experiment

3.1. Setup

The setup used in this experiment is shown in Figure 5.

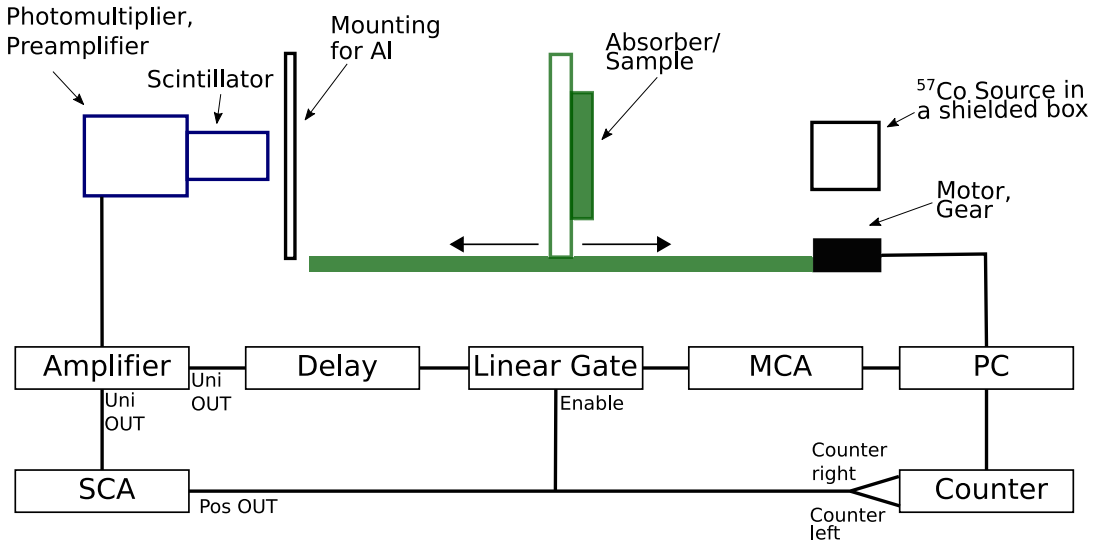


Fig. 5: Schematic setup of the experiment. The sample which is to be analyzed is mounted on a sledge that moves in one dimension.

The ^{57}Co source is placed inside a shielded box with a narrow opening, which directs the radiation towards the mounting in which a sample can be placed. The mounting can be moved with a motor in a velocity range from $0.01\,\mathrm{mm\,s^{-1}}$ to $10\,\mathrm{mm\,s^{-1}}$, towards or away from the source. It is controlled with a computer. The photons are detected by a thallium doped NaI scintillator with an optically coupled photomultiplier tube, which is operated with high bias voltages. In front of the detector there is a mount, which is used to hold aluminum plates for the measurement of the Compton background. The signal from the photomultiplier tube is amplified by a preamplifier. The signal gets further amplified and shaped by the mainamplifier. The unipolar output of this amplifier is split. One signal pathway is delayed by a delay unit for $3.25\,\mu\mathrm{s}$ and then fed into the input of a linear gate. The other signal is sent into a single channel analyzer (SCA) for discrimination. The output of the SCA is a logical yes which is emitted if the incoming signal is in a specified energy range. The signal of the SCA is used to count the number of signals directly via a counter, which is connected to the computer. The SCA signal is also used to enable the linear gate to let the delayed signal pass on to a multi channel analyzer (MCA). The MCA sorts the signals into channels, according to their intensity, which can be linked to the energy of the incident photon causing the signal. This energy spectrum is also measured in the computer.

Photon Detection

In this section the two core components of the setup, which are used to detect photons, the scintillator and the photomultiplier, are described in more detail.

A scintillator is a material, which exhibits scintillating properties, when irradiated with ionizing radiation. Incident photons excite the atoms of the scintillator which decay with

the emission of lower energy photons. Those photons are then detected with a photomultiplier tube PMT (or a photodiode, or a silicon based photomultiplier SiPM) optically coupled to the scintillator. Since often the scintillator and the PMT have a different geometry (i.e., cross section area, circle or square) a light guard is needed to guard the photons onto the detection surface of the PMT. Once a photon hits the photocathode of the PMT electrons are emitted due to the photoeffect. The electrons are accelerated by a bias voltage towards the first dynode. When they hit the first dynode secondary electrons are emitted which again are accelerated towards the next dynode by an higher bias voltage. This leads to an avalanche of electrons until the current is strong enough to be measured. This signal can then be related to the number of incident photons and their energy.

Scintillators are available in a variety of different shapes, materials and states of aggregation. They can be divided into organic or inorganic materials and gasses, liquids or solids. All show different characteristic behaviours with regard to their energy dependent resolution, linearity, time dependency, light yield, etc. The most common scintillators are NaI-crystals, which are also used in this experiment.
Obviously scintillators must be transparent to their own resonant photon-energies, which poses a technical difficulty. This problem can be solved by doping a different material into the crystals. In the case of NaI-crystals mostly thallium (TI) is used to activate the crystals. It introduces energy levels, which lie closely below the conduction band of the NaI-crystal and above the valence band. Excited atoms can decay onto those levels non-radiatively and then decay via emission of photons with an energy lower than the resonance energy of the NaI atoms. This doping can also be used to shift the energy of the scintillation photons into a frequency range, which coincides with the maximum sensitivity of the photomultiplier tubes. For most PMTs this is in the range of visible light, with a tendency to blue and ultraviolet.

There are many factors which have an influence on the statistics and resolutions. The energy resolution is directly proportional to the number of photons produced in the scintillator. The so called light yield L is defined as the number of photons emitted, when an incident particle looses a specific energy E in a certain length x of the crystal

$$\frac{dE}{dx} \propto \frac{dL}{dx}.$$

One would assume a Poisson distribution for this behaviour, which is mostly true. With this assumption it is easy to see, that materials that have a high light yield must have a high energy resolution. To reconstruct the energy of the incident particle correctly, the particle must loose all of its energy in the detection crystal. If that is the case, "the naive assumption of Poisson statistics is incorrect"[†], but it can be corrected for by introducing the Fano factor F. When the incident particle looses all of its energy in the crystal, the scintillation events are not independent of one another, since a definite number of energy is deposited and not a fluctuation amount, in the case of a particle only passing through the detector. The Fano factor describes this behaviour. It is material dependent and can be experimentally determined. For NaI it is approximately 1. In general the energy

[†]Quote from William R. Leo in [18].

dependent resolution R of a scintillator is calculated with

$$R = \frac{\Delta E}{E},$$

where ΔE, identified as the full width at half maximum of a peak is divided by its energy. With the relation between the FWHM of a Gaussian and its standard deviation σ and $J = E/w$, the number of ionizations, with E the deposited energy in the detector and w the mean energy required to ionize the material, the resolution results in

$$R = 2.35\,\frac{\sqrt{FJ}}{J} = 2.35\sqrt{\frac{Fw}{E}},$$

with the Fano factor F. In general a high light yield is wanted, since it increases the energy resolution of the detector. As described, the scintillator is optically coupled to a light guard, which has a collection and transmission efficiency, which feeds the photons into the PMT, which has a quantum efficiency. All this attributes to a loss in photons and therefore resolution.

3.2. Conduction

Assembly of the Setup

The setup was assembled as shown in Figure 5 and as described by the instructions [1]. To check for a proper signal pathway, the signals were displayed with an oscilloscope after each electronic component and compared with the expected curve forms from [1]. The amplification factor, shaping time and delays were adjusted.

Calibration of the MCA

To calibrate the used MCA an ^{241}Am source with a rotatable target wheel was used. With the wheel, different materials (Rb, Mo, Ag, Ba and Tb) with known literature values for their respective K_α decay energies were placed directly in front of the source. The scintillator was then used to obtain the different energy spectra.
A quick preliminary evaluation was performed to find a linear channel-energy relation to identify the 14.4 keV peak of the ^{57}Co source. With this the SCA-discriminator window is set for the rest of the experiment, such that only photons in the energy range of the peak width are detected.

Compton Background

To obtain the background counting rate caused by Compton scattering in the absorber material and the surrounding polymethyl methacrylate (acrylic/plexi glass) casing, aluminium shielding with gradually increasing widths were inserted in front of the detector and the counting rates were measured. This was performed for the two absorber materials used in this experiment to check, whether both yield the same amount of Compton scattering or not. The sledge was at rest. By fitting a double exponential function onto the data and extrapolating to a shielding width of 0 mm, the Compton background counting rate is obtained and used to correct the measured counting rates in the rest of the experiment.

Attenuation of Gamma Radiation by Acrylic Glass

To obtain the attenuation of the photons passing through the acrylic glass casing, the counting rates with and without acrylic glass in the radiations path were measured. Also the theoretical expected attenuation was calculated. The measured attenuation coefficient is then used to correct the measured counting rates in the rest of the experiment.

Sledge Velocity

In order to check, whether the velocity of the sledge is in agreement with the velocity set on the computer, different velocities were determined. This was done by measuring the time and distance the sledge moved at preset velocities. For this a stopwatch and a standard ruler were used.

Mößbauer Spectroscopy

Two different absorber materials were investigated. A stainless steel and natural iron absorber. They were separately placed on the sledge. In the used `LabView` software, start, stop and step velocity, as well as measuring time were adjusted. With this the counting rates at different velocities were measured and Mößbauer spectra obtained. From the absorption spectra different properties like the isomeric shift, the Debye-Waller factor and the lifetime τ of the excited states are calculated.

4. Analysis

4.1. Uncertainty Considerations

In the following the uncertainties on measured counts N are calculated with $s_N = \sqrt{N}$, since counts follow a Poisson distribution. For comparability reasons, counts are always converted into counting rates or simply rates, $\dot{N} = N/t$. This means, that counts are normalized with their respective measurement time t to 1 s. Their uncertainties are calculated as $s_{\dot{N}} = \sqrt{N}/t$. When measuring the counts of the same process in different measurement series, the additive behaviour of Poisson distributed values is used, i.e., the counts and measurement times are simply summed up.
When multiple values for the same quantity are measured or calculated, the weighted mean is calculated with the inverse square of the uncertainties as the weights for each value.
Values obtained directly from fitting functions with the weighted least square reduction method onto measured data posses uncertainties, which are derived from the square root of their respective diagonal element in the covariance matrix.
As a quantification of the quality of a fit, the reduced chi-square statistic χ^2_ν is used, in which a value in close proximity to 1 indicates a good fit of data to the model-function. If necessary a residual plot, which shows the deviations from the data to the fit function value, is given to further show the quality of the fit.
Using equations which contain values with uncertainties, standard Gaussian error propagation is applied. For most propagations the exact formula is not stated due to triviality and are left for the reader as an exercise. The propagations follow Equation 16 for not correlated and Equation 17 for correlated parameters, displayed in the appendix. The propagations of more complex functions are explicitly stated.
Sometimes values are obtained by projecting/marking an x-value in a plot and reading off the y-value. Here the uncertainties of the x-value are also projected onto the y-axis in the same manner. An example of this is Figure 14 or Figure 17. To avoid asymmetric uncertainties always the bigger resulting uncertainty is used as the symmetric standard deviation statistic σ. To obtain values this way, the graph is loaded into `Inkscape` to draw rectangular lines and the pixel coordinate system is used to determine the resulting values. This method yields read-off uncertainties always smaller than the linewidths of the used graphs. Therefore no read-off uncertainties are taken into account and only projected or uncertainties caused by linewidths are used.

4.2. Setup Check

First every component in the signals pathway is considered and its in and outputs displayed on an oscilloscope. Here only the most important signals will be discussed.
Figure 7 shows the output signal of the preamplifier. Its exponential decay is expected, since it originates from the discharge of a capacitor. The signal is then amplified by an amplifier. Its unipolar output is split. One signal is delayed by 3.25 µs with a delay unit. The signals before and after the delay unit are shown in Figure 7. As expected the output signal of the amplifier is of Gaussian shape, since the amplifier not only amplifies, but also shapes the signals with an adjustable shaping time. The other unipolar output signal of the amplifier is processed by the SCA. As can be seen in Figure 8, the SCA output signal is a logical yes of rectangular shape and predefined height and duration. It is emitted if the intensity of the incoming signal is of a defined height. The signal of the SCA opens

the linear gate which then lets the delayed amplifier signal pass through. This is shown in Figure 9. The delayed signal is cut at both sides, which is caused by the linear gate opening and closing. However, this is not a problem, as the peak lies inside this window and the loss in integrated intensity will be corrected for anyway by the calibration of the MCA. Since the linear gate opens in such a way that the delayed unipolar signal passes, the delay is chosen well and the setup can be used for measurements.

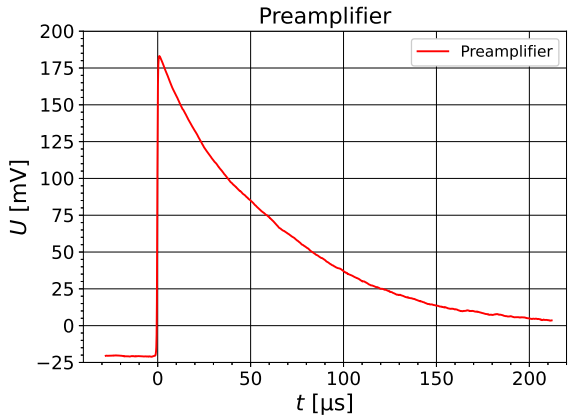


Fig. 6: Output signal of the preamplifier. The exponential decay is caused by the output of the photomultiplier after detection of one or more photons.

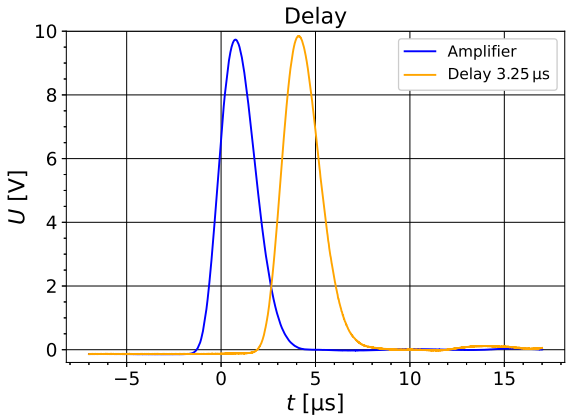


Fig. 7: The signal of the amplifier (blue) is delayed by the delay unit (orange) by 3.25 µs.

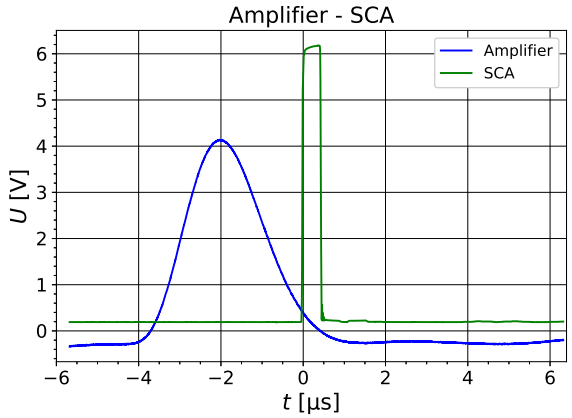


Fig. 8: The output signal of the amplifier (blue) triggers a logical yes as an output signal of the SCA (green).

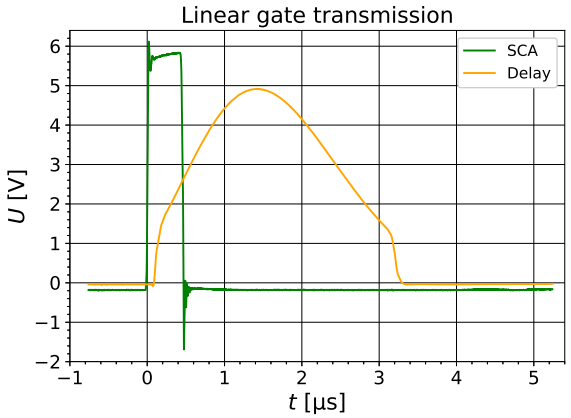


Fig. 9: The delayed signal of the amplifier (orange) passes the linear gate which is opened by the logical yes signal of the SCA (green).

4.3. Calibration of the MCA

As described in Section 3.2, different materials with known transition energies are placed in front of an ^{241}Am source and their spectra are recorded to calibrate the used MCA. The measured spectra are displayed in the appendix in Figure 20−24.
To acquire the positions of the K_α decay peaks, for which the energy values are listed in

the instructions [1], they are fitted with Gaussian functions of the form

$$f(x) = \frac{A}{\sqrt{2\pi}\,\sigma} \exp\left\{-0.5\left(\frac{x-\mu}{\sigma}\right)^2\right\} + B,$$

with A the amplitude, B the offset, μ the position and σ the width fit parameter. The parameters for each spectrum are displayed in the appendix in Table 14.
The parameters μ are used as the positions and the width of the peaks σ as their uncertainties in the unit of channels. Comparing these with the known energy values a linear relation between the two is found, displayed and fitted with a linear function in Figure 10.

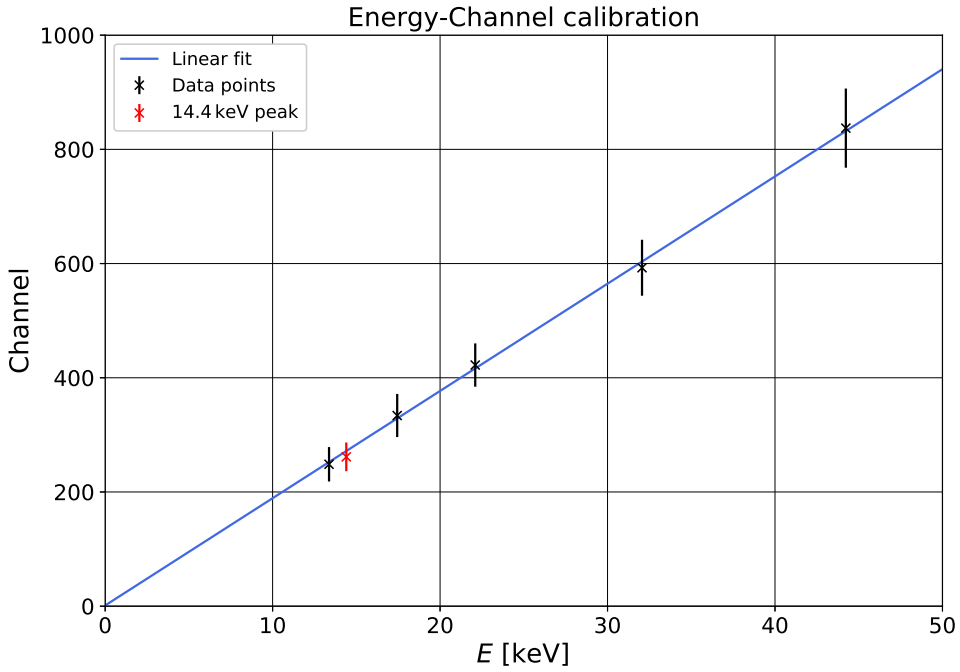


Fig. 10: Calibration of channel and energy. The black points are the K_α peaks obtained from Gaussian fits in Figure 20−24. The red point indicates the 14.4 keV peak obtained from a Gaussian fit onto the peak in the ^{57}Co source spectrum displayed in Figure 11. The energy value is set to 14.4 keV, while the channel value is the position parameter from the fit with the width parameter as its uncertainty.

The calibration function is found to be

$$f(E) = (18.8 \pm 0.4)\,\mathrm{keV}^{-1} \cdot E + (1.2 \pm 0.9). \tag{6}$$

With this the 14.4 keV peak in the source spectrum can be identified as shown in Figure 11.

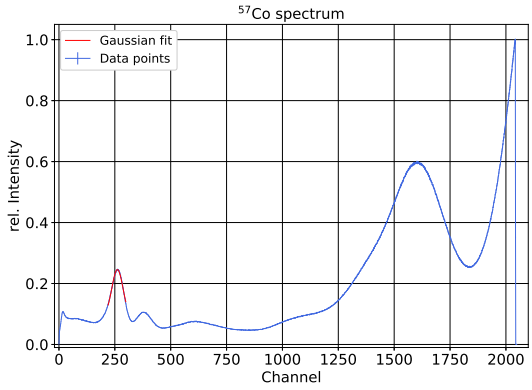


Fig. 11: Spectrum of the used ^{57}Co source. The red Gaussian does not indicate the set window of the SCA, but the data used for the fit. The fit parameters are stated in Table 14 in the appendix.

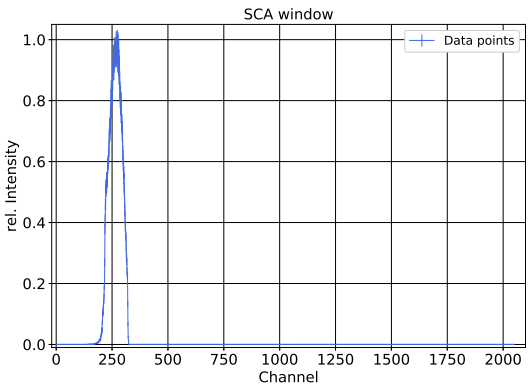


Fig. 12: Effect of the set SCA window on the ^{57}Co source spectrum. Only photons in the energy region of the 14.4 keV peak are let through the linear gate.

With Equation 6 rearranged the fitted peak in Figure 11 has an energy of (13.9 ± 1.3) keV, which lies within $< 1\,\sigma$ to the expected value of 14.4 keV [1]. It holds a relative uncertainty of 9.4 %, which is caused by the detectors resolution.
The SCA-discriminator window is set accordingly to the identified peak for the rest of the experiment so that only photons in the range of the 14.4 keV peak width are measured. The effect of the set window is shown in Figure 12.
The red data point in Figure 10 indicates the channel position of the 14.4 keV peak obtained from the Gaussian fit in Figure 11. The point lies within $1\,\sigma$ to the value expected from the calibration indicated by the straight line and is therefore in good agreement.

Additionally to the fit parameters, Table 14 also shows the energy dependent resolution of the detector calculated with σ/μ [19]. All values are in the order of 10 %, which is typical for such a scintillator in the energy region of keV [19]. The resolution depends on the set shaping time, noise in the used electronics and also the temperature of the scintillation crystal. The resolution shows that the method of Mößbauer spectroscopy is an incredible useful tool, as it can resolve 1 in 10^{12} [11] for the studied 14.4 keV transition with the same scintillation crystal.

4.4. Compton Background

The measured counting rates for the Mößbauer spectra have to be corrected for background events. The most dominant contribution results from Compton scattering of the 122 keV and 136 keV photons from the Cobalt source, which lose energy in the absorber and especially in the acrylic glass casing around the absorber. Their energies are shifted down into the SCA energy-window and distort the measured rates. Therefore measurements have to be performed to quantify the additional counting rates caused by Compton scattering and correct all measured rates.
For this aluminium plates with increasing thickness d are used as shielding in front of the detector. From the instructions [1] it is known that aluminium has a higher transmission for the 122 keV and 136 keV photons than the 14.4 keV photons. The aluminium absorbs

nearly all of the 14.4 keV photons with sufficient shielding thickness and therefore all measured photons are assumed to be Compton-shifted.
From the theory of Compton scattering it is expected, that most of the Compton scattering is caused by the absorber materials. If the absorber and the acrylic glass would be of the same thickness, the denser absorber material with an higher atomic number would be the main cause of Compton scattering. But since the absorber material only has a thickness of around 1.3 % the size of the acrylic glass, the acrylic glass is assumed to yield the most dominant contribution to the Compton background rate. To study if there is a difference between the two materials, two identical measurements were performed, one for the stainless steel (S) and one for the natural iron (N) absorber. A subsequent measurement using only acrylic glass was not performed due to time constraints, but would have been interesting, as it could confirm that the acrylic glass is the main cause of the Compton background rate. However, this is not crucial for our measurements, since the Compton background is determined with the absorber materials and the acrylic glass together, so even if the materials would provide a significant contribution, it would be included in the measured values.
The counting rates with increasing shielding width d for the stainless steel absorber is shown in Figure 13 and the natural iron absorber is displayed in the appendix in Figure 27.

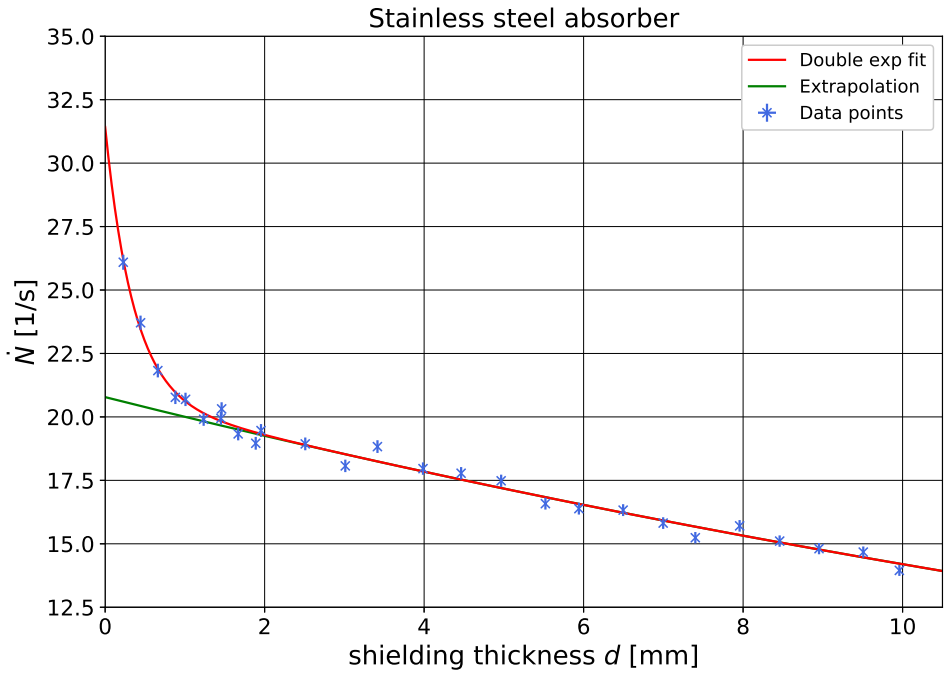


Fig. 13: Counting rate in dependence of the aluminium shielding thickness d for the stainless steel absorber. The red curve shows the double exponential fit, while the green curve indicates the extrapolation of the exponential decay, caused by the high energetic photons to no shielding, which is identified as the Compton background rate.

From the theory a double exponential decay is expected and therefore a function of the form

$$\dot{N}(x) = A \cdot \exp(ax) + B \cdot \exp(bx)$$

is fitted onto the data. The optimal fit parameters are listed in Table 15 in the appendix. The first exponential decay describes the attenuation of the low energetic photons and the second describes the high energetic photons. The second exponential decay is extrapolated to a thickness of 0 mm shielding. This corresponds to the fit parameter B. In this way the background rates with no shielding, caused by the high energetic photons, which are shifted down by Compton scattering into the set energy region, are determined to be

$$\dot{N}^{(\mathrm{S})}_{\mathrm{Compton}} = (20.8 \pm 0.2)\,\mathrm{s}^{-1},$$
$$\dot{N}^{(\mathrm{N})}_{\mathrm{Compton}} = (20.2 \pm 0.2)\,\mathrm{s}^{-1}.$$

Since both values deviate only by $2.1\,\sigma$ with relative uncertainties smaller 1 % and no significant deviation caused by the different materials is expected from theoretical considerations the mean of both values is calculated

$$\dot{N}_{\mathrm{Compton}} = (20.5 \pm 0.3)\,\mathrm{s}^{-1}$$

and used in the following evaluations to correct measured counting rates.
Extrapolating the exponential decay of the 14.4 keV counting rate yields that for the stainless steel absorber the counting rate drops to around $0.1\,\mathrm{s}^{-1}$ with ~ 1.7 mm shielding. After ~ 3.3 mm it drops to around $0.001\,\mathrm{s}^{-1}$. This validates the chosen shielding thickness range, as it suffices to describe the attenuation of the high energetic photons.

4.5. Attenuation of Gamma Radiation by Acrylic Glass

The absorber materials are encased by acrylic glass of ~ 2 mm thickness to fixate the absorber materials, which are only of µm thickness. This additional material causes extra attenuation of photons and has to be quantified. Two ways of determining the attenuation factor are performed and their results compared.
One way is to determine the counting rates with and without acrylic glass in the radiations path

$$\dot{N}_{\text{acrylic glass}} = (61.6 \pm 0.3)\,\mathrm{s}^{-1},$$
$$\dot{N}_0 = (73.9 \pm 0.4)\,\mathrm{s}^{-1},$$

and dividing both to determine the measured transmission

$$T_{\mathrm{meas}} = (83.4 \pm 0.6)\,\%.$$

For this an acrylic glass plate similar to the one holding the absorber is used.
Obviously $\dot{N}_{\text{acrylic glass}}$ has to be Compton corrected for, since the Compton background is mainly produced by the acrylic glass. The corrected attenuation factor therefore results in

$$T_{\mathrm{cor}} = \frac{\dot{N}_{\text{acrylic glass}} - \dot{N}_{\mathrm{Compton}}}{\dot{N}_0} = (55.6 \pm 0.7)\,\%.$$

The corrected transmission can then be used to calculate the mass-attenuation coefficient μ/ρ as described in Section 2.2.4 with Equation 2 and it results in

$$\mu/\rho_{\mathrm{meas}} = (2.49 \pm 0.06)\,\frac{\mathrm{cm}^2}{\mathrm{g}},$$

with the diameter d and ρ the density of the acrylic glass

$$d = (1.98 \pm 0.02)\,\text{mm}, \qquad \rho = 1.19\,\frac{\text{g}}{\text{cm}^3}.$$

While ρ is taken from [1], d is obtained by measuring the width of the acrylic glass multiple times with a micrometer screw and then averaging it.

Another way to determine the transmission ratio is to use a literature value for μ/ρ and calculate the expected transmission factor. The value of μ/ρ can be read off from Figure 14, where the 14.4 keV is marked at the x-axis in the illustration and mapped onto the y-axis. The uncertainty results from the linewidth of the curve and the energy uncertainty caused by drawing into the plot and then projecting that energy range onto the y-axis.

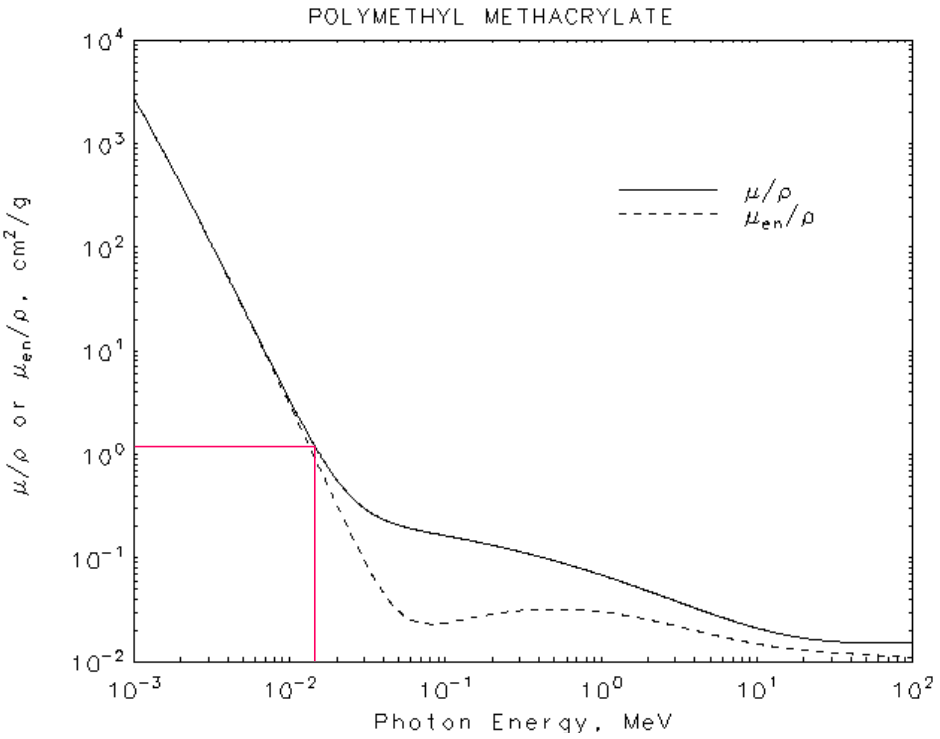


Fig. 14: Mass-attenuation coefficient for acrylic glass for different energies, modified from [20] to obtain μ/ρ for 14.4 keV.

The expected mass-attenuation is

$$\mu/\rho_{\text{expected}}\,(14.4\,\text{keV}) = (1.2 \pm 0.2)\,\frac{\text{cm}^2}{\text{g}},$$

with a relative uncertainty of 16.7 %. With this and Equation 1 the expected attenuation can be calculated to

$$T_{\text{expected}} = (75 \pm 4)\,\%,$$

with a relative uncertainty of 5.3 %.

The values for the mass-attenuation deviate by $6.2\,\sigma$ and the attenuation factors deviate by $4.8\,\sigma$. Clearly the values are not in agreement. The deviations could be caused by impurities in the used acrylic glass, changing the density and effective atomic number. Therefore the corrected, experimental attenuation factor is used to further correct the counting rates, as described in Section 4.7. It describes the setup more accurately, as it does not depend on some literature values, but the used materials in the experiment.

4.6. Velocity of the Sledge

The velocity of the sledge is measured and compared to the set velocity in the software. The results are shown in Figure 15. A linear relation is found and a linear function fitted onto the data.

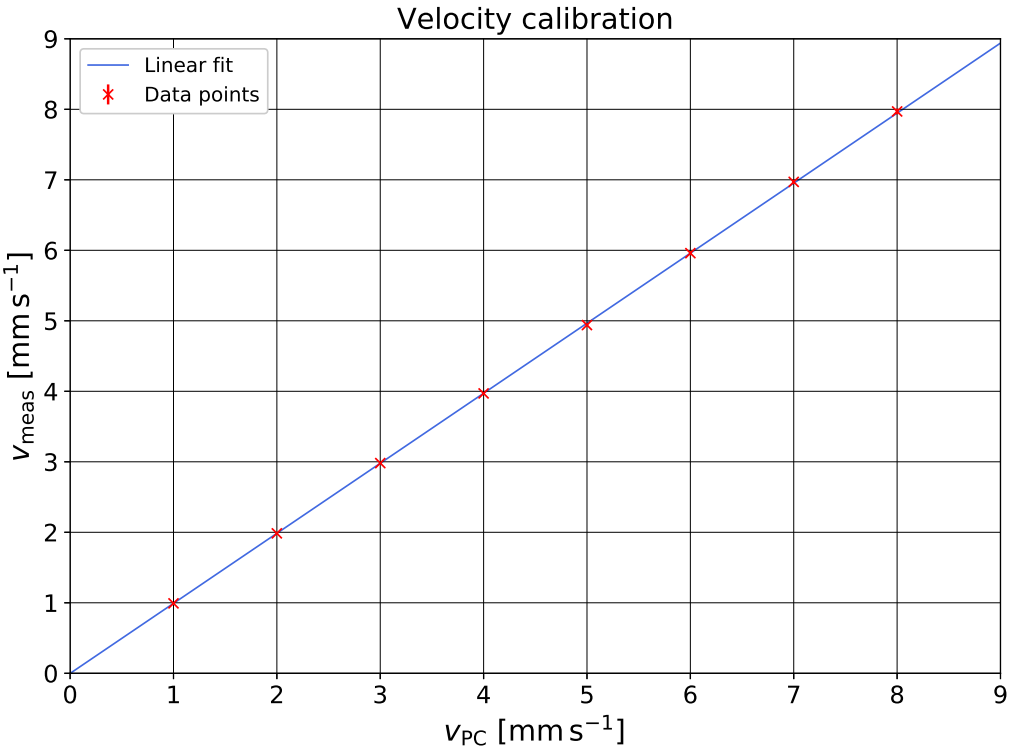


Fig. 15: Measured sledge velocity in dependence of the set velocity in the motor control software. The blue line indicates the linear fit function. The uncertainty on the measured velocities are too small to be visible, as they are in the order of $0.04\,\mathrm{mm\,s^{-1}}$ or smaller.

The fit function results in

$$v_{\mathrm{meas}}(v_{\mathrm{PC}}) = (0.994 \pm 0.002) \cdot v_{\mathrm{PC}} - (0.002 \pm 0.003)\,\mathrm{mm\,s^{-1}},$$

with a $\chi^2_\nu = 0.41$. Since the slope deviates from 1 only by 0.6 % and the offset lies within a $1\,\sigma$ range of 0, no significant deviation from the set velocity is found within the measurement methods resolution. This is dominantly influenced by human reaction time of the experimenters, since a simple stopwatch app was used to determine the time it

takes the sledge to move a certain distance, which was individually determined for each measurement. At velocities smaller $3\,\mathrm{mm\,s^{-1}}$ the uncertainty on the distance is as low as the normal read-off uncertainty on a standard triangle ruler 0.1 mm. As the human reaction time, $s_t = 0.3\,\mathrm{s}$ is chosen. The relative uncertainties of the measured velocities rise with increasing velocity from 0.4 % to 3 %. Only up to that resolution a statement about the velocity can be made. In the following evaluations these uncertainties are ignored, as they only indicate a crude upper limit, which is only dependent on the used measurement technique and does not hold intrinsic information about the real uncertainties.

As will be explained later in Section 4.8.4, statistical fluctuations of the seldges velocity limits the resolution of the Mößbauer spectroscopy and has a big influence on the measured linewidths. This is also explained in more detail in [21].

4.7. Rate Correction

The measured Compton counting rate $\dot{N}_{\mathrm{Compton}}$ from Section 4.4 and the corrected attenuation factor T_{cor} from Section 4.5 are used to correct all measured counting rates. First the measured rates are cleansed from Compton background and then the rate is corrected for by the corrected attenuation factor with

$$\dot{N}_{\mathrm{cor}} = \frac{\dot{N}_{\mathrm{meas}} - \dot{N}_{\mathrm{Compton}}}{T_{\mathrm{cor}}}.$$

With this correction the two most dominant background rates are corrected for. The uncertainty on the corrected rate depends on the uncertainty of the original rate and the uncertainties of the Compton rate and the attenuation factor. Both corrections could be improved by means of longer measurements, but already have relative uncertainties smaller 1.5 %. The relative uncertainties of the rates before correction are in the range of 1 %. After correction the rates yield relative uncertainties of around 3.5 %. This questions the appropriateness of the corrections made, since at first glance it only increases the uncertainties. In the context of the following evaluations however, the counting rate at, for example the peak position is required to calculate the Debye-Waller factor f_{Q} of the used source. Therefore a correction of the measured rates is unavoidable and not performing one would certainly yield deviations from the real values and no usable results.

4.8. Stainless Steel Absorber

The stainless steel absorber sample is placed on the sledge and the measured Mößbauer spectrum is shown in Figure 16. The spectrum is measured in a velocity range from $-2\,\mathrm{mm\,s^{-1}}$ to $2\,\mathrm{mm\,s^{-1}}$ for different measurement times t. The negative velocity values result from the sledge moving towards the source, while positive values mean the opposite. Therefore negative velocity values indicate an increase in the observed energy by the absorber. The data was collected over a period of 3 days during which the weather and thus the temperature in the laboratory was very stable.

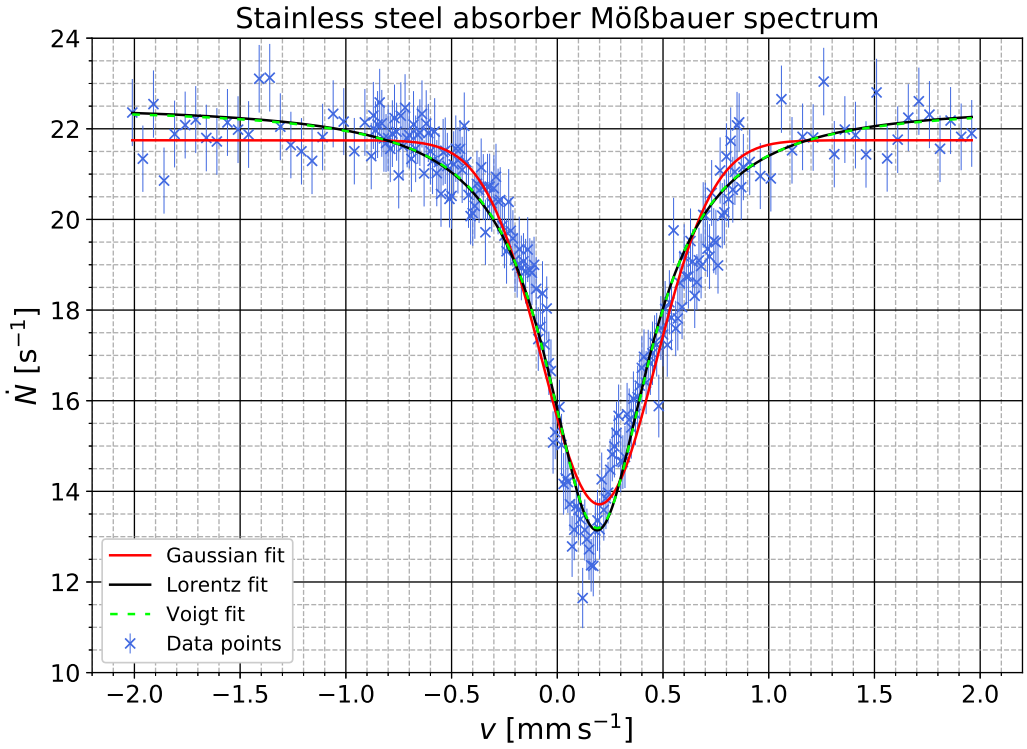


Fig. 16: Mößbauer spectrum of the stainless steel absorber sample. The data points are shown in royal blue with errorbars only on the rate $\dot{N}$, since the uncertainty of the velocity could not be quantified. The red curve shows the Gaussian, the black curve the Lorentz and the lime green dashed curve the Voigt fit function. The optimal fit parameters and the reduced χ^2 are listed in Table 2.

A Gaussian, Lorentz and Voigt function are fitted onto the data. The fit parameters are displayed in Table 2. The reduced χ^2 close to 1 indicate a good agreement between the collected data and the assumed models fitted upon it.
The Lorentz and Voigt fit have a $\chi^2_\nu < 1$ indicating the presence of noise or overestimation of uncertainties. But since they are so close to 1 no serious concerns arise on the quality of the data or the correctness of the models. It has to be noted, that the Lorentz and Voigt fit yield identical χ^2_ν up to two decimal places, which is not surprising, since the Voigt fit very closely follows the Lorentz curve. The reasons behind this are discussed in more detail in Section 4.8.4.
The Gaussian fit yields a $\chi^2_\nu > 1$ indicating that the assumed model of a Gaussian distribution is not correctly describing the collected data. This is to no surprise, as it is

expected from the theory to find Lorentz or Voigt shapes. But since the deviation from 1 is not too extreme the Gaussian should still yield results that do not deviate by much from the expected values.

Function	μ [mm s^{-1}]	σ [mm s^{-1}]	γ [mm s^{-1}]	A [mm s^{-2}]	B [s^{-1}]	χ^2_ν
Gaussian	0.199 ± 0.005	0.270 ± 0.006	-	5.44 ± 0.14	21.75 ± 0.08	1.09
Lorentz	0.188 ± 0.004	-	0.298 ± 0.009	8.8 ± 0.3	22.52 ± 0.10	0.93
Voigt	0.189 ± 0.004	0.05 ± 0.06	0.29 ± 0.03	8.6 ± 0.5	22.48 ± 0.14	0.93

Tab. 2: Fit parameters for the Gaussian, Lorentz and Voigt fit for the stainless steel absorber.

4.8.1. Isomeric Shift E_{Iso}

From the position parameter μ of each fit function the position of the minimum is determined. Its offset from zero is called the isomeric shift (see Section 2.6).
With the Doppler relation

$$E = E_\gamma \frac{v}{c}, \tag{7}$$

the velocity is transformed into the isomeric shifts

$$\begin{aligned} E_{\text{Iso, Gaussian}} &= (9.5 \pm 0.2)\,\text{neV}, \\ E_{\text{Iso, Lorentz}} &= (9.0 \pm 0.2)\,\text{neV}, \\ E_{\text{Iso, Voigt}} &= (9.1 \pm 0.2)\,\text{neV}. \end{aligned}$$

They all coincide within 1 or $2\,\sigma$ and are therefore compatible. No literature value for this sample is known, therefore no comparison can be made, but the values all lie in the expected energy order neV. All three values hold relative uncertainties smaller 2.3 %. The uncertainty depends directly on the quality of the fits and could only be improved by means of more data or longer measurement times to reduce the uncertainty on the data points.

4.8.2. Effective Absorber Thickness T_{A}

To calculate the fraction of recoilless resonance emission f_{Q} in the source the dimensionless ancillary quantity T_{A} is required. It depends on the Debye-Waller factor of the absorber $f_{\text{A}}(20\,^\circ\text{C}) = 0.8$ [1] and is described by

$$T_{\text{A}} = f_{\text{A}} n_{\text{A}} \beta \sigma_0 d_{\text{A}}, \tag{8}$$

with the absorber thickness $d_{\text{A}} = 25\,\mu\text{m}$ [1], the fraction of ^{57}Fe in the isotope mixture $\beta = 0.022$ [1], n_{A} the number of iron atoms per cm^3 in the absorber and the resonant absorption cross section σ_0.
The resonant absorption cross section σ_0 is calculated using

$$\sigma_0 = \frac{\lambda^2}{2\pi} \frac{2I_{\text{e}} + 1}{2I_{\text{g}} + 1} \frac{1}{1 + \alpha} \tag{9}$$

from [22], where $I_e = 3/2$, $I_g = 1/2$ are the spin quantum numbers for the excited and ground state, $\alpha = 8.58 \pm 0.18$ [23] is the internal-conversion coefficient and λ the resonant wavelength corresponding to the $E_\gamma = 14.4\,\mathrm{keV}$ photons with

$$\lambda = \frac{hc}{E_\gamma} = 0.0861\,\mathrm{nm}.$$

With this, σ_0 results in

$$\sigma_0 = (246 \pm 5) \cdot 10^{-24}\,\mathrm{m}.$$

n_A is calculated using

$$n_\mathrm{A} = \rho\,\frac{N_\mathrm{A}}{M}\,f,$$

with the molar mass of iron $M = (55.845 \pm 0.002)\,\mathrm{g\,mol^{-1}}$ [24], the density $\rho = 7.874\,\mathrm{g\,cm^{-3}}$ [25], the iron content in the absorber $f = (70 \pm 5)\,\%$ [1] and the Avogadro constant $N_A = 6.022\,140\,76 \cdot 10^{23}\,\mathrm{mol^{-1}}$ [26]. It results in

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

which leads to an effective absorber thickness of

$$T_\mathrm{A} = 6.4 \pm 0.5.$$

No literature value is known for comparison, but the value holds a relative uncertainty of 7.8 %. This is mainly caused by the high uncertainty on the iron content f and could therefore not be improved.

4.8.3. Debye-Waller Factor of the Source f_Q

Finally, the Debye-Waller factor of the source can be calculated using the formulas

$$f_\mathrm{Q} = \frac{\dot{N}(\infty) - \dot{N}(\mu)}{\dot{N}(\infty)\,[1 - \exp(-T_\mathrm{A}/2)\,\mathrm{J}_0(iT_\mathrm{A}/2)]} \tag{10}$$

and

$$\begin{aligned} s_{f_\mathrm{Q}} = \Bigg\{ &\left(-\frac{\dot{N}(\mu) s_{\dot{N}(\infty)}}{\dot{N}^2(\infty)\,[1 - \exp(-T_\mathrm{A}/2)\,\mathrm{J}_0(iT_\mathrm{A}/2)]} \right)^2 \\ &+ \left(-\frac{s_{\dot{N}(\mu)}}{\dot{N}(\infty)\,[1 - \exp(-T_\mathrm{A}/2)\,\mathrm{J}_0(iT_\mathrm{A}/2)]} \right)^2 \\ &+ \left(-\frac{\exp(-T_\mathrm{A}/2)\;[\mathrm{J}_0(iT_\mathrm{A}/2) + i\mathrm{J}_1(iT_\mathrm{A}/2)]}{2\,[\exp(-T_\mathrm{A}/2) - \mathrm{J}_0(iT_\mathrm{A}/2)]} \right)^2 \Bigg\}^{1/2} \end{aligned}$$

from [27], with T_A the effective absorber thickness calculated in Section 4.8.2 and J_0 and J_1 the Bessel functions of the first kind in the 0^{th} and 1^{st} order. $\dot{N}(\infty)$ is the rate at infinite sledge velocity and is described by the offset parameter in the fit functions. The uncertainty arises from the square root of the corresponding diagonal element of the

covariance matrix. $\dot{N}(\mu)$ results from evaluating the fit functions at their minima. The uncertainties are derived from the projections of the minima position uncertainties onto the y-axis using their respective fit function. Studying Equation 10 in more detail the dependence of f_{Q} on T_{A} shows, that with an increase of the effective absorber thickness the formula approaches

$$f_{\mathrm{Q}} \approx \frac{\dot{N}(\infty) - \dot{N}(\mu)}{\dot{N}(\infty)}. \tag{11}$$

This is also the intuitive understanding of the Debye-Waller factor of the source, as for one it does not depend on the used absorber thickness and it directly calculates the relative number of recoilless emitted and absorbed photons. Therefore a higher T_{A} is desirable, but as stated in [27] and [28], a T_{A} above 10 is not properly described by a Lorentz curve and would therefore induce additional deviations, when using a Lorentz or Voigt function to obtain parameters like the isomeric shift, or the lifetime of the state.

The Debye-Waller factors result in

$$\begin{aligned} f_{\mathrm{Q,\,Gaussian}} &= 0.482 \pm 0.003, \\ f_{\mathrm{Q,\,Lorentz}} &= 0.543 \pm 0.003, \\ f_{\mathrm{Q,\,Voigt}} &= 0.539 \pm 0.005. \end{aligned}$$

Since no literature values are known for this quantity, only a comparison between the three fit functions is possible. The factors calculated from the Lorentz and Voigt fit coincide within a $1\,\sigma$ range, but the factor from the Gaussian fit deviates strongly from the two with $\sim 10\,\sigma$. This is to no surprise, since it can clearly be seen in Figure 16 that $\dot{N}(\mu)$ and $\dot{N}(\infty)$ differ significantly for the Gaussian fit, while the Voigt fit essentially collapses into a Lorentz function and very closely follows the Lorentz fit.
A fraction of around 50 % seems reasonable, if not somewhat low for a source specifically purchased for an experiment on recoilless emission.

4.8.4. Linewidth Γ and Lifetime τ of the 14.4 keV State in ^{57}Fe

To obtain the measured linewidths of the absorption peaks and use them to calculate the lifetime of the excited 14.4 keV state as explained in Section 2.3, the widths of the fitted functions have to be determined. For this the full width at half maximum (FWHM) is used. The FWHM of a Gaussian is related to its standard deviation σ with

$$\mathrm{FWHM}_{\mathrm{Gaussian}} = \Gamma_{\mathrm{Gaussian}} = 2\sqrt{2\,\ln(2)}\,\sigma_{\mathrm{Gaussian}}.$$

The FWHM of a Lorentz function is

$$\mathrm{FWHM}_{\mathrm{Lorentz}} = \Gamma_{\mathrm{Lorentz}} = 2\gamma_{\mathrm{Lorentz}}.$$

The FWHM of a Voigt function has no analytical equation, but an approximation with deviations only as high as 0.023 % is

$$\mathrm{FWHM}_{\mathrm{Voigt}} \approx 0.5346\,\gamma_{\mathrm{Voigt}} + \sqrt{0.2166\,\gamma_{\mathrm{Voigt}}^2 + \sigma_{\mathrm{Voigt}}^2},$$

according to [29], but since the sole purpose of the Voigt fit, as a convolution of a Gaussian and a Lorentz function, is to filter out homogeneous broadening effects (like resolution,

noise, etc.), which have no relevance to the actual lifetime of the excited state, only the Lorentz part is used for the linewidth

$$\Gamma_{\text{Voigt}} = 2\gamma_{\text{Voigt}}.$$

The measured linewidths Γ_{meas} result in

$$\begin{aligned}\Gamma_{\text{meas, Gaussian}} &= (0.635 \pm 0.014)\,\text{mm}\,\text{s}^{-1},\\ \Gamma_{\text{meas, Lorentz}} &= (0.60 \pm 0.02)\,\text{mm}\,\text{s}^{-1},\\ \Gamma_{\text{meas, Voigt}} &= (0.57 \pm 0.06)\,\text{mm}\,\text{s}^{-1}.\end{aligned}$$

Analogous to Section 4.8.1, using Equation 7 the values are converted into neV and result in

$$\begin{aligned}\Gamma_{\text{meas, Gaussian}} &= (30.5 \pm 0.7)\,\text{neV},\\ \Gamma_{\text{meas, Lorentz}} &= (28.6 \pm 0.9)\,\text{neV},\\ \Gamma_{\text{meas, Voigt}} &= (27 \pm 3)\,\text{neV}.\end{aligned}$$

Due to the time-energy-uncertainty relation, as described in Section 2.3, the linewidths correspond to the lifetime of the excited 14.4 keV state and result in

$$\begin{aligned}\tau_{\text{Gaussian}} &= (21.6 \pm 0.5)\,\text{ns},\\ \tau_{\text{Lorentz}} &= (23.0 \pm 0.7)\,\text{ns},\\ \tau_{\text{Voigt}} &= (24 \pm 3)\,\text{ns}.\end{aligned}$$

Compared to the literature value

$$\tau_{\text{lit}} = 141\,\text{ns}$$

from [1], all three values show enormous deviations. All are below the literature value, implying that the measured linewidths are broader than the natural widths. There are multiple reasons for that and some are studied in great detail in [27] and [28]. One reason for the broadening is the overlap of emission and absorption spectra in the absorber and the source respectively. This causes the measured linewidths to be at least twice as big as the natural linewidth.
Going into more detail, a dependency of the finite thickness of the absorber and the source is found. The effective absorber thickness T_{A}, calculated in Section 4.8.2, has the biggest impact in the following consideration, as can be seen in Figure 17. Additionally the effective source thickness has to be calculated as well. This is done with

$$T_{\text{Q}} = f_{\text{Q}} n_{\text{Q}} \beta \sigma_0 d_{\text{Q}}$$

from the instructions [1], with $\beta = 1$, $n_{\text{Q}} \approx n_{\text{A}}$, σ_0 from Equation 9, $d_{\text{Q}} \approx 100\,\text{Å}$, stated as $\mathcal{O}(100\,\text{Å})$ in the instructions and the Debye-Waller factors from Section 4.8.3 for the different fit functions and results in

$$\begin{aligned}T_{\text{Q, Gaussian}} &= 0.071 \pm 0.003,\\ T_{\text{Q, Lorentz}} &= 0.080 \pm 0.003,\\ T_{\text{Q, Voigt}} &= 0.079 \pm 0.005.\end{aligned}$$

Using Figure 17, the relative line broadening is determined. It indicates the ratio between apparent/measured Γ_a and natural linewidth Γ_{nat}. Figure 17 shows the relative broadening in dependence of the effective absorber thickness T_A for different T_Q for a uniform source distribution modified from [27]. It has to be noted that for $T_A = T_Q = 0$ the relative line broadening is also 2. This is caused by the overlap of emission and absorption spectra as explained above. T_A and T_Q lie between 0 and 10, since the absorption and emission spectra in that range show to a very good degree of approximation a Lorentz curve [28]. Since a Gaussian source distribution would only yield a slightly different value for the relative broadening and there was no indication to suspect the source to be not uniformly distributed, no detailed considerations of the source distribution were performed (see [27] page 136 for Figure 17 with a Gaussian source distribution).
Since the different values of T_Q differ only in a range of less than one pixel in the shown image the value $T_Q = 0.08$ is used, which translates to ~ 4 pixel. Furthermore the $T_Q = 0$ line shown has a width of ~ 5 pixel[†]. Therefore the upper edge of the line is used for the chosen T_Q value.

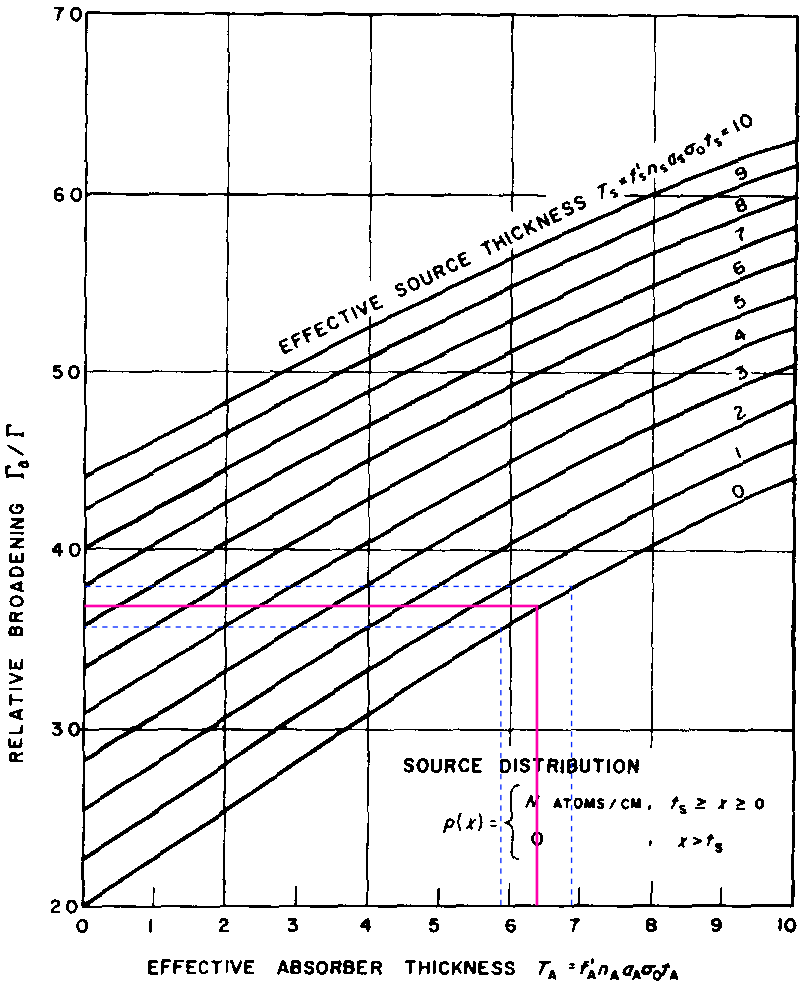


Fig. 17: Relative line broadening Γ_a/Γ_{nat} in dependence of the effective absorber thickness T_A for effective source thicknesses T_Q between 0 and 10, modified from [27]. The purple line and the blue dashes indicate the calculated effective absorber thickness value $T_A = 6.4 \pm 0.5$ from Section 4.8.2.

[†]The pixel values are determined using the `Inkscape` pixel coordinate system.

From Figure 17 the relative line broadening results in

$$\frac{\Gamma_\mathrm{a}}{\Gamma_\mathrm{nat}} = 3.69 \pm 0.12.$$

The effect of this correction factor is shown for the Lorentz fit in Figure 18, where it is also compared to the theoretical expected curve. The relative broadening factor is used to correct the measured linewidth or rather the lifetime with

$$\tau_\mathrm{cor} = \frac{\Gamma_\mathrm{a}}{\Gamma_\mathrm{nat}}\,\tau_\mathrm{meas},$$

to

$$\begin{aligned}\tau_{\mathrm{cor,\ Gaussian}} &= (80 \pm 3)\,\mathrm{ns},\\ \tau_{\mathrm{cor,\ Lorentz}} &= (85 \pm 4)\,\mathrm{ns},\\ \tau_{\mathrm{cor,\ Voigt}} &= (89 \pm 10)\,\mathrm{ns}.\end{aligned}$$

The values still deviate strongly from the literature value $\tau_\mathrm{lit} = 141\,\mathrm{ns}$. The corrected lifetime from the Gaussian fit deviates by $20\,\sigma$, with a relative uncertainty of 3.75 %. The lifetime resulting from the Lorentz fit deviates by $14\,\sigma$, with a relative uncertainty of 4.71 %. The result from the Voigt fit deviates the least standard deviations to the expected value with $5.2\,\sigma$, but also has the highest relative uncertainty of 11.2 %.
Ideally, the Voigt fit should yield the best results, since it can filter out homogeneous broadening effects and therefore makes it possible to acquire a more accurate linewidth in the physical context. However, the convolution of both a Gaussian and Lorentz function, with no analytical solution, makes this fit function hard to optimize and not very stable. As can be seen in Figure 16 the Voigt follows the Lorentz function very closely and looking at the fit parameters displayed in Table 2, the fitting algorithm put nearly 85 % of the full width into the Lorentz parameter γ, but with a high uncertainty of around 10 %. The Gaussian width received a value that lies within $1\,\sigma$ to zero, with a relative uncertainty of 120 %. Clearly the Voigt fit did not work properly.
One way to overcome this problem is to determine the resolution limit of the setup and using it to fix the Gaussian width parameter or input it as a starting guess for the regression algorithm. This is reasonable, since the resolution is expected to be the dominant contribution to the homogeneous broadening effects.
The resolution could be quantified by measuring an extra absorber of known linewidth and setting the Lorentz part of a Voigt fit to that literature value to obtain σ. However this is not practicable in the context of this experiment, due to time constraints and the lack of absorber material being Mößbauer active under ambient conditions (room temperature, etc.). Alternatively, an absorber could be used which has a negligible linewidth compared to the resolution to obtain it.

Figure 18 shows the expected natural line in red, with the amplitude, offset and position value from the Lorentz fit. The width is determined by the literature value $\Gamma_\mathrm{nat} = 4.7 \cdot 10^{-9}\,\mathrm{eV}$ from [10]. The green curve shows the Lorentz fit, corrected with the relative line broadening factor. Clearly the corrected curve is still broader than the natural line, which causes the deviation from the literature value. This extra thickness most probable results from the statistical fluctuations of the sledges velocity, which limits the resolution of the Mößbauer spectroscopy. A more detailed description of this can be found in [21].

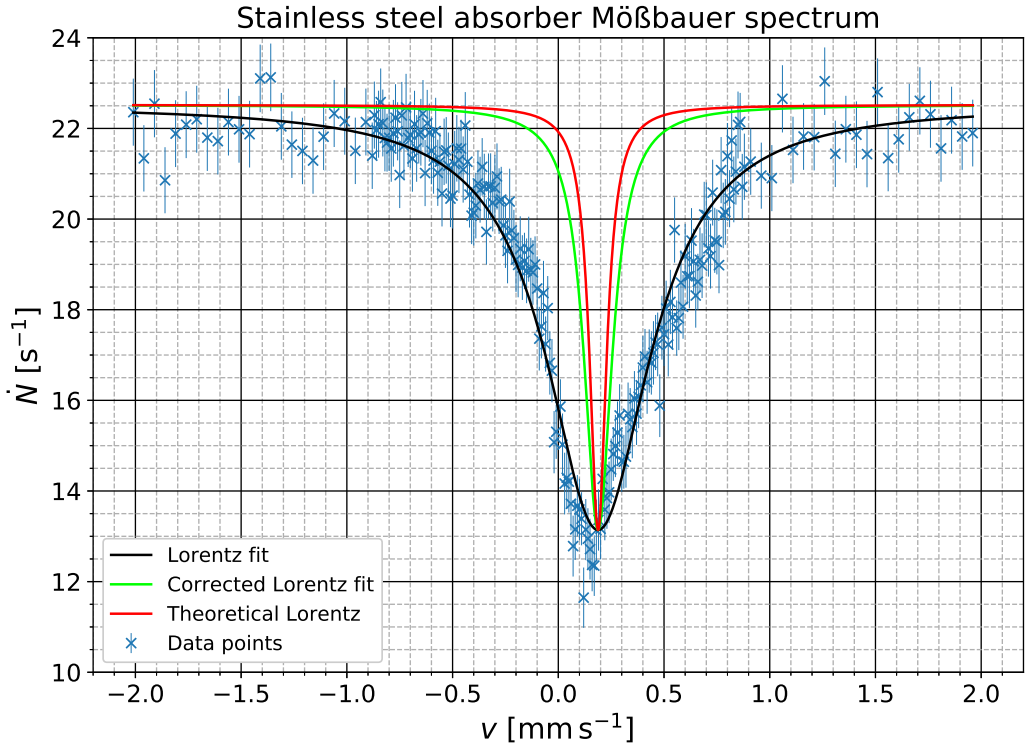


Fig. 18: Exemplary demonstration of the effect of the relative broadening correction factor on the Lorentz fit. The black curve shows the original Lorentz fit, while the green curve shows the corrected Lorentz fit, where the Lorentz width parameter is divided by the relative broadening. As a comparison, the red curve indicates the theoretical Lorentz curve, where the width parameter is set to the literature value and the remaining parameters are taken from the Lorentz fit.

Furthermore the thickness of the source has an impact on the obtained lifetime values. Since its thickness is not precisely known and only estimated to be of $\mathcal{O}(100\,\text{Å})$ we do not know whether it really is only 100 Å as used in the calculations or higher, i.e., in the range of 1000 Å. But since this would shift the relative line broadening only up to roughly 3.85 and, for example the lifetime of the Voigt fit up to around 92.4 ns, which is still in the $1\,\sigma$ interval of the original value, we don't suspect that the deviations are caused by the thickness of the source. Only if d_Q is massively underestimated, so that T_Q would be close to 10, the lifetime would shift into the expected range of 141 ns. This seems very unlikely and is therefore disregarded.

Because of these reasons, the stated lifetimes have to be understood as only lower bounds for the real value of the lifetime.

4.9. Natural Iron Absorber

The natural iron absorber sample is placed on the sledge and the measured Mößbauer spectrum is shown in Figure 19. The spectrum is measured in a velocity range of $-8\,\mathrm{mm\,s^{-1}}$ to $8\,\mathrm{mm\,s^{-1}}$ for different measurement times t. The data was collected over a period of several days during which the weather conditions outside and therefore the temperature inside the not temperature controlled laboratory changed in a range of 30 °C to 20 °C.
The counting rates are Compton and attenuation corrected for as described in Section 4.7. Sixfold Gaussian, Lorentz and Voigt functions are fitted onto the data. The fit parameters are displayed in Table 3, 4 and 5.

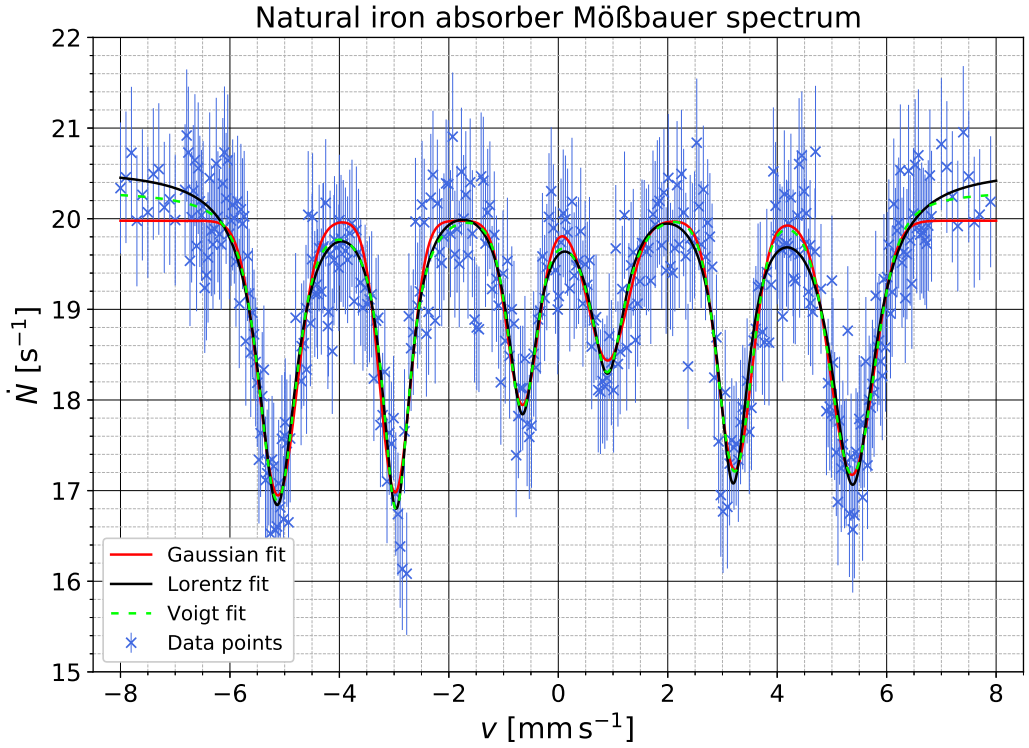


Fig. 19: Mößbauer spectrum of the natural iron absorber sample. The peaks are labeled as 1 through 6 from left to right. The data points are shown in royal blue with errorbars only on the rate $\dot{N}$, since the uncertainty of the velocity could not be quantified. The red curve shows the sixfold Gaussian, the black curve the sixfold Lorentz and the lime green dashed curve the sixfold Voigt fit function. The optimal fit parameters and the reduced χ^2 are listed in Table 3, 4 and 5.

The resulting reduced χ^2 of around 0.5 for all three functions indicates relatively good fits, but also implies that a relevant amount of noise is present or that the uncertainties are overestimated. In the appendix Figure 28, 29 and 30 show the residual plots for the three functions. The noisy behaviour is clearly visible. However, one can also see that only 3 or 4 data points deviate more than $2\,\sigma$ from the fitting curve, indicated by the pink color. Less than 15 % of the data points deviate more than $1\,\sigma$ from the expected value obtained by the fitting curves, indicated by the gold colored data points. No data point lies outside the $3\,\sigma$ interval. The obtained data yields relative uncertainties of

around 5 %. This relatively high uncertainty originates from the uncertainty on the rate, before correction. Therefore the overall precision could have been improved by longer total measurement times for each data point.
Looking at Figure 19 the noisy behaviour of the data points is also clearly visible. As stated above, the temperature conditions in the laboratory and therefore of the source, the absorber and the detection apparatus changed over the period of measurements by 10 °C, which has an effect on the efficiency and response of the used electronics, i.e., the scintillator or the amplifiers. In principle the temperature also has an effect on the fraction of recoilless resonance emission and therefore the measured counting rates. However, looking at Figure 3 in the theoretical considerations of the Debye-Waller factor, the difference is minuscule in the room temperature region and therefore disregarded as the cause for the noise. From the results of [30], it is known that doped NaI scintillators, like the one used in this experiment show a temperature dependent peak position behaviour. A change of 10 °C showed a shift of peak positions of up to 6 % in [30]. In our case this will shift the 14.4 keV peak in or out of our set SCA energy window and therefore change the counting rates significantly. Looking closely at Figure 19, at i.e., peak 6 two dips at different heights can be made out, if one disregards the errorbars for the moment and only takes the data points into account. Also at around $-2\,\mathrm{mm\,s^{-1}}$ a second bow is clearly visible. With a shift of the peaks position in the energy window, only the counting rate, but not the absorption peaks curve will be influenced. Therefore this will mostly affect the evaluations based on the counting rates and only slightly the ones based on the widths or positions of the peaks.

Additionally it has to be noted that the Voigt fit assigned negative width values for peak 6. This has no physically meaning and is simply a mathematical possibility in the fitting process.

Gaussian	μ [$\mathrm{mm\,s^{-1}}$]	σ [$\mathrm{mm\,s^{-1}}$]	A [$\mathrm{mm\,s^{-2}}$]
Peak 1	-5.12 ± 0.02	0.35 ± 0.02	2.7 ± 0.2
Peak 2	-2.98 ± 0.02	0.28 ± 0.02	2.10 ± 0.14
Peak 3	-0.65 ± 0.03	0.28 ± 0.03	1.43 ± 0.14
Peak 4	0.90 ± 0.04	0.36 ± 0.04	1.4 ± 0.2
Peak 5	3.23 ± 0.02	0.31 ± 0.02	2.1 ± 0.2
Peak 6	5.37 ± 0.02	0.39 ± 0.02	2.8 ± 0.2

Tab. 3: Fit parameters for the sixfold Gaussian fit with the offset $B = (19.98 \pm 0.05)\,\mathrm{s^{-1}}$ and $\chi^2_\nu = 0.51$. The curve is shown in Figure 19.

Lorentz	μ [mm s^{-1}]	γ [mm s^{-1}]	A [mm s^{-2}]
Peak 1	-5.14 ± 0.02	0.43 ± 0.03	4.9 ± 0.4
Peak 2	-2.96 ± 0.02	0.32 ± 0.03	3.6 ± 0.3
Peak 3	-0.66 ± 0.03	0.33 ± 0.05	2.6 ± 0.3
Peak 4	0.90 ± 0.04	0.40 ± 0.06	2.5 ± 0.3
Peak 5	3.20 ± 0.02	0.35 ± 0.03	3.5 ± 0.3
Peak 6	5.38 ± 0.02	0.48 ± 0.04	5.1 ± 0.4

Tab. 4: Fit parameters for the sixfold Lorentz fit with the offset $B = (20.56 \pm 0.08)\,\mathrm{s}^{-1}$ and $\chi^2_\nu = 0.50$. The curve is shown in Figure 19.

Voigt	μ [mm s^{-1}]	σ [mm s^{-1}]	γ [mm s^{-1}]	A [mm s^{-2}]
Peak 1	-5.13 ± 0.02	0.21 ± 0.10	0.26 ± 0.12	3.8 ± 0.6
Peak 2	-2.96 ± 0.02	0.1 ± 0.2	0.29 ± 0.08	3.3 ± 0.4
Peak 3	-0.66 ± 0.03	0.19 ± 0.12	0.2 ± 0.2	1.9 ± 0.5
Peak 4	0.90 ± 0.04	0 ± 2	0.4 ± 0.2	2.4 ± 0.6
Peak 5	3.22 ± 0.02	0.25 ± 0.08	0.14 ± 0.13	2.8 ± 0.4
Peak 6	5.38 ± 0.02	-0.32 ± 0.10	-0.2 ± 0.2	-3.9 ± 0.6

Tab. 5: Fit parameters for the sixfold Voigt fit with the offset $B = (20.320 \pm 0.013)\,\mathrm{s}^{-1}$ and $\chi^2_\nu = 0.50$. The curve is shown in Figure 19.

4.9.1. Isomeric Shift E_{Iso}

To obtain the isomeric shift for the natural iron absorber, the offset from zero has to be determined. To do so the positions of half the distance between every pair of symmetric peaks are calculated. The 3 resulting values for each fit function are then averaged with their respective uncertainty as weights. Analogous to the calculations performed in Section 4.8.1 the values are converted into eV. The results are

$$\begin{aligned} E_{\mathrm{Iso,\ Gaussian}} &= (5.9 \pm 0.5)\,\mathrm{neV}, \\ E_{\mathrm{Iso,\ Lorentz}} &= (5.8 \pm 0.4)\,\mathrm{neV}, \\ E_{\mathrm{Iso,\ Voigt}} &= (6.0 \pm 0.4)\,\mathrm{neV}. \end{aligned}$$

The individual values are listed in the appendix in Table 16. The Gaussian result holds the highest relative uncertainty with 8.5 %. The Lorentz and Voigt fit are more precise with 6.9 % and 6.7 % respectively, but not by much. The uncertainties follow directly from the quality of the fits and the data. As stated before the collected data is very noisy and is the reason for the relative high uncertainties. No literature value is known, but the order of magnitude neV is expected.

4.9.2. Magnetic Field Strength B at the Nucleus and the Magnetic Moment μ_{e} of the 14.4 keV State

To calculate the magnetic field strength B at the nucleus and the magnetic moment μ_{e} of the 14.4 keV state, the transition energies due to the underlying hyperfine structure

for the six allowed and observed transitions have to be calculated. For this the positions μ in the Mößbauer spectrum for the six peaks are used with

$$E = E_\gamma \frac{\mu}{c} - E_{\text{Iso}},$$

to obtain the transition energies. Since the absolute value of the transition energies for peak 1 & 6, 2 & 5 and 3 & 4 are supposed to be identical, the means are calculated with

$$E_{i,j} = \frac{|E_i| + |E_j|}{2}, \tag{12}$$

where $\{i, j\} \in \{\{1, 6\}, \{2, 5\}, \{3, 4\}\}$ and the results are stated in Table 6.

Transition energy	Gaussian	Lorentz	Voigt
$E_{1,6}$ [neV]	251.9 ± 0.8	252.7 ± 0.7	252.4 ± 0.8
$E_{2,5}$ [neV]	149.1 ± 0.7	147.9 ± 0.7	148.6 ± 0.7
$E_{3,4}$ [neV]	37.3 ± 1.2	37.4 ± 1.1	37.3 ± 1.2

Tab. 6: Transition energies for the observed peaks averaged for symmetric peak pairs and evaluated with a Gaussian, Lorentz and Voigt function.

From Table 1, Equation 5 and Equation 12 it follows that

$$E_{1,6} = (\mu_\text{g} - \mu_\text{e}) \cdot B, \tag{13}$$

$$E_{2,5} = \left(\mu_\text{g} - \frac{1}{3}\mu_\text{e}\right) \cdot B, \tag{14}$$

$$E_{3,4} = \left(\mu_\text{g} + \frac{1}{3}\mu_\text{e}\right) \cdot B. \tag{15}$$

Combining Equation 14 and 15 yields the magnetic field strength at the nucleus

$$B = \frac{E_{3,4} + E_{2,5}}{2\mu_\text{g}}.$$

With the magnetic moment of the ground state of ^{57}Fe, $\mu_\text{g} = (0.090\,44 \pm 0.000\,07)\,\mu_\text{N}$ from [3], the nuclear magneton $\mu_\text{N} = 3.152\,45 \cdot 10^{-18}\,\text{eV}\,\text{T}^{-1}$ from [31] and the measured data the magnetic field strength at the nucleus results in

$$\begin{aligned} B_\text{Gaussian} &= (32.7 \pm 0.3)\,\text{T}, \\ B_\text{Lorentz} &= (32.5 \pm 0.2)\,\text{T}, \\ B_\text{Voigt} &= (32.6 \pm 0.2)\,\text{T}. \end{aligned}$$

Comparing this to the literature value

$$B_\text{lit} = 33.0\,\text{T}$$

from [2] at 300 K, the values lie within $1\,\sigma$ for the Gaussian, $2.5\,\sigma$ for the Lorentz and $2\,\sigma$ for the Voigt fit to the literature value and have a relative uncertainty $< 1\,\%$. Therefore the values are in agreement and the slight deviations probably arise from the experiment

not being tempered at 300 K.
Equation 13 can be used to calculate the magnetic moment of the 14.4 keV state as

$$\mu_\mathrm{e} = \mu_\mathrm{g} - \frac{E_{1,6}}{B}.$$

The results are

$$\begin{aligned}\mu_{\mathrm{e,\ Gaussian}} &= (-0.154 \pm 0.002)\,\mu_\mathrm{N},\\ \mu_{\mathrm{e,\ Lorentz}} &= (-0.156 \pm 0.002)\,\mu_\mathrm{N},\\ \mu_{\mathrm{e,\ Voigt}} &= (-0.155 \pm 0.002)\,\mu_\mathrm{N}.\end{aligned}$$

Compared to the literature value

$$\mu_{\mathrm{e,\ lit}} = (-0.1549 \pm 0.0002)\,\mu_\mathrm{N}$$

from [3], all values lie within their $1\,\sigma$ interval to the literature value, have a relative uncertainty $< 1.3\,\%$ and therefore yield a high confidence in their correctness.

No significant differences between the Gaussian, Lorentz and Voigt fit arise, which is to no surprise, since the calculated quantities B and μ_e only depend on the position of the absorption peaks and not their width for example, which heavily depends on the form of the curve.

4.9.3. Effective Absorber Thickness T_A and Debye-Waller Factor of the Source f_Q

The effective absorber thickness T_A for the natural iron absorber is calculated analogous as T_A for the stainless steel absorber described in Section 4.8.2. Only the fraction f of the iron content in the absorber changes to $f = (98 \pm 2)\,\%$ as stated in [1]. This changes the number of iron atoms in the absorber to

$$n_\mathrm{A} = (8.3 \pm 0.2) \cdot 10^{28}\,\mathrm{m}^{-3}.$$

With Equation 8 and the stated values in Section 4.8.2, the effective absorber thickness for the natural iron sample results in

$$T_\mathrm{A} = 9.0 \pm 0.3.$$

However, this describes only the total effective absorber thickness for the natural iron absorber. Each absorption peak has its own effective absorber thickness, which is weighted with the respective relative intensity

$$T_\mathrm{A}^j = W_j T_\mathrm{A}$$

for an unsplit emission and split absorption spectrum as explained in [28]. The intensities I_j are calculated with

$$I_j = \dot{N}(\infty) - \dot{N}(\mu_j)$$

for each absorption peak. Summing up the intensities I_j yields the normalization factor

$$N = \sum_1^6 I_j,$$

which is used to determine the individual relative intensity of each absorption peak

$$W_j = \frac{I_j}{N}.$$

Using Equation 10 from Section 4.8.3 and the individual T_A^j, the effective Debye-Waller factors f_Q^j for each absorption peak are calculated and alongside the weights W_j and the effective absorber thicknesses T_A^j listed in Table 7 for the sixfold Gaussian, Table 8 for the sixfold Lorentz and Table 9 for the sixfold Voigt fit.
Since no literature values are known, only the consistency between the values can be checked. All values for the Debye-Waller factors f_Q^j across the different functions coincide within $2\,\sigma$. Studying the tables in more detail, always peak 4 deviates from the other values of the respective fit and holds the highest relative uncertainty. Looking at the fit parameters in Table 3, 4 and 5, peak 4 also holds in principle the highest relative uncertainty in all parameters, though it does not stand out by much. Only the Gaussian width σ in the Voigt fit really stands out, with $(0 \pm 2)\,\mathrm{mm\,s^{-1}}$. But since the deviations are so minuscule, no compatibility problems arise.
Again, the Voigt fit should yield the best results, but as stated previously in Section 4.8.4 it is nearly impossible for the fitting algorithm to determine accurately, what amount of the width belongs to the Gaussian or Lorentz part, with no prior educated and informed starting value guess.
Neither for the weights W_j, nor the individual effective absorber thicknesses T_A^j literature values are known. T_A^j holds relative uncertainties of 5 % to 10 %, while W_j holds 2 % to 4 %.

j	W_j	T_A^j	f_Q^j
1	0.200 ± 0.004	1.80 ± 0.09	0.21 ± 0.02
2	0.198 ± 0.004	1.78 ± 0.09	0.21 ± 0.02
3	0.134 ± 0.004	1.21 ± 0.08	0.21 ± 0.05
4	0.102 ± 0.004	0.92 ± 0.07	0.20 ± 0.10
5	0.182 ± 0.004	1.64 ± 0.09	0.21 ± 0.03
6	0.185 ± 0.004	1.67 ± 0.09	0.21 ± 0.03

Tab. 7: Individual weights W_j, effective absorber thicknesses T_A^j and Debye-Waller factors f_Q^j of the source for each absorption peak j resulting from the sixfold Gaussian fit.

j	W_j	T_A^j	f_Q^j
1	0.191 ± 0.006	1.72 ± 0.12	0.26 ± 0.03
2	0.193 ± 0.006	1.74 ± 0.12	0.26 ± 0.03
3	0.140 ± 0.005	1.26 ± 0.10	0.26 ± 0.07
4	0.117 ± 0.005	1.05 ± 0.10	0.25 ± 0.10
5	0.179 ± 0.006	1.61 ± 0.11	0.26 ± 0.04
6	0.180 ± 0.006	1.62 ± 0.11	0.26 ± 0.04

Tab. 8: Individual weights W_j, effective absorber thicknesses T_A^j and Debye-Waller factors f_Q^j of the source for each absorption peak j resulting from the sixfold Lorentz fit.

j	W_j	T_{A}^j	f_{Q}^j
1	0.194 ± 0.009	1.75 ± 0.17	0.24 ± 0.04
2	0.200 ± 0.009	1.80 ± 0.17	0.24 ± 0.04
3	0.138 ± 0.008	1.24 ± 0.15	0.24 ± 0.10
4	0.114 ± 0.008	1.03 ± 0.14	0.23 ± 0.15
5	0.176 ± 0.009	1.59 ± 0.16	0.24 ± 0.05
6	0.178 ± 0.009	1.61 ± 0.16	0.24 ± 0.05

Tab. 9: Individual weights W_j, effective absorber thicknesses T_{A}^j and Debye-Waller factors f_{Q}^j of the source for each absorption peak j resulting from the sixfold Voigt fit.

4.9.4. Linewidth Γ and Lifetime τ of the 14.4 keV State in ^{57}Fe

The linewidth Γ and the lifetime τ is calculated as explained in Section 4.8.4 for each fit function. Since no data is available for the relative line broadening of the split absorption spectrum, only the lower limit

$$\Gamma_{\mathrm{a}}/\Gamma_{\mathrm{nat}} = 2$$

is used to correct the measured lifetime τ. This originates from the overlapping of emission and absorption spectra as explained before. With this only a crude lower limit for the lifetime can be calculated. The results are listed in Table 10 for the Gaussian, Table 11 for the Lorentz and Table 12 for the Voigt fit.

To no surprise huge deviations from the literature value $\tau_{\mathrm{lit}} = 141$ ns result, since we can only give a crude lower bound for the lifetime τ due to the unknown relative broadening of the linewidths and additional broadening resulting from the resolution limit as explained in Section 4.8.4.
Interesting are the results from the Voigt fit as the relative uncertainties are high and in some cases close to 100 %. Peak 6 also yields negative values, which arise from the fit function assigning negative width values in Table 5. The sign can be ignored since it is obviously an error in the fitting algorithm with no physical meaning.

Gaussian	Γ_{meas} [neV]	τ_{meas} [ns]	$\tau_{\mathrm{low.\ bo.}}$ [ns]
Peak 1	40 ± 2	16.6 ± 0.9	33 ± 2
Peak 2	32 ± 2	21 ± 2	42 ± 3
Peak 3	32 ± 3	21 ± 2	42 ± 4
Peak 4	40 ± 5	16 ± 2	33 ± 4
Peak 5	35 ± 3	18.8 ± 1.4	38 ± 3
Peak 6	44 ± 3	14.8 ± 0.9	30 ± 2

Tab. 10: Linewidth, lifetime and the corrected lower bound lifetime of the 14.4 keV state resulting from a sixfold Gaussian fit.

Lorentz	Γ_{meas} [neV]	τ_{meas} [ns]	$\tau_{\text{low. bo.}}$ [ns]
Peak 1	41 ± 3	16.0 ± 1.3	32 ± 3
Peak 2	31 ± 3	21 ± 2	42 ± 4
Peak 3	32 ± 4	21 ± 3	41 ± 6
Peak 4	38 ± 6	17 ± 3	35 ± 5
Peak 5	33 ± 3	20 ± 2	40 ± 4
Peak 6	46 ± 4	14.3 ± 1.1	29 ± 2

Tab. 11: Linewidth, lifetime and the corrected lower bound lifetime of the 14.4 keV state resulting from a sixfold Lorentz fit.

Voigt	Γ_{meas} [neV]	τ_{meas} [ns]	$\tau_{\text{low. bo.}}$ [ns]
Peak 1	25 ± 11	26 ± 12	52 ± 24
Peak 2	28 ± 7	23 ± 6	47 ± 12
Peak 3	17 ± 15	40 ± 37	80 ± 74
Peak 4	38 ± 19	17 ± 8	35 ± 17
Peak 5	13 ± 12	50 ± 48	101 ± 95
Peak 6	-20 ± 15	-33 ± 25	-67 ± 51

Tab. 12: Linewidth, lifetime and the corrected lower bound lifetime of the 14.4 keV state resulting from a sixfold Voigt fit.

5. Summary and Discussion

Mößbauer spectra were obtained for a stainless steel and natural iron absorber with the 14.4 keV transition of the excited ^{57}Fe state. To do so multiple preliminary calibrations and measurements had to be performed.
The used electronics were checked with an oscilloscope and delays and the shaping time were set. The used MCA had to be calibrated with the use of an ^{241}Am source with a rotatable wheel in front of it to excite different materials with known K_α decays. These were used to obtain a linear relation between channels and energy

$$f(E) = (18.8 \pm 0.4)\,\mathrm{keV}^{-1} \cdot E + (1.2 \pm 0.9).$$

With this the spectrum of the used ^{57}Co source, which decays via electron capture into an excited state of ^{57}Fe, is measured and the 14.4 keV transition peak identified. A SCA discriminator window was set, so that only photons of that transition were used for detection. Due to Compton scattering, photons of higher energies are also measured in this setup and therefore the Compton background had to be determined

$$\dot{N}_{\mathrm{Compton}} = (20.5 \pm 0.3)\,\mathrm{s}^{-1}$$

and used for correction. Additionally the attenuation of photons in the acrylic glass encasing the absorber material has been quantified

$$T_{\mathrm{cor}} = (55.6 \pm 0.7)\,\%$$

and is also used for further correction of the measured counting rates.
The velocity of the sledge was checked to see, whether the set velocity at the PC is also the real velocity of the sledge. The results did not show any relevant deviations, though the measurement was not very precise. No uncertainty for the velocity was used in the evaluation, since no reasonable uncertainty could be determined or estimated, but the velocity uncertainty is expected to have an effect on the Mößbauer spectra, causing deviations in the analysis, especially for the linewidths of the peaks and the resulting lifetimes.

The measured Mößbauer spectra were fitted with Gaussian, Lorentz and Voigt functions for the stainless steel absorber and sixfold versions of these functions for the natural iron absorber in order to obtain quantities like the isomeric shift E_{Iso}, the Debye-Waller factor of the source f_{Q}, the linewidth Γ and the lifetime τ of the 14.4 keV state and additionally for the natural iron absorber the magnetic field strength B at the nucleus and the magnetic moment μ_{e} of the 14.4 keV state.
Only the results obtained by the Lorentz fits will be shown, since they hold the highest confidence of correctness. The Gaussian fit results are discarded since the absorption peaks show an asymmetric behaviour, better described by a Lorentz curve, which is also expected from theoretical considerations of atomic decay. The Voigt fit, as a convolution of both Gaussian and Lorentz curve, is in principle the optimal function, since it includes the theoretical expectation and the experimental distortions, caused mostly by the used electronics. However additional data is required to obtain an accurate fit, like the resolution limit caused by the statistical fluctuations in the sledges velocity. The resolution mainly contributes to the observed homogeneous broadening of the absorption peaks, but this could not be measured due to the lack of available Mößbauer active materials

in the experiment.
The measured isomeric shifts are

$$E_{\mathrm{Iso}}^{(\mathrm{S})} = (9.0 \pm 0.2)\,\mathrm{neV},$$
$$E_{\mathrm{Iso}}^{(\mathrm{N})} = (5.8 \pm 0.4)\,\mathrm{neV},$$

for the stainless steel (S) and natural iron (N) absorber. No literature values are known, but the order of neV is expected. The effective absorber thicknesses, which are required for the calculation of the Debye-Waller factors are

$$T_{\mathrm{A}}^{(\mathrm{S})} = 6.4 \pm 0.5,$$
$$T_{\mathrm{A}}^{(\mathrm{N})} = 9.0 \pm 0.3,$$

with relative uncertainties of 7.8 % and 3.3 % respectively. No manufacturer value is known for the two absorber materials, so no comparison can be made. For the natural iron absorber, T_{A} has to be weighted individually with the relative intensity of each absorption peak and the results are listed in Table 13, with their individual Debye-Waller factors of the source. The Debye-Waller factor calculated from the stainless steel absorber spectrum is

$$f_{\mathrm{Q}}^{(\mathrm{S})} = 0.543 \pm 0.003,$$

with a relative uncertainty of 0.5 %. The values from the natural iron absorber deviate strongly from the value for the stainless steel absorber with around $9\,\sigma$. Peak 4 only deviates by $2.8\,\sigma$, but holds a relative uncertainty of 40 %, which relativizes the comparably small deviation. The noisiness of the data, which is discussed in Section 4.9, and the overlap of measurements from different temperatures in the laboratory for the natural iron absorber could be the reason for this deviation. Also no literature value is known for the source, so no comparison with the real value can be made. The value from the stainless steel absorber seems reasonable, if not somewhat low for a source purchased specifically for an experiment on recoilless emission of gamma radiation.
It has to be noted that S. Margulies and J. R. Ehrman state in [27], that the used equation for the Debye-Waller factor of the source, Equation 10, is exact only for non-resonantly absorbing sources and if that is not the case only an approximation for an effective source thickness $T_{\mathrm{Q}} \ll 1$. Also it is only valid, if the FWHM of the source and absorber are equal to the natural widths Γ_{nat} of the transitions. If they deviate, but Γ_{A} and Γ_{Q} are identical the absorption cross section σ_0 in Equation 9 has to be multiplied by the factor $\Gamma_{\mathrm{nat}}^{\mathrm{Q}}/\Gamma_{\mathrm{Q}}$. Since Γ_{Q} can only be bigger than the natural width, the additional factor would decrease σ_0 and therefore increase f_{Q}. Also the slow variation of $\mathrm{J}_0(iT_{\mathrm{A}}/2)$ with T_{A} makes the results obtained with this method not very precise [28]. Since it is not known, whether the source is non-resonantly absorbing or to which degree and no information about the actual linewidths are noted, deviations from the real value caused by the used theoretical considerations cannot be ruled out. The values from the natural iron absorber are disregarded for being to small and $f_{\mathrm{Q}}^{(\mathrm{S})}$ is assumed to be of the correct order, but with an overestimated accuracy.

The lifetime of the 14.4 keV state can be directly calculated from the width of the absorption peaks. It has to be corrected for unavoidable broadening effects caused by the measurements. The relative line broadening is at least 2, since there is always the overlap of emission and absorption spectra. For the stainless steel absorber literature values

are available that also include the effects of the finite thickness of the absorber and the source. With a relative line broadening of $\Gamma_\mathrm{a}/\Gamma_\mathrm{nat} = 3.69 \pm 0.12$, the corrected lifetime results in

$$\tau_\mathrm{cor}^{(\mathrm{S})} = (85 \pm 4)\,\mathrm{ns}.$$

This deviates strongly from the literature value

$$\tau_\mathrm{lit} = 141\,\mathrm{ns}$$

from [1], with $14\,\sigma$ and a relative uncertainty of 4.71 %. From the measured lifetime one can deduct, that the width of the absorption peak is still broader than the natural one. Most of the additional broadening can be attributed to the resolution, which results from statistical fluctuations of the sledges velocity [21]. Here the Voigt function would be the optimal use, if the resolution would be known, since it would filter out this homogeneous broadening and yield more accurate results. But since the resolution is now known and could not be estimated, the stated lifetime is only a lower bound for the real value.
No literature values are known for the broadening effects for the natural iron absorber and therefore only the minimal relative broadening of 2 is used to give a crude lower bound on the lifetime τ.

j	$T_\mathrm{A}^{j,\,(\mathrm{N})}$	$f_\mathrm{Q}^{j,\,(\mathrm{N})}$	$\tau_\mathrm{low.\,bo.}^{j,\,(\mathrm{N})}$ [ns]
1	1.72 ± 0.12	0.26 ± 0.03	32 ± 3
2	1.74 ± 0.12	0.26 ± 0.03	42 ± 4
3	1.26 ± 0.10	0.26 ± 0.07	41 ± 6
4	1.05 ± 0.10	0.25 ± 0.10	35 ± 5
5	1.61 ± 0.11	0.26 ± 0.04	40 ± 4
6	1.62 ± 0.11	0.26 ± 0.04	29 ± 2

Tab. 13: Individual effective absorber thicknesses $T_\mathrm{A}^{j,\,(\mathrm{N})}$, Debye-Waller factors $f_\mathrm{Q}^{j,\,(\mathrm{N})}$ and lower bound lifetimes $\tau_\mathrm{low.\,bo.}^{j,\,(\mathrm{N})}$ of the 14.4 keV state for the natural iron absorber resulting from the sixfold Lorentz fit.

For the natural iron absorber it was also possible to calculate the magnetic field strength B at the nucleus from the transition energies of the six absorption peaks. The Lorentz fit resulted in

$$B^{(\mathrm{N})} = (32.5 \pm 0.2)\,\mathrm{T}.$$

Compared to the literature value from [2]

$$B_\mathrm{lit} = 33.0\,\mathrm{T},$$

the measured value lies within $2.5\,\sigma$ to the literature, with a relative uncertainty of 0.62 %. The magnetic moment of the excited state results in

$$\mu_\mathrm{e}^{(\mathrm{N})} = (-0.156 \pm 0.002)\,\mu_\mathrm{N},$$

with a relative uncertainty of 1.3 %. Compared to the literature value

$$\mu_{\mathrm{e,\,lit}} = (-0.1549 \pm 0.0002)\,\mu_{\mathrm{N}}$$

from [3], an agreement of both values within $1\,\sigma$ is found. The results from the Gaussian and Voigt fit also agree with the literature value for B and μ_{e}, since these values only depend on the position of the absorption peaks, which are not deeply influenced by the form of the curve of the used fit functions.

These measurements illustrate the high energy resolution that can be reached with Mößbauer spectroscopy, making it a highly useful measurement tool for material characterisations. As some possible improvements a temperature isolation of the scintillator would be proposed. Also a more stable version of the measurement software would simplify the conduction of the experiment.

Appendix

A. Additional Plots

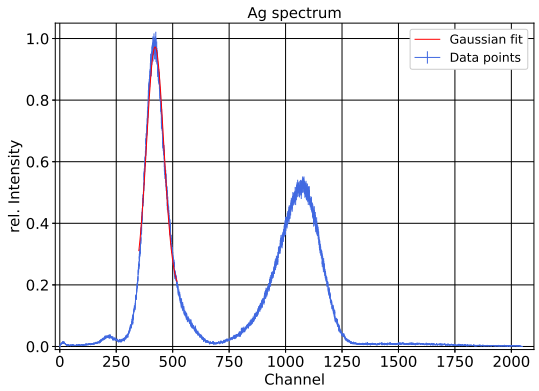


Fig. 20: Spectrum of Ag activated with an ^{241}Am source. The red curve indicates the Gaussian fit used to obtain the position value of the K_α peak. The fit parameters are listed in Table 14.

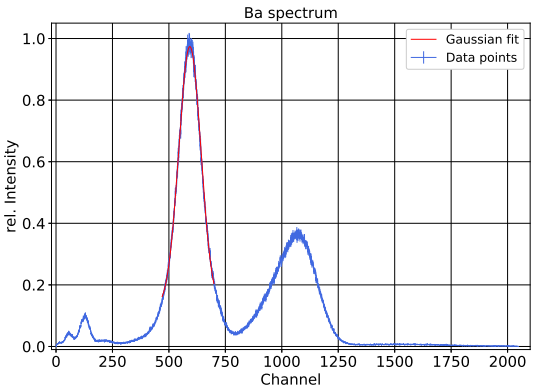


Fig. 21: Spectrum of Ba activated with an ^{241}Am source. The red curve indicates the Gaussian fit used to obtain the position value of the K_α peak. The fit parameters are listed in Table 14.

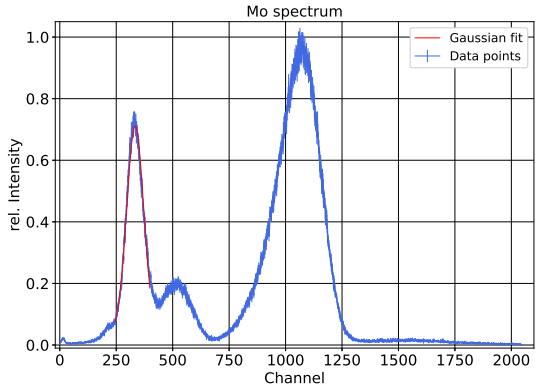


Fig. 22: Spectrum of Mo activated with an ^{241}Am source. The red curve indicates the Gaussian fit used to obtain the position value of the K_α peak. The fit parameters are listed in Table 14.

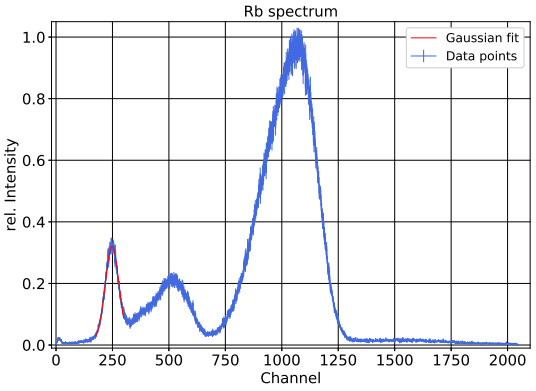


Fig. 23: Spectrum of Rb activated with an ^{241}Am source. The red curve indicates the Gaussian fit used to obtain the position value of the K_α peak. The fit parameters are listed in Table 14.

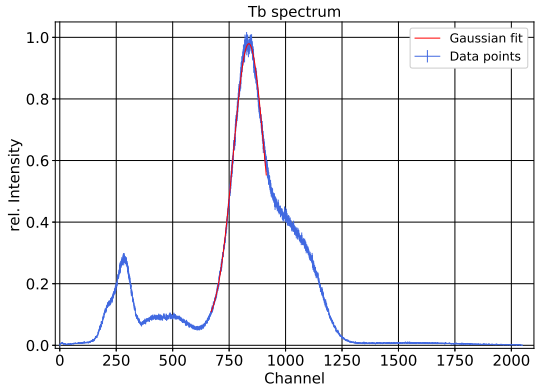


Fig. 24: Spectrum of Tb activated with an ^{241}Am source. The red curve indicates the Gaussian fit used to obtain the position value of the K_α peak. The fit parameters are listed in Table 14.

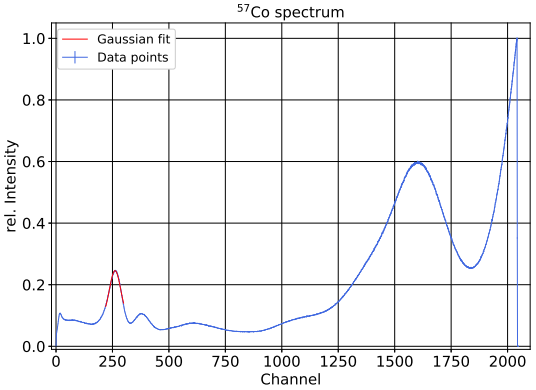


Fig. 25: Spectrum of the used ^{57}Ce source. The red curve indicates the Gaussian fit used to obtain the position value of the 14.4 keV peak. The fit parameters are listed in Table 14.

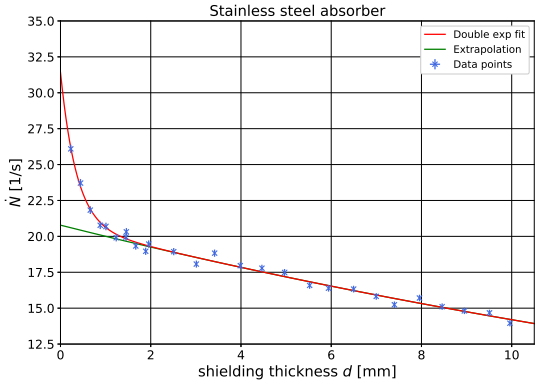


Fig. 26: Counting rate in dependence of the aluminium shielding thickness d for the stainless steel absorber. The red curve shows the double exponential fit, while the green curve indicates the extrapolation of the exponential decay, caused by the high energetic photons to no shielding, which is identified as the Compton background rate.

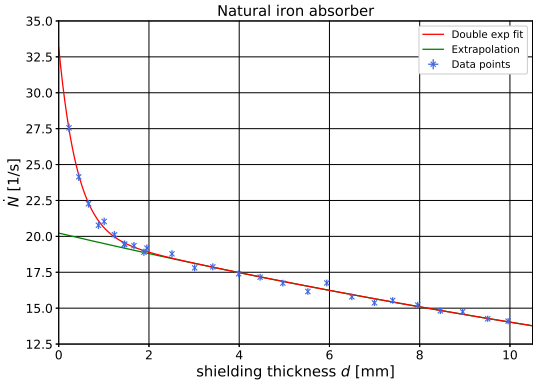


Fig. 27: Counting rate in dependence of the aluminium shielding thickness d for the natural iron absorber. The red curve shows the double exponential fit, while the green curve indicates the extrapolation of the exponential decay, caused by the high energetic photons to no shielding, which is identified as the Compton background rate.

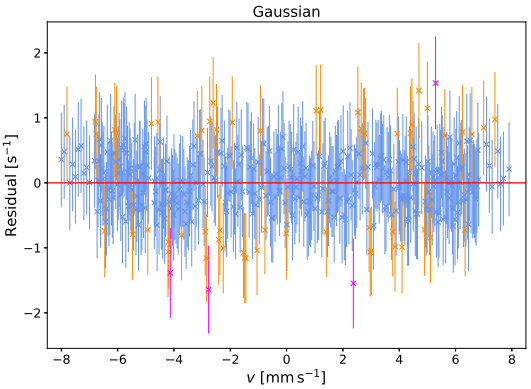


Fig. 28: Residual plot of the sixfold Gaussian fit for the natural iron absorber. The red line indicates the expected values from the fit function. Blue colored data points lie within $< 1\,\sigma$, orange points within $< 2\,\sigma$ and purple points within $< 3\,\sigma$ of the fit functions value. Only the uncertainties from the data points are used for this consideration.

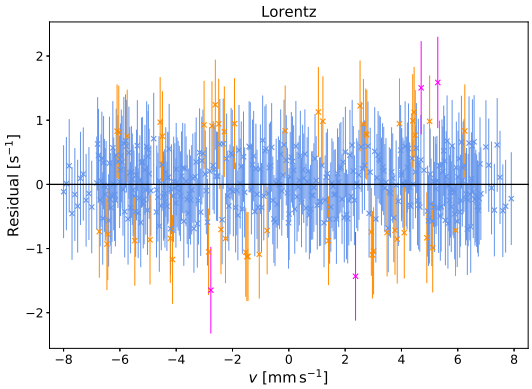


Fig. 29: Residual plot of the sixfold Lorentz fit for the natural iron absorber. The black line indicates the expected values from the fit function. Blue colored data points lie within $< 1\,\sigma$, orange points within $< 2\,\sigma$ and purple points within $< 3\,\sigma$ of the fit functions value. Only the uncertainties from the data points are used for this consideration.

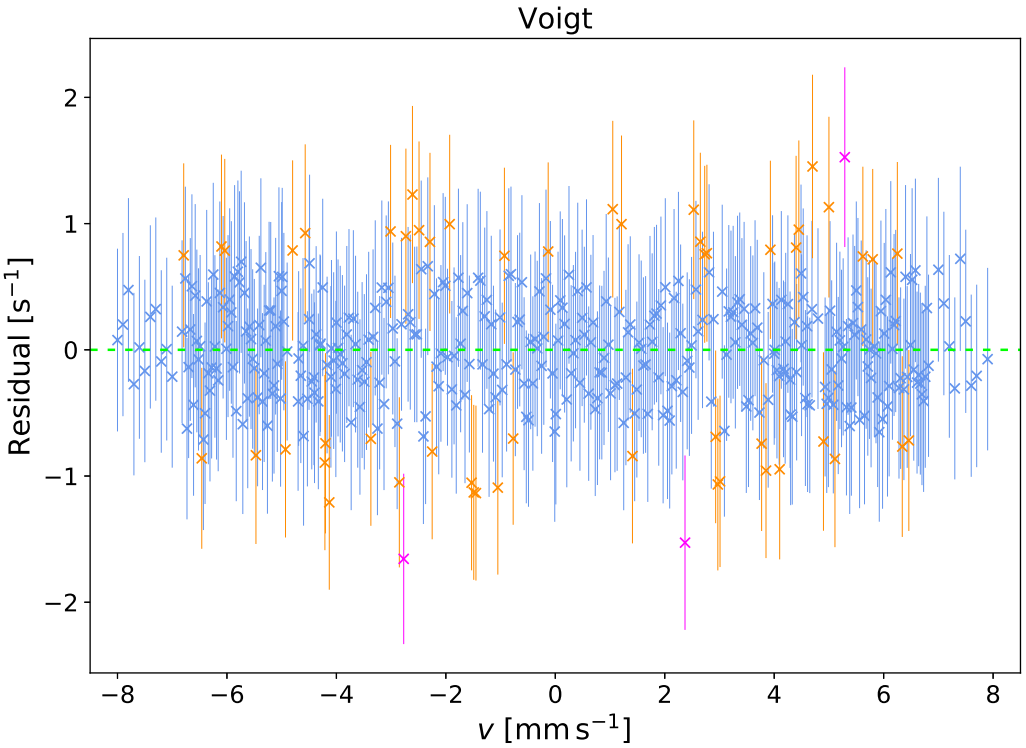


Fig. 30: Residual plot of the sixfold Voigt fit for the natural iron absorber. The dashed lime green line indicates the expected values from the fit function. Blue colored data points lie within $< 1\,\sigma$, orange points within $< 2\,\sigma$ and purple points within $< 3\,\sigma$ of the fit functions value. Only the uncertainties from the data points are used for this consideration.

B. Additional Tables

	Ag	Ba	Mo	Rb	Tb	^{57}Ce
μ	422.3 ± 0.2	592.69 ± 0.10	333.9 ± 0.2	248.5 ± 0.2	837.2 ± 0.2	261.52 ± 0.10
σ	37.9 ± 0.5	49.0 ± 0.2	37.7 ± 0.5	30.1 ± 0.4	69.3 ± 0.3	25.1 ± 0.7
A	75.2 ± 1.3	104.3 ± 0.6	64.1 ± 1.1	23.1 ± 0.4	160.2 ± 0.9	9.8 ± 0.5
B	0.182 ± 0.006	0.125 ± 0.002	0.033 ± 0.006	0.0142 ± 0.0002	0.058 ± 0.002	0.089 ± 0.005
Res.	$(9.0 \pm 0.7)\,\%$	$(8.3 \pm 0.4)\,\%$	$(11.3 \pm 0.8)\,\%$	$(12 \pm 1)\,\%$	$(8.3 \pm 0.5)\,\%$	$(9.6 \pm 0.7)\,\%$

Tab. 14: Fit parameters of the K_α peaks obtained with Gaussian fits onto the measured calibration spectra shown in Figure 20−24 and the ^{57}Co spectrum in Figure 11. The last line indicates the detectors energy (channel) dependent resolution, calculated with σ/μ [19].

	A [s^{-1}]	a [mm^{-1}]	B [s^{-1}]	b [mm^{-1}]
(S)	20.8 ± 0.2	-0.0381 ± 0.0010	10.6 ± 1.2	-2.8 ± 0.4
(N)	20.2 ± 0.2	-0.0366 ± 0.0012	12.9 ± 0.8	-2.5 ± 0.2

Tab. 15: Fit parameters of the double exponential fit for quantization of the Compton background for the stainless steel (S) and natural iron (N) absorber shown in Figure 26 and 27 respectively.

	Gaussian			Lorentz			Voigt		
Peaks	1 & 6	2 & 5	3 & 4	1 & 6	2 & 5	3 & 4	1 & 6	2 & 5	3 & 4
E_{Iso} [mm s^{-1}]	0.12 ± 0.02	0.122 ± 0.014	0.13 ± 0.03	0.122 ± 0.014	0.117 ± 0.014	0.12 ± 0.02	0.120 ± 0.012	0.129 ± 0.014	0.12 ± 0.02
E_{Iso} [neV]	5.9 ± 0.7	5.9 ± 0.7	6.0 ± 1.2	5.9 ± 0.7	5.6 ± 0.6	5.9 ± 1.1	5.8 ± 0.7	6.2 ± 0.7	5.7 ± 1.1

Tab. 16: Isomeric shifts from the sixfold Gaussian, Lorentz and Voigt fits for the natural iron absorber, averaged for the symmetric peak pairs 1 & 6, 2 & 5, 3 & 4.

C. Error Propagation

If the N variables x_i of a function f are not correlated,

$$s_f = \sqrt{\left(\frac{df}{dx_1} s_1\right)^2 + ... + \left(\frac{df}{dx_N} s_N\right)^2} \tag{16}$$

is used with the error s_i of x_i. In case that the variables are correlated,

$$s_f = \sqrt{(\nabla f)^T \cdot M \cdot \nabla f} \tag{17}$$

has to be applied. Here M is the covariance matrix.

D. List of Figures

E. List of Tables

F. References

[1] Zwerger, A., et al. *Advanced Lab Course: Mößbauer-Effect*, 2007.

[2] Fultz, Brent. *Mössbauer Spectrometry*. Department of Applied Physics and Materials Science, California Institute of Technology, Pasadena, California, page 9, 2011.

[3] Stone, N. J. *Table of Nuclear Magnetic Dipole and Electric Quadrupole Moments*. Oxford Physics, Clarendon Laboratory Parks Road, page 17, 1998.

[4] Unterweger, M. P., et al. *Radionuclide Half-Life Measurements Data*. `https://www.nist.gov/pml/radionuclide-half-life-measurements/radionuclide-half-life-measurements-data`. Accessed: 2020-09-14, original data published in [32].

[5] Pratt, R. H. *Tutorial on fundamentals of radiation physics: interactions of photons with matter*. Radiation Physics and Chemistry, 70 (4-5): 595–603, 2004.

[6] Nelson, G., et al. *Gamma-Ray Interactions with Matter*. `https://faculty.washington.edu/agarcia3/phys575/Week2/Gamma%20ray%20interactions.pdf`. Accessed: 2020-09-17.

[7] Einstein, Albert. *Über einem die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt*. Annalen der Physik, 4, 1905.

[8] Compton, Arthur. *A quantum theory of the scattering of X-rays by light elements*. Physical review, 21 (5): 483, 1923.

[9] NIST. *X-Ray Mass Attenuation Coefficients*. `https://physics.nist.gov/PhysRefData/XrayMassCoef/chap2.html`. Accessed: 2020-09-21.

[10] Mayer-Kuckuk, T. *Kernphysik, Teubner Studienbücher Physik*. Teubner Stuttgart, 1984.

[11] RSC. *Introduction to Mossbauer Spectroscopy*. `https://www.rsc.org/Membership/Networking/InterestGroups/MossbauerSpect/intro.asp`. Accessed: 2020-09-22.

[12] Ashcroft, N. & Mermin, D. *Festkörperphysik*. Oldenburg Verlag München Wien, 2007.

[13] Kittel, Charles. *Einführung in die Festkörperphysik*. Oldenburg Verlag München Wien, 1988.

[14] Klingelhöfer, G., et al. *The Miniaturized Mössbauer Spectrometer MIMOS II of the Athena Payload for the 2003 MER Missions*. Sixth International Conference on Mars, `https://www.lpi.usra.edu/meetings/sixthmars2003/pdf/3132.pdf`, 2003.

[15] Caltech. *The Mossbauer effect: hyperfine splitting*. `http://www.sophphx.caltech.edu/Physics_7/Experiment_29.pdf`, 2017. Accessed: 2020-09-19.

[16] Wertheim, G. K. *Measurement of Local Fields at Impurity Fe 57 Atoms Using the Mössbauer Effect.* Physical Review Letters, 4 (8): 403, 1960.

[17] Marx, Matthias. *Untersuchung der VUV - Emission atomarer Fragmente nach Anregung von molekularen Gasen durch schnelle Ion - Molekül - Stöße.* Doktorarbeit, Albert-Ludwigs-Universität Freiburg, page 176 ff., 1992.

[18] Leo, William R. *Techniques for Nuclear and Particle Physics Experiments - A How-to Approach - 2nd Edition.* page 118, 1994.

[19] Leo, William R. *Techniques for Nuclear and Particle Physics Experiments - A How-to Approach - 2nd Edition.* 1994.

[20] NIST. *X-Ray Mass Attenuation Coefficients: Polymethyl Methacrylate.* `https://physics.nist.gov/PhysRefData/XrayMassCoef/ComTab/pmma.html`. Accessed: 2020-09-03.

[21] Irkaev, Sobir M. *Trends in Mössbauer Spectrometer Designs.* Mössbauer Effect Reference and Data Journal, 28 (10), December 2005.

[22] Heberle, J. *Linewidth of Mössbauer Absorption.* Nuclear Instruments and Methods, 58: 90–92, 1967.

[23] Chechev, V. P. & Kuzmenko, N. K. *Table de Radionuclèides*, 2001-2004.

[24] IUPAC. *Atomic weights of the elements 2011 (IUPAC Technical Report).* `http://dx.doi.org/10.1351/PAC-REP-13-03-02`. Accessed: 2020-09-03.

[25] NIST. *Composition of IRON.* `https://physics.nist.gov/cgi-bin/Star/compos.pl?matno=026`. Accessed: 2020-09-03.

[26] NIST. *Fundamental Physical Constants.* `https://physics.nist.gov/cgi-bin/cuu/Value?na`. Accessed: 2020-09-03.

[27] Margulies, S. & Ehrman, J. R. *Transmission and line broadening of resonance radiation incident on a resonance absorber.* Nuclear Instruments and Methods 12, pages 131–137, February 1961.

[28] Margulies, S. & Debrunner, P. & Frauenfelder, H. *Transmission and line broadening in the Mössbauer effect II.* Nuclear Instruments and Methods 12, pages 217–231, August 1962.

[29] Olivero, J. J. & Longbothum, R. L. *Empirical fits to the Voigt line width: A brief review.* Journal of Quantitative Spectroscopy & Radiative Transfer `https://doi.org/10.1016/0022-4073(77)90161-3`, 17 (2): 233–236, 1977.

[30] Alexandrov, Boian, et al. *Temperature behaviour of the doped NaI (Tl) scintillators and its impact on the pulse height analysis instrumentation.* Los Alamos National Laboratory, 87544 NM, July 2005.

[31] NIST. *Fundamental Physical Constants.* `https://physics.nist.gov/cgi-bin/cuu/Value?munev|search_for=nuclear+magneton`. Accessed: 2020-09-15.

[32] Unterweger, M. P., et al. *New and revised half-life measurements results*. Nuclear instruments and methods in physics research section a: accelerators, spectrometers, detectors and associated equipment, 312 (1-2): 349–352, 1992.

G. Python Code

G.1. Setup Check

```
import numpy as np
import sympy as sp
import pylab as pl
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

import peakutils as peak
import glob
import os

def read_in(name,col1,col2):
        x=np.array(np.genfromtxt(np.str(name),usecols=col1,dtype=np.float,
            ↪ delimiter=",",skip_header=1,skip_footer=0))
        y=np.array(np.genfromtxt(np.str(name),usecols=col2,dtype=np.float,
            ↪ delimiter=",",skip_header=1,skip_footer=0))
        return [x,y]

for file in glob.glob("*.CSV"):
        print(file)
        x,y=read_in(file,0,1)
        plt.plot(x,y)
        plt.title(file[:(len(file)-4)])
        plt.savefig(file[:(len(file)-4)]+".svg")
        plt.show()
```

G.2. Calibration of the MCA

Calibration Spectra

```
import numpy as np
import sympy as sp
import pylab as pl
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
plt.rcParams.update({'axes.titlesize': 'xx-large'})
plt.rcParams.update({'axes.labelsize': 'xx-large'})
plt.rcParams.update({'xtick.labelsize': 'xx-large'})
plt.rcParams.update({'ytick.labelsize': 'xx-large'})
plt.rcParams.update({'legend.fontsize': 'x-large'})


def read_in(name,col2):
        y=np.array(np.genfromtxt(np.str(name),usecols=col2,dtype=np.float,
            ↪ delimiter="",skip_header=2,skip_footer=0))
        return y
def err_sqrt(Liste):
```

```
17          Liste_err=[]
18          for a in Liste:
19                  Liste_err.append(np.sqrt(a))
20          return Liste_err
21
22  def Gaussian(x, sigma, mu, A, B):
23      return A/np.sqrt(2*np.pi*sigma**2) * np.exp(-0.5*((x-mu)/sigma)**2) + B
24
25  Name = ['Ag_spec', 'Ba_spec', 'Mo_spec', 'Rb_spec', 'Tb_spec']
26  Title_names = ['Ag', 'Ba', 'Mo', 'Rb', 'Tb']
27
28  fit_ranges = [[350,520], [470,700], [245,400], [175,300], [670,915]]
29  p0_liste = [(39,423,83,0.13), (53,592,117,0.07), (37,335,58,0.07),
        ↪ (30,249,21,0.03), (68,837,156,0.07)]
30
31  L=[]
32  for i in range(0,len(Name)):
33      y = read_in(Name[i]+'.TKA',0)
34      y_err = y/max(y)*np.sqrt(1/(y+1) + 1/max(y))
35      x = np.linspace(0,len(y),len(y))
36      y = y/max(y)
37
38      plt.errorbar(x,y,yerr=y_err,zorder=1, linewidth=0.3, color='royalblue',
            ↪ label='Data points', fmt='-')
39
40      xfitdata = x[fit_ranges[i][0]:fit_ranges[i][1]]
41      yfitdata = y[fit_ranges[i][0]:fit_ranges[i][1]]
42      yfiterr = y_err[fit_ranges[i][0]:fit_ranges[i][1]]
43      fitPara, fitCova = curve_fit(Gaussian, xfitdata, yfitdata, sigma=
            ↪ yfiterr,p0=p0_liste[i])
44      sigmas = fitPara[0]
45      mus = fitPara[1]
46      sigma_error = np.sqrt(fitCova[0][0])
47      mu_errors = np.sqrt(fitCova[1][1])
48      A = fitPara[2]
49      B = fitPara[3]
50      A_err = np.sqrt(fitCova[2][2])
51      B_err = np.sqrt(fitCova[3][3])
52
53      plt.plot(xfitdata, Gaussian(xfitdata, fitPara[0], fitPara[1], fitPara
            ↪ [2], fitPara[3]), linewidth=1, color='r', label='Gaussian fit',
            ↪ zorder=2)
54
55      plt.title(Title_names[i]+' spectrum')
56      plt.xlabel('Channel')
57      plt.ylabel('rel. Intensity')
58      plt.xlim(-20,2100)
59      plt.ylim(-0.01,1.05)
60      plt.legend()
61      plt.grid(which='major',color='k',linewidth=0.3)
62      plt.rc('axes', axisbelow=True)
63      plt.savefig(f'{Name[i]}.eps')
64      plt.show()
65      print(Name[i]+f': Mu={round(mus,2)}+-{round(mu_errors,2)}, Sigma={round
            ↪ (sigmas,2)}+-{round(sigma_error,2)}, A={round(A,2)}+-{round(A_err
            ↪ ,2)}, B={round(B,4)}+-{round(B_err,4)} \n sigma/mu={round(sigmas/
            ↪ mus*100,5)}%')
```

```
66      L.append(Name[i]+f': Mu=;{round(mus,2)};+-;{round(mu_errors,2)}; [
            ↪ Channel], Sigma=;{round(sigmas,2)};+-;{round(sigma_error,2)}; [
            ↪ Channel], A=;{round(A,2)};+-;{round(A_err,2)}; [Channel], B=;{
            ↪ round(B,4)};+-;{round(B_err,4)};, sigma/mu={round(sigmas/mus
            ↪ *100,5)}% \n')
67
68
69  #Background
70  y = read_in('background-long.TKA',0)
71  y_err = y/max(y)*np.sqrt(1/(y+1) + 1/max(y))
72  x = np.linspace(0,len(y),len(y))
73  print(max(y))
74  y = y/max(y)
75
76
77  plt.errorbar(x,y, yerr=y_err, color='royalblue',label='Data points',
        ↪ linewidth=0.4)
78  plt.title('Background spectrum')
79  plt.xlabel('Channel')
80  plt.ylabel('rel. Intensity')
81  plt.legend()
82  plt.xlim(-20,2100)
83  plt.ylim(-0.01,1.1)
84  plt.grid(which='major',color='k',linewidth=0.3)
85  plt.rc('axes', axisbelow=True)
86  plt.savefig('background.eps')
87  plt.show()
88
89  # source spectrum night measurement
90
91  y = read_in('spectrum-night-meas.TKA',0)
92  y_err = y/max(y)*np.sqrt(1/(y+1) + 1/max(y))
93  x = np.linspace(0,len(y),len(y))
94  y=y/max(y)
95  plt.errorbar(x,y, yerr=y_err, zorder=1, color='royalblue', label='Data
        ↪ points', linewidth=1)
96
97  a=220
98  b=300
99  xfitdata = x[a:b]
100 yfitdata = y[a:b]
101 p0=(50,300,50,0.2)
102 yfiterr = y_err[a:b]
103 fitPara, fitCova = curve_fit(Gaussian, xfitdata, yfitdata, sigma=yfiterr,p0
        ↪ =p0)
104 sigmas = fitPara[0]
105 mus = fitPara[1]
106 sigma_error = np.sqrt(fitCova[0][0])
107 mu_errors = np.sqrt(fitCova[1][1])
108 A = fitPara[2]
109 B = fitPara[3]
110 A_err = np.sqrt(fitCova[2][2])
111 B_err = np.sqrt(fitCova[3][3])
112
113 plt.plot(xfitdata, Gaussian(xfitdata, fitPara[0], fitPara[1], fitPara[2],
        ↪ fitPara[3]), linewidth=1, color='r', label='Gaussian fit', zorder=2)
114
115 plt.title(r'$^{57}$Co spectrum')
```

```
116 plt.xlabel('Channel')
117 plt.ylabel('rel. Intensity')
118 plt.legend()
119 plt.xlim(-20,2100)
120 plt.ylim(-0.01,1.05)
121 plt.grid(which='major',color='k',linewidth=0.3)
122 plt.rc('axes', axisbelow=True)
123 plt.savefig('source-spec.eps')
124 plt.show()
125
126 print(f'Source: Mu={round(mus,2)}+-{round(mu_errors,2)}, Sigma={round(
        ↪ sigmas,2)}+-{round(sigma_error,2)}, A={round(A,2)}+-{round(A_err,2)},
        ↪  B={round(B,4)}+-{round(B_err,4)} \n sigma/mu={round(sigmas/mus
        ↪ *100,5)}%')
127 L.append(f'Source: Mu=;{round(mus,2)};+-;{round(mu_errors,2)}; [Channel],
        ↪ Sigma=;{round(sigmas,2)};+-;{round(sigma_error,2)}; [Channel], A=;{
        ↪ round(A,2)};+-;{round(A_err,2)}; [Channel], B=;{round(B,4)};+-;{round
        ↪ (B_err,4)};, sigma/mu={round(sigmas/mus*100,5)}% \n')
128
129 #Daten in File Speichern
130 file=open(r'fit-daten.txt','w+')
131 file.writelines(L)
132 file.close()
133
134 y = read_in('alu-null.TKA',0)
135 x = np.linspace(0,len(y),len(y))
136 y_err = y/max(y)*np.sqrt(1/(y+1) + 1/max(y))
137 x = np.linspace(0,len(y),len(y))
138 y=y/max(y)
139 plt.errorbar(x,y, yerr=y_err, color='royalblue',label='Data points',
        ↪ linewidth=1)
140 plt.title(r'SCA window')
141 plt.xlabel('Channel')
142 plt.ylabel('rel. Intensity')
143 plt.legend()
144 plt.xlim(-20,2100)
145 plt.ylim(-0.01,1.05)
146 plt.grid(which='major',color='k',linewidth=0.3)
147 plt.rc('axes', axisbelow=True)
148 plt.savefig('window.eps')
149 plt.show()
```

Calibration Fit

```
1  import numpy as np
2  import sympy as sp
3  import pylab as pl
4  import matplotlib.pyplot as plt
5  from scipy.optimize import curve_fit
6  plt.rcParams.update({'axes.titlesize': 'xx-large'})
7  plt.rcParams.update({'axes.labelsize': 'xx-large'})
8  plt.rcParams.update({'xtick.labelsize': 'x-large'})
9  plt.rcParams.update({'ytick.labelsize': 'x-large'})
10 plt.rcParams.update({'legend.fontsize': 'large'})
11
12 def read_in(name,col1,col2,skipheader,skipfooter):
```

```
13          a = np.array(np.genfromtxt(np.str(name),usecols=col1,dtype=np.float
                ↪ ,delimiter=";",skip_header=skipheader,skip_footer=skipfooter)
                ↪ )
14          b = np.array(np.genfromtxt(np.str(name),usecols=col2,dtype=np.float
                ↪ ,delimiter=";",skip_header=skipheader,skip_footer=skipfooter)
                ↪ )
15          return [a,b]
16
17  y,y_error = read_in('fit-daten.txt',1,5,0,1)
18
19  real_energy = [22.10,32.06,17.44,13.37,44.23] #keV
20
21  def lin_fit(x, a, b):
22      return a*x + b
23
24  fitPara, fitCova = curve_fit(lin_fit, real_energy, y, sigma=y_error)
25  a_err = np.sqrt(fitCova[0][0])
26  b_err = np.sqrt(fitCova[1][1])
27  a = fitPara[0]
28  b = fitPara[1]
29
30  x = np.linspace(0,50,5000)
31  plt.plot(x,lin_fit(x,a,b), label='Linear fit',zorder=1,color='royalblue')
32  plt.errorbar(real_energy, y, yerr=y_error, fmt='x', label='Data points',
        ↪ zorder=2,color='k')
33
34  y2,y_error2 = read_in('fit-daten.txt',1,5,5,0)
35  plt.errorbar(14.4,y2,yerr=y_error2, fmt='x', color='red', label='14.4$\,
        ↪ $keV peak',zorder=2)
36
37  plt.xlabel(r'$E$ [keV]')
38  plt.ylabel('Channel')
39  plt.title('Energy-Channel calibration')
40  plt.legend()
41  plt.xlim(0,50)
42  plt.ylim(0,1000)
43  plt.grid(which='major',color='k',linewidth=0.3)
44  plt.rc('axes', axisbelow=True)
45  file_name='energy-channel-calibration-fit'
46  plt.savefig(file_name+'.eps')
47  plt.show()
48
49  print(f'f(x) = a={round(fitPara[0],2)}+-{round(a_err,2)}[1/keV] * x + b={
        ↪ round(fitPara[1],1)}+-{round(b_err,1)}')
50
51  r = []
52  DOF = len(real_energy)-2
53  for i in range(0,len(real_energy)):
54      r.append(((y[i]-lin_fit(real_energy[i],*fitPara))/y_error[i])**2)
55
56  print(sum(r)/DOF)
57
58  ############ Energy of the fitted 14.4keV peak
59  channel = 261.52
60  s_channel = 25.12
61  print((channel-fitPara[1])/fitPara[0])
62  print((s_channel-fitPara[1])/fitPara[0])
```

G.3. Compton Background

```
import numpy as np
import sympy as sp
import pylab as pl
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
plt.rcParams.update({'axes.titlesize': 'xx-large'})
plt.rcParams.update({'axes.labelsize': 'xx-large'})
plt.rcParams.update({'xtick.labelsize': 'x-large'})
plt.rcParams.update({'ytick.labelsize': 'x-large'})
plt.rcParams.update({'legend.fontsize': 'large'})

def read_in(name,col1,skipheader,skipfooter):
        a = np.array(np.genfromtxt(np.str(name),usecols=col1,dtype=np.float
            ↪ ,delimiter=";",skip_header=skipheader,skip_footer=skipfooter)
            ↪ )
        return a

def doppel_e(x,A,a,B,b):
    return A*np.exp(a*x) + B*np.exp(b*x)

# Langen der Plattchen

Alu_daten = [[1.005,1,1.005,1.005,1,1,1.005,1,1,1.07,1,1.005,1.005],
    ↪ [1.46,1.47,1.46,1.46,1.46,1.465,1.455,1.455,1.46,1.46],
    ↪ [1.95,1.95,1.955,1.955,1.955,1.95,1.945,1.975,1.955],
    ↪ [2.51,2.51,2.51,2.51,2.51,2.505,2.51,2.51,2.51,2.51,2.51],
    ↪ [3.01,3.01,3.01,3.015,3.015,3.015,3.01,3.01,3.01,3.02,3.01,3.01,3.005],
    ↪  [3.985,3.99,3.99,3.985,3.98,3.98,4,3.985,3.98],
    ↪ [0.28,0.215,0.21,0.21,0.21,0.21,0.26,0.225],
    ↪ [0.245,0.215,0.22,0.21,0.205,0.205,0.21,0.22],
    ↪ [0.21,0.22,0.21,0.225,0.215,0.225,0.21,0.22],
    ↪ [0.215,0.22,0.245,0.22,0.23,0.205,0.22,0.205]] #mm

Alu_langen_mittel = [] #mm
Alu_langen_fehler = [] #mm

for i in range(0,len(Alu_daten)):
    Alu_langen_mittel.append(sum(Alu_daten[i])/len(Alu_daten[i]))
    Alu_langen_fehler.append(np.std(Alu_daten[i],ddof=1)/np.sqrt(np.float(
        ↪ len(Alu_daten[i]))))


a = Alu_langen_mittel[0]
b = Alu_langen_mittel[1]
c = Alu_langen_mittel[2]
d = Alu_langen_mittel[3]
e = Alu_langen_mittel[4]
f = Alu_langen_mittel[5]
p1 = Alu_langen_mittel[6]
p2 = Alu_langen_mittel[7]
p3 = Alu_langen_mittel[8]
p4 = Alu_langen_mittel[9]
Length = [a,b,c,d,e,b+c,f,c+d,c+e,d+e,c+f,f+d,f+e,f+b+c,f+b+d,f+e+b,f+c+e,f
    ↪ +e+d,f+e+c+a,p1,p1+p2,p1+p2+p3,p1+p2+p3+p4,a+p1,a+p1+p2,a+p1+p2+p3,a+
    ↪ p1+p2+p3+p4] #mm

```

```
43 a = Alu_langen_fehler[0]
44 b = Alu_langen_fehler[1]
45 c = Alu_langen_fehler[2]
46 d = Alu_langen_fehler[3]
47 e = Alu_langen_fehler[4]
48 f = Alu_langen_fehler[5]
49 p1 = Alu_langen_fehler[6]
50 p2 = Alu_langen_fehler[7]
51 p3 = Alu_langen_fehler[8]
52 p4 = Alu_langen_fehler[9]
53 Length_err = [a,b,c,d,e,np.sqrt(b**2 + c**2),f,np.sqrt(c**2+d**2),np.sqrt(c
       ↪ **2+e**2),np.sqrt(d**2+e**2),np.sqrt(c**2+f**2),np.sqrt(f**2+d**2),np
       ↪ .sqrt(f**2+e**2),np.sqrt(f**2+b**2+c**2),np.sqrt(f**2+b**2+d**2),np.
       ↪ sqrt(f**2+e**2+b**2),np.sqrt(f**2+c**2+e**2),np.sqrt(f**2+e**2+d**2),
       ↪ np.sqrt(f**2+e**2+c**2+a**2),p1,np.sqrt(p1**2+p2**2),np.sqrt(p1**2+p2
       ↪ **2+p3**2),np.sqrt(p1**2+p2**2+p3**2+p4**2),np.sqrt(a**2+p1**2),np.
       ↪ sqrt(a**2+p1**2+p2**2),np.sqrt(a**2+p1**2+p2**2+p3**2),np.sqrt(a**2+
       ↪ p1**2+p2**2+p3**2+p4**2)] #mm
54
55 #1 linien absorber
56
57 Counts = []
58 Counts_Fehler = []
59
60 for i in range(1,len(Length)+1):
61     y2 = read_in(f'alu-st-{i}.TKA',0,2,0)
62     x = np.linspace(0,len(y2),len(y2))
63     if i==1:
64         a=max(y2)
65     y = y2/a
66     Counts.append(sum(y2)/300)
67     Counts_Fehler.append(np.sqrt(sum(y2))/300)
68
69 plt.errorbar(Length, Counts,xerr=Length_err, yerr=Counts_Fehler,color='
       ↪ royalblue', fmt='x',label='Data points',zorder=3)
70 plt.xlabel(r'shielding thickness $d$ [mm]')
71 plt.ylabel(r'$\dot{N}$ [1/s]')
72
73 p0 = (22,-0.035,2.54,-1.28)
74 x = np.linspace(0,12,5000)
75 fitPara, fitCova = curve_fit(doppel_e, Length, Counts, sigma=Counts_Fehler,
       ↪ p0=p0)
76 A = fitPara[0]
77 a = fitPara[1]
78 B = fitPara[2]
79 b = fitPara[3]
80 plt.plot(x,doppel_e(x,A,a,B,b),zorder=2,color='r',label='Double exp fit')
81 plt.title('Stainless steel absorber')
82 plt.xlim(0,10.5)
83 plt.ylim(12.5,35)
84 plt.plot(x, doppel_e(x,A,a,0,0),color='g',label='Extrapolation',zorder=1)
85
86 A_err = np.sqrt(fitCova[0][0])
87 a_err = np.sqrt(fitCova[1][1])
88 B_err = np.sqrt(fitCova[2][2])
89 b_err = np.sqrt(fitCova[3][3])
90
91 plt.legend()
```

```
92  plt.grid(which='major',color='k',linewidth=0.3)
93  plt.rc('axes', axisbelow=True)
94  plt.savefig('1-linien-alu.eps')
95  plt.show()
96
97  print(f'bei stelle 0: {round(A,3)}+-{round(np.sqrt(A_err**2),2)}[1/s]')
98
99  compton1 = A
100 compton1_err = A_err
101
102 print(*fitPara,'\n')
103 print(round(A_err,1),round(a_err,4),round(B_err,2),round(b_err,1))
104
105 ############### 6 linien absorber ##########################
106
107 Counts = []
108 Counts_Fehler = []
109
110 for i in range(1,len(Length)+1):
111     y2 = read_in(f'alu-ei-{i}.TKA',0,2,0)
112     x = np.linspace(0,len(y2),len(y2))
113     if i==1:
114         a=max(y2)
115     y = y2/a
116     Counts.append(sum(y2)/300)
117     Counts_Fehler.append(np.sqrt(sum(y2))/300)
118
119 plt.errorbar(Length, Counts,xerr=Length_err, yerr=Counts_Fehler, color='
    ↪ royalblue',fmt='x',label='Data points',zorder=3)
120 plt.xlabel(r'shielding thickness $d$ [mm]')
121 plt.ylabel(r'$\dot{N}$ [1/s]')
122
123 a = 20.6
124 b = -0.04
125 c = 2.9
126 d = -1.3
127 p0 = (a,b,c,d)
128 x = np.linspace(0,12,5000)
129 plt.legend()
130 fitPara, fitCova = curve_fit(doppel_e, Length, Counts, sigma=Counts_Fehler,
    ↪ p0=p0)
131 A = fitPara[0]
132 a = fitPara[1]
133 B = fitPara[2]
134 b = fitPara[3]
135 A_err = np.sqrt(fitCova[0][0])
136 a_err = np.sqrt(fitCova[1][1])
137 B_err = np.sqrt(fitCova[2][2])
138 b_err = np.sqrt(fitCova[3][3])
139
140 plt.plot(x,doppel_e(x,A,a,B,b),zorder=2,color='r',label=r'Double exp fit')
141 plt.title('Natural iron absorber')
142 plt.xlim(0,10.5)
143 plt.ylim(12.5,35)
144 plt.plot(x, doppel_e(x,A,a,0,0), label='Extrapolation',color='g',zorder=1)
145 plt.legend()
146 plt.grid(which='major',color='k',linewidth=0.3)
147 plt.rc('axes', axisbelow=True)
```

```
148 plt.savefig('6-linien-alu.eps')
149 plt.show()
150
151 print(f'bei stelle 0: {round(A,3)}+-{round(np.sqrt(A_err**2),2)}[1/s]\n\n')
152
153 compton6 = A
154 compton6_err = A_err
155
156 compton_mittel = (compton1 + compton6) / 2
157 compton_mittel_err = compton_mittel*np.sqrt((compton1_err/compton1)**2 + (
    ↪ compton6_err/compton6)**2)
158
159 print(f'Compton Background rate mean of both 1 and 6 line absorber: {round(
    ↪ compton_mittel,3)}+-{round(compton_mittel_err,2)} [1/s]')
160
161 L = []
162 L.append('Compton Background [1/s], Uncertainty [1/s]\n')
163 L.append(f'{round(compton_mittel,3)},{round(compton_mittel_err,2)}')
164 file=open(r'compton-back.csv','w+')
165 file.writelines(L)
166 file.close()
167
168 print(*fitPara,'\n')
169 print(round(A_err,2),round(a_err,4),round(B_err,2),round(b_err,1))
```

G.4. Attenuation of Gamma Radiation by Acrylic Glass

```
1  import numpy as np
2  import sympy as sp
3  import pylab as pl
4  import matplotlib.pyplot as plt
5  from scipy.optimize import curve_fit
6  plt.rcParams.update({'axes.titlesize': 'xx-large'})
7  plt.rcParams.update({'axes.labelsize': 'xx-large'})
8  plt.rcParams.update({'xtick.labelsize': 'x-large'})
9  plt.rcParams.update({'ytick.labelsize': 'x-large'})
10 plt.rcParams.update({'legend.fontsize': 'large'})
11
12 def read_in(name):
13         a = np.array(np.genfromtxt(np.str(name),usecols=0,dtype=np.float,
               ↪ delimiter=",",skip_header=2,skip_footer=0))
14         return a
15
16 T=600 #s
17 print("-----------------------------------------------------")
18 N1_spec=read_in("plexi.TKA")
19 N1_all=sum(N1_spec)
20 s1_all=np.sqrt(N1_all)
21 N1=N1_all/T
22 s1=s1_all/T
23 print(f"N_Plexi: {round(N1,2)}+-{round(s1,2)} [1/s]")
24
25
26 N2_spec=read_in("no_plexi.TKA")
27
28 N2_all=sum(N2_spec)
```

```
s2_all=np.sqrt(N2_all)
N2=N2_all/T
s2=s2_all/T
print(f"N_Free: {round(N2,2)}+-{round(s2,2)} [1/s]")

print("-------------------------------------------------\n")


print(-np.log(0.556)/(1.19*0.198))
print(np.sqrt((0.7/(100*0.556*1.19*0.198))**2 + (np.log(0.556)
    ↪ *0.002/(1.19*0.198**2))**2))
```

G.5. Velocity of the Sledge

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
plt.rcParams.update({'axes.titlesize': 'xx-large'})
plt.rcParams.update({'axes.labelsize': 'xx-large'})
plt.rcParams.update({'xtick.labelsize': 'x-large'})
plt.rcParams.update({'ytick.labelsize': 'x-large'})
plt.rcParams.update({'legend.fontsize': 'large'})

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

def red_chisquare(x,y,y_err,fit_function,fitPara):
        DOS=len(x)-len(fitPara)
        r=[]
        for i in range(0,len(x)):
                r.append(((y[i]-fit_function(x[i],*fitPara))/y_err[i])**2)
        chisq=sum(r)/DOS
        return chisq

vPC=np.array(np.genfromtxt("velocity.txt",usecols=0,dtype=np.float,
    ↪ delimiter=",",skip_header=1)) #mm/s
xstart=np.array(np.genfromtxt("velocity.txt",usecols=1,dtype=np.float,
    ↪ delimiter=",",skip_header=1)) #cm
xstop=np.array(np.genfromtxt("velocity.txt",usecols=2,dtype=np.float,
    ↪ delimiter=",",skip_header=1)) #cm
t=np.array(np.genfromtxt("velocity.txt",usecols=3,dtype=np.float,delimiter=
    ↪ ",",skip_header=1)) #s
sx=np.array(np.genfromtxt("velocity.txt",usecols=4,dtype=np.float,delimiter
    ↪ =",",skip_header=1)) #cm

s_t=0.3 #s
v_pc = [1,2,3,4,5,6,7,8]
v = [0.992,1.985,2.982,3.97,4.94,5.96,6.97,7.97]
s_v = [0.003,0.006,0.011,0.02,0.02,0.03,0.03,0.04]

plt.errorbar(v_pc,v,yerr=s_v,fmt='x',label='Data points',color='r')
plt.title("Velocity calibration")
plt.grid(which='major',color='k',linewidth=0.3,)
plt.rc('axes', axisbelow=True)
plt.xlim(0,9)
plt.ylim(0,9)
```

```
plt.xlabel(r"${v}_\mathrm{PC}$ [mm$\,$s$^{-1}$]")
plt.ylabel(r"${v}_\mathrm{meas}$ [mm$\,$s$^{-1}$]")

fitPara,fitCova=curve_fit(lin_fit,v_pc,v,sigma=s_v)
a_fit=fitPara[0]
b_fit=fitPara[1]
a_err=np.sqrt(fitCova[0][0])
b_err=np.sqrt(fitCova[1][1])

chisq=red_chisquare(v_pc,v,s_v,lin_fit,fitPara)
print(f"red chisquare: {chisq} \n a: {round(a_fit,4)} +- {round(a_err,4)} \
    ↪ n b: {round(b_fit,3)} +- {round(b_err,3)} \n")

xfit=np.linspace(0,9)
plt.plot(xfit,lin_fit(xfit,a_fit,b_fit),zorder=1,label="Linear fit",color='
    ↪ royalblue',linewidth=1)
plt.legend()
plt.savefig("velocity.eps")
plt.show()
```

G.6. Stainless Steel Absorber

Data Processing

```
import numpy as np
import matplotlib.pyplot as plt

def read_in(name,col1,col2,col3):
        a = np.array(np.genfromtxt(np.str(name),usecols=col1,dtype=np.float
            ↪ ,delimiter=",",skip_header=0,skip_footer=0))
        b = np.array(np.genfromtxt(np.str(name),usecols=col2,dtype=np.float
            ↪ ,delimiter=",",skip_header=0,skip_footer=0))
        c = np.array(np.genfromtxt(np.str(name),usecols=col3,dtype=np.float
            ↪ ,delimiter=",",skip_header=0,skip_footer=0))
        return [a,b,c]

def read_in2(name,col1,col2):
        a = np.array(np.genfromtxt(np.str(name),usecols=col1,dtype=np.float
            ↪ ,delimiter=",",skip_header=1,skip_footer=0))
        b = np.array(np.genfromtxt(np.str(name),usecols=col2,dtype=np.float
            ↪ ,delimiter=",",skip_header=1,skip_footer=0))
        return [a,b]

velocity2,time2,counts2 = read_in('1linien-alle-daten.csv',0,1,2)
velocity = []
time = []
counts = []

i=0
while i < len(velocity2)-1:
    if velocity2[i]==velocity2[i+1]:
        if velocity2[i+1]==velocity2[i+2]:
            if velocity2[i+2]==velocity2[i+3]:
                if velocity2[i+3]==velocity2[i+4]:
                    velocity.append(velocity2[i])
                    time.append(time2[i]+time2[i+1]+time2[i+2]+time2[i+3]+
                        ↪ time2[i+4])
```

```
28                      counts.append(counts2[i]+counts2[i+1]+counts2[i+2]+
                            ↪ counts2[i+3]+counts2[i+4])
29                      i=i+5
30                  else:
31                      velocity.append(velocity2[i])
32                      time.append(time2[i]+time2[i+1]+time2[i+2]+time2[i+3])
33                      counts.append(counts2[i]+counts2[i+1]+counts2[i+2]+
                            ↪ counts2[i+3])
34                      i=i+4
35              else:
36                  velocity.append(velocity2[i])
37                  time.append(time2[i]+time2[i+1]+time2[i+2])
38                  counts.append(counts2[i]+counts2[i+1]+counts2[i+2])
39                  i=i+3
40          else:
41              velocity.append(velocity2[i])
42              time.append(time2[i]+time2[i+1])
43              counts.append(counts2[i]+counts2[i+1])
44              i=i+2
45      else:
46          velocity.append(velocity2[i])
47          time.append(time2[i])
48          counts.append(counts2[i])
49          i=i+1
50
51  rate = []
52  s_rate = []
53  for i in range(0,len(velocity)):
54      rate.append(counts[i]/(time[i]/1000))
55      s_rate.append(np.sqrt(counts[i])/(time[i]/1000))
56
57  compton_back,compton_back_err = read_in2('compton-back.csv',0,1)
58  err_korrigiert = []
59
60  for i in range(0,len(s_rate)):
61      err_korrigiert.append(np.sqrt(s_rate[i]**2 + compton_back_err**2))
62
63
64  T = 0.556 #korrigierte transmission
65  s_T = 0.007
66
67  rate_korrigiert = []
68  s_rate_korrigiert = []
69  for i in range(0,len(rate)):
70      rate_korrigiert.append((rate[i]-compton_back)/T)
71      s_rate_korrigiert.append(np.sqrt( (s_rate[i]/T)**2 + (compton_back_err/
            ↪ T)**2 + ((rate[i]-compton_back)*s_T/T**2)**2) )
72
73  plt.errorbar(velocity, rate_korrigiert, yerr=s_rate_korrigiert, fmt='x',
        ↪ linewidth=0.3)
74  plt.title('corrected 1 line absorber spectrum')
75  plt.xlabel('Velocity [mm/s]')
76  plt.ylabel('Rate [1/s]')
77  plt.savefig('1linien.png')
78  plt.show()
79
80
81  #daten speichern in ein file
```

```
L = []
L.append('Velocity [mm/s], corr Rate [1/s], s_corr_Rate [1/s]\n')
for i in range(0,len(velocity)):
    L.append(f'{velocity[i]},{round(rate_korrigiert[i],3)},{round(
        ↪ s_rate_korrigiert[i],3)}\n')

file=open(r'daten.csv','w+')
file.writelines(L)
file.close()
```

Evaluation

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

plt.rcParams.update({'axes.titlesize': 'xx-large'})
plt.rcParams.update({'axes.labelsize': 'xx-large'})
plt.rcParams.update({'xtick.labelsize': 'x-large'})
plt.rcParams.update({'ytick.labelsize': 'x-large'})
plt.rcParams.update({'legend.fontsize': 'large'})



def read_in(name,col1,col2,col3):
        a = np.array(np.genfromtxt(np.str(name),usecols=col1,dtype=np.float
            ↪ ,delimiter=",",skip_header=1,skip_footer=0))
        b = np.array(np.genfromtxt(np.str(name),usecols=col2,dtype=np.float
            ↪ ,delimiter=",",skip_header=1,skip_footer=0))
        c = np.array(np.genfromtxt(np.str(name),usecols=col3,dtype=np.float
            ↪ ,delimiter=",",skip_header=1,skip_footer=0))
        return [a,b,c]

def Gaussian(x, mu, sigma, A, B):
    return -A/(np.sqrt(2*np.pi)*sigma) * np.exp(-0.5*((x-mu)/sigma)**2) + B

def Lorentz(x,mu,gamma,A,B):
    return -A/(np.pi) * gamma/((x-mu)**2 + gamma**2) + B

def Voigt(x,mu,sigma,gamma,A,B):
    return -A/(sigma*np.sqrt(2*np.pi)) * np.real(wofz((x-mu+1j*gamma)
                                /(sigma*np.sqrt(2)))) + B



velocity, rate, s_rate = read_in('daten.csv',0,1,2)
plt.errorbar(velocity, rate, yerr=s_rate, fmt='x', linewidth=0.5, zorder=1,
    ↪ color='royalblue',label='Data points')


###################### Gaussian Fit ##########################
velocity, rate, s_rate = read_in('daten.csv',0,1,2)
a=220
b=0
velo_gauss      = velocity[b:a]
```

```
42 rate_gauss       = rate[b:a]
43 s_rate_gauss     = s_rate[b:a]
44
45 p0=(0.2,0.3,10,20)
46 fitPara, fitCova = curve_fit(Gaussian, velo_gauss, rate_gauss,sigma=
       ↪ s_rate_gauss, p0=p0)
47
48 mu_gauss    = fitPara[0]
49 sigma_gauss = fitPara[1]
50 A_gauss     = fitPara[2]
51 B_gauss     = fitPara[3]
52
53 mu_err_gauss    = np.sqrt(fitCova[0][0])
54 sigma_err_gauss = np.sqrt(fitCova[1][1])
55 A_err_gauss     = np.sqrt(fitCova[2][2])
56 B_err_gauss     = np.sqrt(fitCova[3][3])
57
58 x=np.linspace(velocity[b],velocity[a],1000)
59 plt.plot(x,Gaussian(x,mu_gauss,sigma_gauss,A_gauss,B_gauss), zorder=2,
       ↪ linewidth='1.5', color='r' ,label='Gaussian fit')
60
61
62 DOS=len(velo_gauss)-4
63 r = []
64 for i in range(0,len(velo_gauss)):
65     r.append(((rate_gauss[i]-Gaussian(velo_gauss[i],mu_gauss,sigma_gauss,
          ↪ A_gauss,B_gauss))/s_rate_gauss[i])**2)
66 redchisq_gauss = sum(r)/DOS
67
68
69 ######################## Loretz Fit ############################
70 velocity, rate, s_rate = read_in('daten.csv',0,1,2)
71 b=0
72 a=220
73
74 velo_lorentz    = velocity[b:a]
75 rate_lorentz    = rate[b:a]
76 s_rate_lorentz  = s_rate[b:a]
77
78 p0=(0.2,0.3,10,20)
79 fitPara, fitCova = curve_fit(Lorentz, velo_lorentz, rate_lorentz, sigma=
       ↪ s_rate_lorentz,p0=p0)
80
81 mu_lorentz      = fitPara[0]
82 gamma_lorentz   = fitPara[1]
83 A_lorentz       = fitPara[2]
84 B_lorentz       = fitPara[3]
85
86 mu_err_lorentz      = np.sqrt(fitCova[0][0])
87 gamma_err_lorentz   = np.sqrt(fitCova[1][1])
88 A_err_lorentz       = np.sqrt(fitCova[2][2])
89 B_err_lorentz       = np.sqrt(fitCova[3][3])
90
91
92 x=np.linspace(velocity[b],velocity[a],1000)
93 plt.plot(x,Lorentz(x,mu_lorentz,gamma_lorentz,A_lorentz,B_lorentz),
       ↪ linestyle='-',linewidth='1.5', color='k', zorder=3, label='Lorentz
       ↪ fit')
```

```
94
95  DOS=len(velo_lorentz)-4
96  r = []
97  for i in range(0,len(velo_lorentz)):
98      r.append(((rate_lorentz[i]-Lorentz(velo_lorentz[i],mu_lorentz,
            ↪ gamma_lorentz,A_lorentz,B_lorentz))/s_rate_lorentz[i])**2)
99  redchisq_lorentz = sum(r)/DOS
100
101
102 ###################### Voigt fit ##########################
103 velocity, rate, s_rate = read_in('daten.csv',0,1,2)
104 b=0
105 a=220
106
107 velo_voigt      = velocity[b:a]
108 rate_voigt      = rate[b:a]
109 s_rate_voigt    = s_rate[b:a]
110
111 p0=(0.188,0.0535,0.298,12,22)
112 fitPara, fitCova = curve_fit(Voigt, velo_voigt, rate_voigt, sigma=
        ↪ s_rate_voigt,p0=p0)
113
114 mu_voigt    = fitPara[0]
115 sigma_voigt = fitPara[1]
116 gamma_voigt = fitPara[2]
117 A_voigt     = fitPara[3]
118 B_voigt     = fitPara[4]
119
120 mu_err_voigt    = np.sqrt(fitCova[0][0])
121 sigma_err_voigt = np.sqrt(fitCova[1][1])
122 gamma_err_voigt = np.sqrt(fitCova[2][2])
123 A_err_voigt     = np.sqrt(fitCova[3][3])
124 B_err_voigt     = np.sqrt(fitCova[4][4])
125
126
127 x=np.linspace(velocity[b],velocity[a],1000)
128 plt.plot(x,Voigt(x,mu_voigt, sigma_voigt,gamma_voigt,A_voigt,B_voigt),
        ↪ dashes=(3,4), linewidth='1.5', color='lime', zorder=4, label='Voigt
        ↪ fit')
129
130 DOS=len(velo_voigt)-5
131 r = []
132 for i in range(0,len(velo_voigt)):
133     r.append(((rate_voigt[i]-Voigt(velo_voigt[i],mu_voigt,sigma_voigt,
134                 gamma_voigt,A_voigt,B_voigt))/s_rate_voigt[i])**2)
135 redchisq_voigt= sum(r)/DOS
136
137
138 plt.title('Stainless steel absorber M  bauer spectrum')
139 plt.xlabel(r'$v$ [mm$\,$s$^{-1}$]')
140 plt.ylabel(r'$\dot{N}$ [s$^{-1}$]')
141 plt.xlim(-2.2,2.2)
142 plt.ylim(10,24)
143 plt.legend()
144 plt.minorticks_on()
145 plt.grid(b=True, which='minor', linestyle='--')
146 plt.grid(which='major',color='k')
147 plt.rc('axes', axisbelow=True)
```

```
148 plt.savefig('1linien.eps')
149 plt.show()
150
151 print(f' Gauss: mu = {round(mu_gauss,4)}+-{round(mu_err_gauss,4)} mm/s,
        ↪ sigma = {round(sigma_gauss,4)}+-{round(sigma_err_gauss,4)} mm/s, A =
        ↪ {round(A_gauss,4)}+-{round(A_err_gauss,4)} mm/s^2, B = {round(B_gauss
        ↪ ,4)}+-{round(B_err_gauss,4)} 1/s, red. chi^2={round(redchisq_gauss,3)
        ↪ }\n')
152 print(f'Lorentz: mu = {round(mu_lorentz,4)}+-{round(mu_err_lorentz,4)} mm/s
        ↪ , gamma = {round(gamma_lorentz,4)}+-{round(gamma_err_lorentz,4)} mm/s
        ↪ , A = {round(A_lorentz,4)}+-{round(A_err_lorentz,4)} mm/s^2, B = {
        ↪ round(B_lorentz,4)}+-{round(B_err_lorentz,4)} 1/s, red. chi^2={round(
        ↪ redchisq_lorentz,6)}, A={A_lorentz} \n')
153 print(f'Voigt: mu = {round(mu_voigt,5)}+-{round(mu_err_voigt,5)} mm/s,
        ↪ sigma = {round(sigma_voigt,4)}+-{round(sigma_err_voigt,4)} mm/s,gamma
        ↪  = {round(gamma_voigt,4)}+-{round(gamma_err_voigt,4)} mm/s, A = {
        ↪ round(A_voigt,4)}+-{round(A_err_voigt,4)} mm/s^2, B = {round(B_voigt
        ↪ ,4)}+-{round(B_err_voigt,4)} 1/s, red. chi^2={round(redchisq_voigt,6)
        ↪ }\n')
154
155 ############ Isomeric Shift ##############
156
157 E_gamma = 14.4*1000 #eV
158 c       = 2.99792458*10**11 #mm/s
159 reee    = E_gamma/c
160
161 print('-------------------------')
162 print('Isomeric Shift')
163 print(f'E_iso_gauss={round(mu_gauss*reee*10**9,1)}+-{round(mu_err_gauss*
        ↪ reee*10**9,1)}neV')
164 print(f'E_iso_lorentz={round(mu_lorentz*reee*10**9,1)}+-{round(
        ↪ mu_err_lorentz*reee*10**9,1)}neV')
165 print(f'E_iso_voigt={round(mu_voigt*reee*10**9,1)}+-{round(mu_err_voigt*
        ↪ reee*10**9,1)}neV\n')
166
167
168 ############ effective absorber thickness ##############
169 print('-------------------------')
170 print('effective Absorver thickness T_A')
171 f_A     = 0.8 #debye-waller aus anleitung
172 d_A     = 25*10**(-6) #meter
173 beta    = 0.022 # anteil von 57^Fe in Probe
174 f       = 0.7 #%, iron content in absorber
175 s_f     = 0.05 #%
176
177
178 ## sigma_0 berechnen
179 lambdaa = 0.0861*10**(-9)#meter
180 I_e     = 3/2 #spin excited state
181 I_g     = 1/2 #sping ground state
182 alpha   = 8.58
183 s_alpha = 0.18
184
185 sigma_0     = (lambdaa**2/(2*np.pi)) * (2*I_e+1)/(2*I_g+1) * 1/(1+alpha)# m
        ↪ ^2
186 s_sigma_0   = sigma_0*s_alpha/(1+alpha)
187 print(f'sigma={round(sigma_0*10**(24),0)}+-{round(s_sigma_0*10**(24),0)}
        ↪ 10^-24 m^2')
```

```
188
189 rho = 7874 #kg/m^3
190 M   = 55.845*10**(-3) # kg/mol
191 s_M = 0.002*10**(-3) #kg/mol
192 N_A = 6.02214076 * 10**(23) #1/mol avogadro
193
194
195 n_A     = rho*(N_A/M)*f
196 s_n_A   = n_A*np.sqrt((s_f/f)**2 + (s_M/M)**2)
197 print(f'n_A={round(n_A*10**(-28),1)}+-{round(s_n_A*10**(-28),1)} *10^28 m
    ↪ ^-3')
198
199
200 T_A   = f_A*n_A*beta*sigma_0*d_A
201 s_T_A = T_A*np.sqrt((s_n_A/n_A)**2 + (s_sigma_0/sigma_0)**2)
202
203 print(f'T_A={round(T_A,1)}+-{round(s_T_A,1)}\n')
204
205
206
207 ############## Debye-Waller Factor ##################
208 print('-----------------------------')
209
210 N_infty_gauss   = B_gauss
211 N_infty_lorentz = B_lorentz
212 N_infty_voigt   = B_voigt
213
214 s_N_infty_gauss   = B_err_gauss
215 s_N_infty_lorentz = B_err_lorentz
216 s_N_infty_voigt   = B_err_voigt
217
218
219 N_mu_gauss   = Gaussian(mu_gauss,mu_gauss,sigma_gauss,A_gauss,B_gauss)
220 N_mu_lorentz = Lorentz(mu_lorentz,mu_lorentz,gamma_lorentz,A_lorentz,
    ↪ B_lorentz)
221 N_mu_voigt   = Voigt(mu_voigt,mu_voigt, sigma_voigt,gamma_voigt,A_voigt,
    ↪ B_voigt)
222
223 s_N_mu_gauss   = Gaussian(mu_gauss-mu_err_gauss,mu_gauss,sigma_gauss,
    ↪ A_gauss,B_gauss) - N_mu_gauss
224 s_N_mu_lorentz = Lorentz(mu_lorentz-mu_err_lorentz,mu_lorentz,gamma_lorentz
    ↪ ,A_lorentz,B_lorentz) - N_mu_lorentz
225 s_N_mu_voigt   = Voigt(mu_voigt-mu_err_voigt,mu_voigt, sigma_voigt,
    ↪ gamma_voigt,A_voigt,B_voigt) - N_mu_voigt
226
227
228
229 f_Q_gauss   = (N_infty_gauss-N_mu_gauss)/(N_infty_gauss*(1-np.exp(-T_A/2)*
    ↪ jv(0,1j* T_A/2)))
230 f_Q_lorentz = (N_infty_lorentz-N_mu_lorentz)/(N_infty_lorentz*(1-np.exp(-
    ↪ T_A/2)*jv(0,1j* T_A/2)))
231 f_Q_voigt   = (N_infty_voigt-N_mu_voigt)/(N_infty_voigt*(1-np.exp(-T_A/2)*
    ↪ jv(0,1j* T_A/2)))
232
233 s_f_Q_gauss = np.sqrt( (N_mu_gauss*s_N_infty_gauss/(N_infty_gauss**2*(1-np.
    ↪ exp(-T_A/2)*jv(0,1j* T_A/2))))**2 + (-s_N_mu_gauss/(N_infty_gauss*(1-
    ↪ np.exp(-T_A/2)*jv(0,1j*T_A/2))))**2 + (-np.exp(-T_A/2)*(jv(0,1j* T_A
    ↪ /2)+1j*jv(1,1j*T_A/2))*s_T_A/(2*(np.exp(-T_A/2)-jv(0,1j*T_A/2))**2))
```

```
    ↪ **2 )
234 s_f_Q_lorentz = np.sqrt( (N_mu_lorentz*s_N_infty_lorentz/(N_infty_lorentz
    ↪ **2*(1-np.exp(-T_A/2)*jv(0,1j* T_A/2))))**2 + (-s_N_mu_lorentz/(
    ↪ N_infty_lorentz*(1-np.exp(-T_A/2)*jv(0,1j*T_A/2))))**2 + (-np.exp(-
    ↪ T_A/2)*(jv(0,1j* T_A/2)+1j*jv(1,1j*T_A/2))*s_T_A/(2*(np.exp(-T_A/2)-
    ↪ jv(0,1j*T_A/2))**2))**2 )
235 s_f_Q_voigt = np.sqrt( (N_mu_voigt*s_N_infty_voigt/(N_infty_voigt**2*(1-np.
    ↪ exp(-T_A/2)*jv(0,1j* T_A/2))))**2 + (-s_N_mu_voigt/(N_infty_voigt*(1-
    ↪ np.exp(-T_A/2)*jv(0,1j*T_A/2))))**2 + (-np.exp(-T_A/2)*(jv(0,1j* T_A
    ↪ /2)+1j*jv(1,1j*T_A/2))*s_T_A/(2*(np.exp(-T_A/2)-jv(0,1j*T_A/2))**2))
    ↪ **2 )
236
237 print(f'Debye-Waller_Q Gaus: {round(np.real(f_Q_gauss),3)}+-{round(np.real(
    ↪ s_f_Q_gauss),3)}')
238 print(f'Debye-Waller_Q Lorentz: {round(np.real(f_Q_lorentz),3)}+-{round(np.
    ↪ real(s_f_Q_lorentz),3)}')
239 print(f'Debye-Waller_Q Voigt: {round(np.real(f_Q_voigt),3)}+-{round(np.real
    ↪ (s_f_Q_voigt),3)}\n')
240
241
242
243 ########## effective source thickness ############
244 print('-------------------------')
245 print('effective source thickness T_Q')
246
247 n_Q      = n_A
248 s_n_Q    = s_n_A
249 beta     = 1
250 d_Q      = 100*10**(-10) #meter, 100Angstrom also
251
252 T_Q_gauss        = f_Q_gauss*n_Q*beta*sigma_0*d_Q
253 s_T_Q_gauss      = T_Q_gauss*np.sqrt((s_n_Q/n_Q)**2 + (s_sigma_0/sigma_0)**2
    ↪  + (s_f_Q_gauss/f_Q_gauss)**2)
254 T_Q_lorentz      = f_Q_lorentz*n_Q*beta*sigma_0*d_Q
255 s_T_Q_lorentz    = T_Q_lorentz*np.sqrt((s_n_Q/n_Q)**2 + (s_sigma_0/sigma_0)
    ↪ **2 +(s_f_Q_lorentz/f_Q_lorentz)**2)
256 T_Q_voigt        = f_Q_voigt*n_Q*beta*sigma_0*d_Q
257 s_T_Q_voigt      = T_Q_voigt*np.sqrt((s_n_Q/n_Q)**2 + (s_sigma_0/sigma_0)**2
    ↪  + (s_f_Q_voigt/f_Q_voigt)**2)
258
259 print(f'T_Q Gauss: {round(np.real(T_Q_gauss),3)}+-{round(np.real(
    ↪ s_f_Q_gauss),3)}')
260 print(f'T_Q Lorentz: {round(np.real(T_Q_lorentz),3)}+-{round(np.real(
    ↪ s_f_Q_lorentz),3)}')
261 print(f'T_Q Voigt: {round(np.real(T_Q_voigt),3)}+-{round(np.real(
    ↪ s_f_Q_voigt),3)}\n')
262
263
264 ########## line width ############
265 print('-------------------------')
266 print('line width Gamma in mm/s')
267
268
269 Gamma_gauss      = 2*np.sqrt(2*np.log(2))*sigma_gauss #mm/s
270 s_Gamma_gauss    = 2*np.sqrt(2*np.log(2))*sigma_err_gauss
271 print(f'Gamma Gauss: {round(Gamma_gauss,3)}+-{round(s_Gamma_gauss,3)} mm/s'
    ↪ )
272
```

```
273 Gamma_lorentz    = 2*gamma_lorentz #mm/s
274 s_Gamma_lorentz = 2*gamma_err_lorentz
275 print(f'Gamma Lorentz: {round(Gamma_lorentz,2)}+-{round(s_Gamma_lorentz,2)}
        ↪  mm/s')
276
277 Gamma_voigt    = 2*gamma_voigt
278 s_Gamma_voigt = 2*gamma_err_voigt
279
280 print(f'Gamma Voigt: {round(Gamma_voigt,3)}+-{round(s_Gamma_voigt,3)} mm/s\
        ↪ n')
281
282 # umrechnen in Energie durch Doppler
283 print('-------------------------')
284 print('line width Gamma in neV')
285
286 gamma_g          = Gamma_gauss*reee #eV
287 s_gamma_g        = s_Gamma_gauss*reee
288 gamma_l          = Gamma_lorentz*reee #eV
289 s_gamma_l        = s_Gamma_lorentz*reee
290 gamma_v      = Gamma_voigt*reee #eV
291 s_gamma_v    = s_Gamma_voigt*reee
292
293 print(f'Gamma Gauss: {round(gamma_g*10**9,1)}+-{round(s_gamma_g*10**9,1)}
        ↪ neV')
294 print(f'Gamma Lorentz: {round(gamma_l*10**9,1)}+-{round(s_gamma_l*10**9,1)}
        ↪  neV')
295 print(f'Gamma Voigt: {round(gamma_v*10**9,0)}+-{round(s_gamma_v*10**9,0)}
        ↪ neV')
296
297
298 # in lifetime umrechnen
299 print('-------------------------')
300 print('lifetime tau')
301
302 hquer = 6.582119569 *10**(-16) #eVs
303
304 tau_g        = hquer/gamma_g #s
305 tau_l        = hquer/gamma_l
306 tau_v        = hquer/gamma_v
307
308 s_tau_g      = tau_g * s_gamma_g/gamma_g #s
309 s_tau_l      = tau_l * s_gamma_l/gamma_l
310 s_tau_v      = tau_v * s_gamma_v/gamma_v
311
312
313 print(f'tau Gauss: {round(tau_g*10**9,1)}+-{round(s_tau_g*10**9,1)} ns')
314 print(f'tau Lorentz: {round(tau_l*10**9,1)}+-{round(s_tau_l*10**9,1)} ns')
315 print(f'tau Voigt: {round(tau_v*10**9,0)}+-{round(s_tau_v*10**9,0)} ns')
316
317
318 # lifetime correction
319 print('-------------------------')
320 print('lifetime tau corrected')
321
322 rel_width    = 3.69
323 s_rel_width = 0.12
324
325 tau_cor_g        = rel_width * tau_g #s
```

```
326 tau_cor_l        = rel_width * tau_l
327 tau_cor_v        = rel_width * tau_v
328
329 s_tau_cor_g      = tau_cor_g*np.sqrt((s_rel_width/rel_width)**2 + (s_tau_g/
    ↪ tau_g)**2)
330 s_tau_cor_l      = tau_cor_l*np.sqrt((s_rel_width/rel_width)**2 + (s_tau_l/
    ↪ tau_l)**2)
331 s_tau_cor_v      = tau_cor_v*np.sqrt((s_rel_width/rel_width)**2 + (s_tau_v/
    ↪ tau_v)**2)
332
333 print(f'tau corr Gauss: {round(tau_cor_g*10**9,0)}+-{round(s_tau_cor_g
    ↪ *10**9,0)} ns')
334 print(f'tau corr Lorentz: {round(tau_cor_l*10**9,0)}+-{round(s_tau_cor_l
    ↪ *10**9,0)} ns')
335 print(f'tau corr Voigt: {round(tau_cor_v*10**9,0)}+-{round(s_tau_cor_v
    ↪ *10**9,0)} ns')
336
337
338 ###################### extra stuff ######################
339
340 print('gamma parameter from literature linewidth Gamma:', round
    ↪ ((4.7*10**(-9))/(reee*2),5), 'mm/s')
341
342 velocity, rate, s_rate = read_in('daten.csv',0,1,2)
343 plt.errorbar(velocity, rate, yerr=s_rate, fmt='x', linewidth=0.3, zorder=1,
    ↪ label='Data points')
344
345 x=np.linspace(velocity[b],velocity[a],9000)
346 plt.plot(x,Lorentz(x,mu_lorentz,gamma_lorentz,A_lorentz,B_lorentz),
    ↪ linestyle='-',linewidth='1.5', color='k', zorder=2, label='Lorentz
    ↪ fit')
347 plt.plot(x,Lorentz(x,mu_lorentz,gamma_lorentz/3.69,A_lorentz/3.69,B_lorentz
    ↪ ),linestyle='-',linewidth='1.5', color='lime', zorder=3, label='
    ↪ Corrected Lorentz fit')
348 plt.plot(x,Lorentz(x,mu_lorentz,0.04892446363194445,A_lorentz/gamma_lorentz
    ↪ *0.04892446363194445,B_lorentz),label='Theoretical Lorentz',color='r'
    ↪  ,linewidth='1.5',zorder=4)
349
350
351 plt.title('Stainless steel absorber M  bauer spectrum')
352 plt.xlabel(r'$v$ [mm$\,$s$^{-1}$]')
353 plt.ylabel(r'$\dot{N}$ [s$^{-1}$]')
354 plt.xlim(-2.2,2.2)
355 plt.ylim(10,24)
356 plt.legend()
357 plt.minorticks_on()
358 plt.grid(b=True, which='minor', linestyle='--')
359 plt.grid(which='major',color='k')
360 plt.rc('axes', axisbelow=True)
361
362 plt.savefig('resolution.eps')
363 plt.show()
```

G.7. Natural Iron Absorber

Data Processing

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4
5
6  def read_in(name,col1,col2,col3):
7          a = np.array(np.genfromtxt(np.str(name),usecols=col1,dtype=np.float
               ↪ ,delimiter=",",skip_header=0,skip_footer=0))
8          b = np.array(np.genfromtxt(np.str(name),usecols=col2,dtype=np.float
               ↪ ,delimiter=",",skip_header=0,skip_footer=0))
9          c = np.array(np.genfromtxt(np.str(name),usecols=col3,dtype=np.float
               ↪ ,delimiter=",",skip_header=0,skip_footer=0))
10         return [a,b,c]
11
12 def read_in2(name,col1,col2):
13         a = np.array(np.genfromtxt(np.str(name),usecols=col1,dtype=np.float
               ↪ ,delimiter=",",skip_header=1,skip_footer=0))
14         b = np.array(np.genfromtxt(np.str(name),usecols=col2,dtype=np.float
               ↪ ,delimiter=",",skip_header=1,skip_footer=0))
15         return [a,b]
16
17 compton_back,compton_back_err = read_in2('compton-back.csv',0,1)
18 velocity2,time2,counts2 = read_in('6linien_2.csv',0,1,2)
19
20 velocity = []
21 time = []
22 counts = []
23
24 i=0
25 while i < len(velocity2)-1:
26     if velocity2[i]==velocity2[i+1]:
27         if velocity2[i+1]==velocity2[i+2]:
28             if velocity2[i+2]==velocity2[i+3]:
29                 if velocity2[i+3]==velocity2[i+4]:
30                     if velocity2[i+4]==velocity2[i+5]:
31                         velocity.append(velocity2[i])
32                         time.append(time2[i]+time2[i+1]+time2[i+2]+time2[i
                               ↪ +3]+time2[i+4]+time2[i+5])
33                         counts.append(counts2[i]+counts2[i+1]+counts2[i+2]+
                               ↪ counts2[i+3]+counts2[i+4]+counts2[i+5])
34                         i=i+6
35                     else:
36                         velocity.append(velocity2[i])
37                         time.append(time2[i]+time2[i+1]+time2[i+2]+time2[i
                               ↪ +3]+time2[i+4])
38                         counts.append(counts2[i]+counts2[i+1]+counts2[i+2]+
                               ↪ counts2[i+3]+counts2[i+4])
39                         i=i+5
40                 else:
41                     velocity.append(velocity2[i])
42                     time.append(time2[i]+time2[i+1]+time2[i+2]+time2[i+3])
43                     counts.append(counts2[i]+counts2[i+1]+counts2[i+2]+
                           ↪ counts2[i+3])
44                     i=i+4
45             else:
46                 velocity.append(velocity2[i])
47                 time.append(time2[i]+time2[i+1]+time2[i+2])
```

```
48                      counts.append(counts2[i]+counts2[i+1]+counts2[i+2])
49                      i=i+3
50              else:
51                  velocity.append(velocity2[i])
52                  time.append(time2[i]+time2[i+1])
53                  counts.append(counts2[i]+counts2[i+1])
54                  i=i+2
55          else:
56              velocity.append(velocity2[i])
57              time.append(time2[i])
58              counts.append(counts2[i])
59              i=i+1
60
61  rate = []
62  s_rate = []
63  for i in range(0,len(velocity)):
64      rate.append(counts[i]/(time[i]/1000))
65      s_rate.append(np.sqrt(counts[i])/(time[i]/1000))
66
67  err_korrigiert = []
68  for i in range(0,len(s_rate)):
69      err_korrigiert.append(np.sqrt(s_rate[i]**2 + compton_back_err**2))
70
71  T = 0.556 #korrigierte transmission
72  s_T = 0.007
73
74  rate_korrigiert = []
75  s_rate_korrigiert = []
76  for i in range(0,len(rate)):
77      rate_korrigiert.append((rate[i]-compton_back)/T)
78      s_rate_korrigiert.append(np.sqrt( (s_rate[i]/T)**2 + (compton_back_err/
            ↪ T)**2 + ((rate[i]-compton_back)*s_T/T**2)**2) )
79
80  plt.errorbar(velocity, rate_korrigiert, yerr=s_rate_korrigiert, fmt='x',
        ↪ linewidth=0.3)
81  plt.title('corrected 6 line absorber spectrum')
82  plt.xlabel('Velocity [mm/s]')
83  plt.ylabel('Rate [1/s]')
84  plt.grid()
85  plt.show()
86
87
88  #daten speichern in ein file
89  L = []
90  L.append('Velocity [mm/s], corr Rate [1/s], s_corr_Rate [1/s]\n')
91  for i in range(0,len(velocity)):
92      L.append(f'{velocity[i]},{round(rate_korrigiert[i],3)},{round(
            ↪ s_rate_korrigiert[i],3)}\n')
93
94  file=open(r'daten_2.csv','w+')
95  file.writelines(L)
96  file.close()
```

Evaluation

```
1  import numpy as np
2  import matplotlib.pyplot as plt
```

```
3  from scipy.optimize import curve_fit
4  from scipy.special import wofz
5  from scipy.special import jv
6
7  plt.rcParams.update({'axes.titlesize': 'xx-large'})
8  plt.rcParams.update({'axes.labelsize': 'xx-large'})
9  plt.rcParams.update({'xtick.labelsize': 'x-large'})
10 plt.rcParams.update({'ytick.labelsize': 'x-large'})
11 plt.rcParams.update({'legend.fontsize': 'large'})
12
13 def read_in(name,col1,col2,col3):
14         a = np.array(np.genfromtxt(np.str(name),usecols=col1,dtype=np.float
       ↪ ,delimiter=",",skip_header=1,skip_footer=0))
15         b = np.array(np.genfromtxt(np.str(name),usecols=col2,dtype=np.float
       ↪ ,delimiter=",",skip_header=1,skip_footer=0))
16         c = np.array(np.genfromtxt(np.str(name),usecols=col3,dtype=np.float
       ↪ ,delimiter=",",skip_header=1,skip_footer=0))
17         return [a,b,c]
18
19 def Gaussian(x, mu, sigma, A, B):
20     return -A/np.sqrt(2*np.pi*sigma**2) * np.exp(-0.5*((x-mu)/sigma)**2) +
       ↪ B
21
22 def Lorentz(x,mu,gamma,A,B):
23     return -A/np.pi * gamma/((x-mu)**2 + gamma**2) + B
24
25 def Voigt(x,mu,sigma,gamma,A,B):
26     return -A*np.real(wofz((x-mu+1j*gamma)/(sigma*np.sqrt(2))))/(sigma*np.
       ↪ sqrt(2*np.pi)) + B
27
28
29 ############ Gauss fit ###############
30 velocity, rate, s_rate = read_in('daten.csv',0,1,2)
31
32 #plt.errorbar(velocity, rate, yerr=s_rate, fmt='x', linewidth=0.3, color='
       ↪ royalblue', zorder=1, label='Data points')
33 #plt.show()
34
35 start    = [40    ,140,230,295,370,460] #start daten eingeben
36 end      = [130   ,215,290,350,440,545] # end daten eingeben
37
38
39 p0_gauss = [[-5,0.5,2,20], [-3,0.5,4,20], [-1,0.5,2,20], [1,0.5,2,20],
       ↪ [3,0.5,3,20], [5,0.6,4,20]] #fit tips
40
41 redchisq_gauss = []
42 mu_gauss = []
43 sigma_gauss = []
44 A_gauss = []
45 B_gauss = []
46 mu_err_gauss = []
47 sigma_err_gauss = []
48 A_err_gauss = []
49 B_err_gauss = []
50
51 for i in range(0,6):
52     a=start[i]
53     b=end[i]
```

```
54
55      velo_gauss = velocity[a:b]
56      rate_gauss = rate[a:b]
57      s_rate_gauss = s_rate[a:b]
58
59      fitPara, fitCova = curve_fit(Gaussian, velo_gauss, rate_gauss, sigma=
            ↪ s_rate_gauss, p0=p0_gauss[i])
60
61      mu_gauss.append(fitPara[0])
62      sigma_gauss.append(fitPara[1])
63      A_gauss.append(fitPara[2])
64      B_gauss.append(fitPara[3])
65      mu_err_gauss.append(np.sqrt(fitCova[0][0]))
66      sigma_err_gauss.append(np.sqrt(fitCova[1][1]))
67      A_err_gauss.append(np.sqrt(fitCova[2][2]))
68      B_err_gauss.append(np.sqrt(fitCova[3][3]))
69
70      x=np.linspace(velocity[a],velocity[b],1000)
71      #plt.plot(x,Gaussian(x,mu_gauss[i],sigma_gauss[i],A_gauss[i],B_gauss[i
            ↪ ]), linewidth='1', color='r', label=f'Gaussian fit {i}', zorder
            ↪ =2)
72
73      DOS=len(velo_gauss)-4
74      r = []
75      for j in range(0,len(velo_gauss)):
76          r.append((rate_gauss[j]-Gaussian(velo_gauss[j],mu_gauss[i],
                ↪ sigma_gauss[i],A_gauss[i],B_gauss[i]))/s_rate_gauss[j])
77      redchisq_gauss.append(sum(r)/DOS)
78
79  #plt.legend()
80  #plt.show()
81
82
83  ###################### Loretz Fit ############################
84  velocity, rate, s_rate = read_in('daten.csv',0,1,2)
85
86  start   = [40   ,140,230,295,370,460] #start daten eingeben
87  end     = [130  ,215,290,350,440,545] # end daten eingeben
88
89
90  p0_lorentz = [[-5,0.5,2,20], [-3,0.5,4,20], [-1,0.5,2,20], [1,0.5,2,20],
        ↪ [3,0.5,3,20], [5,0.6,4,20]] #fit tips
91
92  redchisq_lorentz = []
93  mu_lorentz = []
94  gamma_lorentz = []
95  A_lorentz = []
96  B_lorentz = []
97  mu_err_lorentz = []
98  gamma_err_lorentz = []
99  A_err_lorentz = []
100 B_err_lorentz = []
101
102 for i in range(0,6):
103     a=start[i]
104     b=end[i]
105
106     velo_lorentz = velocity[a:b]
```

```
107     rate_lorentz = rate[a:b]
108     s_rate_lorentz = s_rate[a:b]
109
110     fitPara, fitCova = curve_fit(Lorentz, velo_lorentz, rate_lorentz, sigma
            ↪ =s_rate_lorentz, p0=p0_lorentz[i])
111
112     mu_lorentz.append(fitPara[0])
113     gamma_lorentz.append(fitPara[1])
114     A_lorentz.append(fitPara[2])
115     B_lorentz.append(fitPara[3])
116     mu_err_lorentz.append(np.sqrt(fitCova[0][0]))
117     gamma_err_lorentz.append(np.sqrt(fitCova[1][1]))
118     A_err_lorentz.append(np.sqrt(fitCova[2][2]))
119     B_err_lorentz.append(np.sqrt(fitCova[3][3]))
120
121     x=np.linspace(velocity[a],velocity[b],1000)
122     #plt.plot(x,Lorentz(x,mu_lorentz[i],gamma_lorentz[i],A_lorentz[i],
            ↪ B_lorentz[i]),linestyle='-',linewidth='1', color='k',label=f'
            ↪ Lorentz fit {i}', zorder=3)
123
124     DOS=len(velo_lorentz)-4
125     r = []
126     for j in range(0,len(velo_lorentz)):
127         r.append((rate_lorentz[j]-Lorentz(velo_lorentz[j],mu_lorentz[i],
                ↪ gamma_lorentz[i],A_lorentz[i],B_lorentz[i]))/s_rate_lorentz[j
                ↪ ])
128     redchisq_lorentz.append(sum(r)/DOS)
129
130 #plt.legend()
131 #plt.show()
132
133
134 ###################### Voigt Fit ############################
135 velocity, rate, s_rate = read_in('daten.csv',0,1,2)
136
137 start   = [60   ,140,230,300,350,452] #start daten eingeben
138 end     = [130  ,215,300,350,455,550] # end daten eingeben
139
140
141 p0_voigt = [[-5.8,0.01,0.5,4.2,20], [-3,0.01,0.5,4,20],
        ↪ [-0.5,0.01,0.5,2,20], [0.8,0.01,0.5,4,21], [3.5,0.01,0.5,4,20],
        ↪ [5,0.01,0.6,4,20]] #fit tips
142
143 redchisq_voigt = []
144 mu_voigt = []
145 sigma_voigt = []
146 gamma_voigt = []
147 A_voigt = []
148 B_voigt = []
149 mu_err_voigt = []
150 sigma_err_voigt = []
151 gamma_err_voigt = []
152 A_err_voigt = []
153 B_err_voigt = []
154
155 for i in range(0,6):
156     a=start[i]
157     b=end[i]
```

```
158
159     velo_voigt = velocity[a:b]
160     rate_voigt = rate[a:b]
161     s_rate_voigt = s_rate[a:b]
162
163     fitPara, fitCova = curve_fit(Voigt, velo_voigt, rate_voigt, sigma=
            ↪ s_rate_voigt, p0=p0_voigt[i])
164
165     mu_voigt.append(fitPara[0])
166     sigma_voigt.append(fitPara[1])
167     gamma_voigt.append(fitPara[2])
168     A_voigt.append(fitPara[3])
169     B_voigt.append(fitPara[4])
170     mu_err_voigt.append(np.sqrt(fitCova[0][0]))
171     sigma_err_voigt.append(np.sqrt(fitCova[1][1]))
172     gamma_err_voigt.append(np.sqrt(fitCova[2][2]))
173     A_err_voigt.append(np.sqrt(fitCova[3][3]))
174     B_err_voigt.append(np.sqrt(fitCova[4][4]))
175
176     x=np.linspace(velocity[a],velocity[b],1000)
177     #plt.plot(x,Voigt(x,mu_voigt[i],sigma_voigt[i],gamma_voigt[i],A_voigt[i
            ↪ ],B_voigt[i]),dashes=(3,4), linewidth='2', color='lime', zorder
            ↪ =4, label=f'Voigt fit {i}')
178
179     DOS=len(velo_voigt)-4
180     r = []
181     for j in range(0,len(velo_voigt)):
182         r.append((rate_voigt[j]-Voigt(velo_voigt[j],mu_voigt[i],sigma_voigt
                ↪ [i],gamma_voigt[i],A_voigt[i],B_voigt[i]))/s_rate_voigt[j])
183     redchisq_voigt.append(sum(r)/DOS)
184
185 #plt.legend()
186 #plt.show()
187
188
189 #print(f'{redchisq_gauss}')
190 #print(f'{redchisq_lorentz}')
191 #print(f'{redchisq_voigt}')
192
193
194 #plt.errorbar(velocity, rate, yerr=s_rate, fmt='x', linewidth=0.3, color='
        ↪ royalblue', zorder=1, label='Data points')
195 #x = np.linspace(-8,8,5000)
196 #for i in range(0,6):
197     #plt.plot(x, Gaussian(x,mu_gauss[i],sigma_gauss[i],A_gauss[i],B_gauss[i
            ↪ ]))
198     #plt.plot(x, Lorentz(x,mu_lorentz[i],gamma_lorentz[i],A_lorentz[i],
            ↪ B_lorentz[i]))
199     #plt.plot(x, Voigt(x,mu_voigt[i],sigma_voigt[i],gamma_voigt[i],A_voigt[
            ↪ i],B_voigt[i]))
200 #plt.show()
201
202
203
204
205
206 #### six fold version
207
```

```
208 velocity, rate, s_rate = read_in('daten_2.csv',0,1,2)
209 plt.errorbar(velocity, rate, yerr=s_rate, fmt='x', linewidth=0.3, color='
    ↪ royalblue', zorder=1, label='Data points')
210
211
212
213 #### multi gauss fit
214
215 def multi_gauss(x,mu1,mu2,mu3,mu4,mu5,mu6,sigma1,sigma2,sigma3,sigma4,
    ↪ sigma5,sigma6,A1,A2,A3,A4,A5,A6,B):
216     return -A1/np.sqrt(2*np.pi*sigma1**2) * np.exp(-0.5*((x-mu1)/sigma1)
        ↪ **2)-A2/np.sqrt(2*np.pi*sigma2**2) * np.exp(-0.5*((x-mu2)/sigma2)
        ↪ **2)-A3/np.sqrt(2*np.pi*sigma3**2) * np.exp(-0.5*((x-mu3)/sigma3)
        ↪ **2)-A4/np.sqrt(2*np.pi*sigma4**2) * np.exp(-0.5*((x-mu4)/sigma4)
        ↪ **2)-A5/np.sqrt(2*np.pi*sigma5**2) * np.exp(-0.5*((x-mu5)/sigma5)
        ↪ **2)-A6/np.sqrt(2*np.pi*sigma6**2) * np.exp(-0.5*((x-mu6)/sigma6)
        ↪ **2) + B
217
218
219 x = np.linspace(-8,8,5000)
220 p0=(mu_gauss[0],mu_gauss[1],mu_gauss[2],mu_gauss[3],mu_gauss[4],mu_gauss
    ↪ [5],sigma_gauss[0],sigma_gauss[1],sigma_gauss[2],sigma_gauss[3],
    ↪ sigma_gauss[4],sigma_gauss[5],A_gauss[0],A_gauss[1],A_gauss[2],
    ↪ A_gauss[3],A_gauss[4],A_gauss[5],20.5)
221
222 fitPara_gauss, fitCova_gauss = curve_fit(multi_gauss, velocity, rate, sigma
    ↪ =s_rate, p0=p0)
223 plt.plot(x, multi_gauss(x, *fitPara_gauss),'r', label='Gaussian fit')
224
225 DOS_multi_gauss = len(velocity)-19
226 r = []
227 for i in range(0,len(velocity)):
228     r.append(((rate[i]-multi_gauss(velocity[i],*fitPara_gauss))/s_rate[i])
        ↪ **2)
229 redchisq_multi_gauss = sum(r)/DOS_multi_gauss
230 print(redchisq_multi_gauss)
231
232
233
234 #### multi lorentz fit
235
236 def multi_lorentz(x,mu1,mu2,mu3,mu4,mu5,mu6,gamma1,gamma2,gamma3,gamma4,
    ↪ gamma5,gamma6,A1,A2,A3,A4,A5,A6,B):
237     return -A1/np.pi * gamma1/((x-mu1)**2 + gamma1**2)-A2/np.pi * gamma2/((
        ↪ x-mu2)**2 + gamma2**2)-A3/np.pi * gamma3/((x-mu3)**2 + gamma3**2)
        ↪ -A4/np.pi * gamma4/((x-mu4)**2 + gamma4**2)-A5/np.pi * gamma5/((x
        ↪ -mu5)**2 + gamma5**2)-A6/np.pi * gamma6/((x-mu6)**2 + gamma6**2)
        ↪ + B
238
239 p0=(mu_lorentz[0],mu_lorentz[1],mu_lorentz[2],mu_lorentz[3],mu_lorentz[4],
    ↪ mu_lorentz[5],gamma_lorentz[0],gamma_lorentz[1],gamma_lorentz[2],
    ↪ gamma_lorentz[3],gamma_lorentz[4],gamma_lorentz[5],A_lorentz[0],
    ↪ A_lorentz[1],A_lorentz[2],A_lorentz[3],A_lorentz[4],A_lorentz
    ↪ [5],20.5)
240 fitPara_lorentz, fitCova_lorentz = curve_fit(multi_lorentz, velocity, rate,
    ↪  sigma=s_rate, p0=p0)
241 plt.plot(x, multi_lorentz(x, *fitPara_lorentz),'k', label='Lorentz fit')
242
```

```
243 DOS_multi_lorentz = len(velocity)-19
244 r = []
245 for i in range(0,len(velocity)):
246     r.append(((rate[i]-multi_lorentz(velocity[i],*fitPara_lorentz))/s_rate[
        ↪ i])**2)
247 redchisq_multi_lorentz = sum(r)/DOS_multi_lorentz
248 print(redchisq_multi_lorentz)
249
250
251
252 #### multi voigt fit
253
254 def multi_voigt(x,mu1,mu2,mu3,mu4,mu5,mu6,sigma1,sigma2,sigma3,sigma4,
    ↪ sigma5,
255                 sigma6,gamma1,gamma2,gamma3,gamma4,gamma5,gamma6,A1,A2,A3,
                        ↪ A4,A5,A6,B):
256     return -A1*np.real(wofz((x-mu1+1j*gamma1)/(sigma1*np.sqrt(2))))/(sigma1
        ↪ *np.sqrt(2*np.pi))-A2*np.real(wofz((x-mu2+1j*gamma2)/(sigma2*np.
        ↪ sqrt(2))))/(sigma2*np.sqrt(2*np.pi))-A3*np.real(wofz((x-mu3+1j*
        ↪ gamma3)/(sigma3*np.sqrt(2))))/(sigma3*np.sqrt(2*np.pi))-A4*np.
        ↪ real(wofz((x-mu4+1j*gamma4)/(sigma4*np.sqrt(2))))/(sigma4*np.sqrt
        ↪ (2*np.pi))-A5*np.real(wofz((x-mu5+1j*gamma5)/(sigma5*np.sqrt(2)))
        ↪ )/(sigma5*np.sqrt(2*np.pi))-A6*np.real(wofz((x-mu6+1j*gamma6)/(
        ↪ sigma6*np.sqrt(2))))/(sigma6*np.sqrt(2*np.pi)) + B
257
258
259
260 p0=(mu_voigt[0],mu_voigt[1],mu_voigt[2],mu_voigt[3],mu_voigt[4],mu_voigt
    ↪ [5],sigma_voigt[0],sigma_voigt[1],sigma_voigt[2],sigma_voigt[3],
    ↪ sigma_voigt[4],sigma_voigt[5],gamma_voigt[0],gamma_voigt[1],
    ↪ gamma_voigt[2],gamma_voigt[3],gamma_voigt[4],gamma_voigt[5],A_voigt
    ↪ [0],A_voigt[1],A_voigt[2]+1,A_voigt[3]+1,A_voigt[4]+1,A_voigt
    ↪ [5],20.9)
261
262 fitPara_voigt, fitCova_voigt = curve_fit(multi_voigt, velocity, rate, sigma
    ↪ =s_rate, p0=p0)
263 plt.plot(x, multi_voigt(x, *fitPara_voigt),color='lime',dashes=(3,4),label=
    ↪ 'Voigt fit')
264
265
266 DOS_multi_voigt = len(velocity)-25
267 r = []
268 for i in range(0,len(velocity)):
269     r.append(((rate[i]-multi_voigt(velocity[i],*fitPara_voigt))/s_rate[i])
        ↪ **2)
270 redchisq_multi_voigt= sum(r)/DOS_multi_voigt
271 print(redchisq_multi_voigt)
272
273
274 plt.minorticks_on()
275 plt.grid(which='major',color='k', linewidth=0.5)
276 plt.grid(b=True, which='minor', linestyle='--', linewidth=0.5)
277 plt.rc('axes', axisbelow=True)
278 plt.xlabel(r'$v$ [mm$\,$s$^{-1}$]')
279 plt.ylabel(r'$\dot{N}$ [s$^{-1}$]')
280 plt.xlim(-8.5,8.5)
281 plt.ylim(15,22)
282 plt.title('Natural iron absorber M  bauer spectrum')
```

```
283 plt.legend()
284 plt.savefig('6linien.eps')
285 plt.show()
286
287
288
289
290 ##################### residuen plots ####################
291
292
293 #Gaussian
294 resi = []
295 s_resi = []
296 resi_velo = []
297
298 eins_sigma = []
299 s_eins_sigma = []
300 eins_sigma_velo = []
301 zwei_sigma = []
302 s_zwei_sigma = []
303 zwei_sigma_velo = []
304
305 for i in range(0,len(velocity)):
306     if (abs(rate[i]-multi_gauss(velocity[i],*fitPara_gauss))-s_rate[i])>0:
307         if (abs(rate[i]-multi_gauss(velocity[i],*fitPara_gauss))-2*s_rate[i
            ↪ ])>0:
308             zwei_sigma.append(rate[i]-multi_gauss(velocity[i],*
                ↪ fitPara_gauss))
309             zwei_sigma_velo.append(velocity[i])
310             s_zwei_sigma.append(s_rate[i])
311         else:
312             eins_sigma.append(rate[i]-multi_gauss(velocity[i],*
                ↪ fitPara_gauss))
313             eins_sigma_velo.append(velocity[i])
314             s_eins_sigma.append(s_rate[i])
315     else:
316         resi.append(rate[i]-multi_gauss(velocity[i],*fitPara_gauss))
317         resi_velo.append(velocity[i])
318         s_resi.append(s_rate[i])
319 plt.errorbar(resi_velo, resi, yerr=s_resi, fmt='x',color='cornflowerblue',
    ↪ linewidth=0.3, zorder=1)
320 plt.errorbar(eins_sigma_velo, eins_sigma, yerr=s_eins_sigma, fmt='x',color=
    ↪ 'darkorange',linewidth=0.3, zorder=1)
321 plt.errorbar(zwei_sigma_velo, zwei_sigma, yerr=s_zwei_sigma, fmt='x',color=
    ↪ 'fuchsia',linewidth=0.3, zorder=1)
322 plt.axhline(0,color='r', zorder=2)
323 plt.title('Gaussian')
324 plt.xlabel(r'$v$ [mm$\,$s$^{-1}$]')
325 plt.ylabel(r'Residual [s$^{-1}$]')
326 plt.xlim(-8.5,8.5)
327 plt.savefig('resi-gauss.eps')
328 plt.show()
329
330
331 #Lorentz
332 resi = []
333 s_resi = []
334 resi_velo = []
```

```
335
336 eins_sigma = []
337 s_eins_sigma = []
338 eins_sigma_velo = []
339 zwei_sigma = []
340 s_zwei_sigma = []
341 zwei_sigma_velo = []
342
343 for i in range(0,len(velocity)):
344     if (abs(rate[i]-multi_lorentz(velocity[i],*fitPara_lorentz))-s_rate[i])
        ↪ >0:
345         if (abs(rate[i]-multi_lorentz(velocity[i],*fitPara_lorentz))-2*
            ↪ s_rate[i])>0:
346             zwei_sigma.append(rate[i]-multi_lorentz(velocity[i],*
                ↪ fitPara_lorentz))
347             zwei_sigma_velo.append(velocity[i])
348             s_zwei_sigma.append(s_rate[i])
349         else:
350             eins_sigma.append(rate[i]-multi_lorentz(velocity[i],*
                ↪ fitPara_lorentz))
351             eins_sigma_velo.append(velocity[i])
352             s_eins_sigma.append(s_rate[i])
353     else:
354         resi.append(rate[i]-multi_lorentz(velocity[i],*fitPara_lorentz))
355         resi_velo.append(velocity[i])
356         s_resi.append(s_rate[i])
357 plt.errorbar(resi_velo, resi, yerr=s_resi, fmt='x',color='cornflowerblue',
    ↪ linewidth=0.3, zorder=1)
358 plt.errorbar(eins_sigma_velo, eins_sigma, yerr=s_eins_sigma, fmt='x',color=
    ↪ 'darkorange',linewidth=0.3, zorder=1)
359 plt.errorbar(zwei_sigma_velo, zwei_sigma, yerr=s_zwei_sigma, fmt='x',color=
    ↪ 'fuchsia',linewidth=0.3, zorder=1)
360 plt.axhline(0,color='k', zorder=2)
361 plt.title('Lorentz')
362 plt.xlabel(r'$v$ [mm$\,$s$^{-1}$]')
363 plt.ylabel(r'Residual [s$^{-1}$]')
364 plt.xlim(-8.5,8.5)
365 plt.savefig('resi-lorentz.eps')
366 plt.show()
367
368
369 #Voigt
370 resi = []
371 s_resi = []
372 resi_velo = []
373
374 eins_sigma = []
375 s_eins_sigma = []
376 eins_sigma_velo = []
377 zwei_sigma = []
378 s_zwei_sigma = []
379 zwei_sigma_velo = []
380
381 for i in range(0,len(velocity)):
382     if (abs(rate[i]-multi_voigt(velocity[i],*fitPara_voigt))-s_rate[i])>0:
383         if (abs(rate[i]-multi_voigt(velocity[i],*fitPara_voigt))-2*s_rate[i
            ↪ ])>0:
```

```
384                 zwei_sigma.append(rate[i]-multi_voigt(velocity[i],*
                        ↪ fitPara_voigt))
385                 zwei_sigma_velo.append(velocity[i])
386                 s_zwei_sigma.append(s_rate[i])
387             else:
388                 eins_sigma.append(rate[i]-multi_voigt(velocity[i],*
                        ↪ fitPara_voigt))
389                 eins_sigma_velo.append(velocity[i])
390                 s_eins_sigma.append(s_rate[i])
391         else:
392             resi.append(rate[i]-multi_voigt(velocity[i],*fitPara_voigt))
393             resi_velo.append(velocity[i])
394             s_resi.append(s_rate[i])
395 plt.errorbar(resi_velo, resi, yerr=s_resi, fmt='x',color='cornflowerblue',
        ↪ linewidth=0.3, zorder=1)
396 plt.errorbar(eins_sigma_velo, eins_sigma, yerr=s_eins_sigma, fmt='x',color=
        ↪ 'darkorange',linewidth=0.3, zorder=1)
397 plt.errorbar(zwei_sigma_velo, zwei_sigma, yerr=s_zwei_sigma, fmt='x',color=
        ↪ 'fuchsia',linewidth=0.3, zorder=1)
398 plt.axhline(0,color='lime',dashes=(3,4), zorder=2)
399 plt.title('Voigt')
400 plt.xlabel(r'$v$ [mm$\,$s$^{-1}$]')
401 plt.ylabel(r'Residual [s$^{-1}$]')
402 plt.xlim(-8.5,8.5)
403 plt.savefig('resi-voigt.eps')
404 plt.show()
405
406
407 print('------------------------ fit Para Gauss ------------------------')
408 for i in range(0,6):
409     print(f'mu_{i+1}: {round(fitPara_gauss[i],4)}+-{round(np.sqrt(
            ↪ fitCova_gauss[i][i]),4)} mm/s')
410 for i in range(0,6):
411     print(f'sigma_{i+1}: {round(fitPara_gauss[i+6],4)}+-{round(np.sqrt(
            ↪ fitCova_gauss[i+6][i+6]),4)} mm/s')
412 for i in range(0,6):
413     print(f'A_{i+1}: {round(fitPara_gauss[i+12],2)}+-{round(np.sqrt(
            ↪ fitCova_gauss[i+12][i+12]),2)} 1/s')
414 print(f'B: {round(fitPara_gauss[-1],3)}+-{round(np.sqrt(fitCova_gauss
        ↪ [-1][-1]),3)} 1/s \n')
415
416 print('------------------------ fit Para Lorentz ------------------------')
417 for i in range(0,6):
418     print(f'mu_{i+1}: {round(fitPara_lorentz[i],4)}+-{round(np.sqrt(
            ↪ fitCova_lorentz[i][i]),4)} mm/s')
419 for i in range(0,6):
420     print(f'gamma_{i+1}: {round(fitPara_lorentz[i+6],4)}+-{round(np.sqrt(
            ↪ fitCova_lorentz[i+6][i+6]),4)} mm/s')
421 for i in range(0,6):
422     print(f'A_{i+1}: {round(fitPara_lorentz[i+12],2)}+-{round(np.sqrt(
            ↪ fitCova_lorentz[i+12][i+12]),2)} 1/s')
423 print(f'B: {round(fitPara_lorentz[-1],3)}+-{round(np.sqrt(fitCova_lorentz
        ↪ [-1][-1]),3)} 1/s \n')
424
425
426 print('------------------------ fit Para Voigt ------------------------')
427 for i in range(0,6):
```

```
428     print(f'mu_{i+1}: {round(fitPara_voigt[i],4)}+-{round(np.sqrt(
            ↪ fitCova_voigt[i][i]),4)} mm/s')
429 for i in range(0,6):
430     print(f'sigma_{i+1}: {round(fitPara_voigt[i+6],3)}+-{round(np.sqrt(
            ↪ fitCova_voigt[i+6][i+6]),3)} mm/s')
431 for i in range(0,6):
432     print(f'gamma_{i+1}: {round(fitPara_voigt[i+12],3)}+-{round(np.sqrt(
            ↪ fitCova_voigt[i+12][i+12]),3)} 1/s')
433 for i in range(0,6):
434     print(f'A_{i+1}: {round(fitPara_voigt[i+18],2)}+-{round(np.sqrt(
            ↪ fitCova_voigt[i+18][i+18]),2)} 1/s')
435 print(f'B: {round(fitPara_voigt[-1],3)}+-{round(fitCova_voigt[-1][-1],3)}
        ↪ 1/s \n')
436
437
438
439 ##################################################
440 print('---------------------- Iso Shift ----------------------\n')
441
442 ########## isomeric shift#########
443
444 E_gamma = 14.4*1000 #eV
445 c       = 2.99792458*10**11 #mm/s
446 reee    = E_gamma/c
447
448 iso_shift = [] #neV
449 s_iso_shift = []
450
451 ###### gauss ########
452
453 mu_gauss = []
454 s_mu_gauss = []
455 for i in range(0,6):
456     mu_gauss.append(fitPara_gauss[i])
457     s_mu_gauss.append(np.sqrt(fitCova_gauss[i][i]))
458
459 print('-----------------Gauss-----------------')
460 for i in range(0,6):
461     print(f'Mu {i+1}: {round(mu_gauss[i],2)}+-{round(s_mu_gauss[i],2)} mm/s
            ↪ ')
462
463 print('')
464 iso_shift_gauss = []
465 s_iso_shift_gauss = []
466 for i in range(0,3):
467     iso_shift_gauss.append((mu_gauss[5-i]+mu_gauss[i])/2)
468     s_iso_shift_gauss.append(np.sqrt(s_mu_gauss[5-i]**2+s_mu_gauss[i]**2)
            ↪ /2)
469
470 for i in range(0,3):
471     print(f'Iso Shift {i+1}: {round(iso_shift_gauss[i],3)}+-{round(
            ↪ s_iso_shift_gauss[i],3)} mm/s')
472
473 for i in range(0,3):
474     print(f'Iso Shift {i+1}: {round(reee*iso_shift_gauss[i]*10**9,1)}+-{
            ↪ round(reee*s_iso_shift_gauss[i]*10**9,1)} neV')
475
476 a = []
```

```
477 b = []
478 for i in range(0,3):
479     a.append(iso_shift_gauss[i]/s_iso_shift_gauss[i]**2)
480     b.append(1/s_iso_shift_gauss[i]**2)
481 print('')
482
483 iso_shift.append(sum(a)/sum(b)*reee*10**9)
484 s_iso_shift.append(np.sqrt(1/sum(b))*reee*10**9)
485 print(f'Iso Shift gewichtet: {round(iso_shift[0],1)}+-{round(s_iso_shift
    ↪ [0],1)} neV')
486 print('\n')
487
488
489 ###### lorentz ########
490
491 mu_lorentz = []
492 s_mu_lorentz = []
493 for i in range(0,6):
494     mu_lorentz.append(fitPara_lorentz[i])
495     s_mu_lorentz.append(np.sqrt(fitCova_lorentz[i][i]))
496
497 print('----------------Lorentz----------------')
498 for i in range(0,6):
499     print(f'Mu {i+1}: {round(mu_lorentz[i],2)}+-{round(s_mu_lorentz[i],2)}
        ↪ mm/s')
500
501 print('')
502 iso_shift_lorentz = []
503 s_iso_shift_lorentz = []
504 for i in range(0,3):
505     iso_shift_lorentz.append((mu_lorentz[5-i]+mu_lorentz[i])/2)
506     s_iso_shift_lorentz.append(np.sqrt(s_mu_lorentz[5-i]**2+s_mu_lorentz[i]
507                                        **2)/2)
508
509
510 for i in range(0,3):
511     print(f'Iso Shift {i+1}: {round(iso_shift_lorentz[i],3)}+-{round(
        ↪ s_iso_shift_lorentz[i],3)} mm/s')
512
513 for i in range(0,3):
514     print(f'Iso Shift {i+1}: {round(reee*iso_shift_lorentz[i]*10**9,1)}+-{
        ↪ round(reee*s_iso_shift_lorentz[i]*10**9,1)} neV')
515
516 a = []
517 b = []
518 for i in range(0,3):
519     a.append(iso_shift_lorentz[i]/s_iso_shift_lorentz[i]**2)
520     b.append(1/s_iso_shift_lorentz[i]**2)
521 print('')
522
523 iso_shift.append(sum(a)/sum(b)*reee*10**9)
524 s_iso_shift.append(np.sqrt(1/sum(b))*reee*10**9)
525 print(f'Iso Shift gewichtet: {round(iso_shift[1],1)}+-{round(s_iso_shift
    ↪ [1],1)} neV')
526 print('\n')
527
528 ###### voigt ########
529
```

```
530 mu_voigt = []
531 s_mu_voigt = []
532 for i in range(0,6):
533     mu_voigt.append(fitPara_voigt[i])
534     s_mu_voigt.append(np.sqrt(fitCova_voigt[i][i]))
535
536 print('----------------Voigt----------------')
537 for i in range(0,6):
538     print(f'Mu {i+1}: {round(mu_voigt[i],2)}+-{round(s_mu_voigt[i],2)} mm/s
        ↪ ')
539
540 print('')
541 iso_shift_voigt = []
542 s_iso_shift_voigt = []
543 for i in range(0,3):
544     iso_shift_voigt.append((mu_voigt[5-i]+mu_voigt[i])/2)
545     s_iso_shift_voigt.append(np.sqrt(s_mu_voigt[5-i]**2+s_mu_voigt[i]**2)
        ↪ /2)
546
547
548 for i in range(0,3):
549     print(f'Iso Shift {i+1}: {round(iso_shift_voigt[i],3)}+-{round(
        ↪ s_iso_shift_voigt[i],3)} mm/s')
550
551 for i in range(0,3):
552     print(f'Iso Shift {i+1}: {round(reee*iso_shift_voigt[i]*10**9,1)}+-{
        ↪ round(reee*s_iso_shift_voigt[i]*10**9,1)} neV')
553
554 a = []
555 b = []
556 for i in range(0,3):
557     a.append(iso_shift_voigt[i]/s_iso_shift_voigt[i]**2)
558     b.append(1/s_iso_shift_voigt[i]**2)
559 print('')
560
561 iso_shift.append(sum(a)/sum(b)*reee*10**9) #neV
562 s_iso_shift.append(np.sqrt(1/sum(b))*reee*10**9)
563 print(f'Iso Shift gewichtet: {round(iso_shift[2],1)}+-{round(s_iso_shift
    ↪ [2],1)} neV')
564 print('\n')
565
566
567 ######## Momente, transition Energien und B-Felder
568 print('---------- Momente, transition Energien und B-Felder ----------\n')
569
570
571
572 ########### Gauss ###########
573 print('----------Gauss----------')
574 E_trans_gauss = [] #neV
575 s_E_trans_gauss = []
576 for i in range(0,6):
577     E_trans_gauss.append(mu_gauss[i]*reee*10**(9) - iso_shift[0])
578     s_E_trans_gauss.append(np.sqrt((s_mu_gauss[i]*reee*10**(9))**2 + (
        ↪ s_iso_shift[0])**2))
579
580 for i in range(0,6):
```

```
581     print(f'E-trans_{i+1}: {round(E_trans_gauss[i],1)}+-{round(
            ↪ s_E_trans_gauss[i],1)} neV')
582
583
584 E_trans_mittel_gauss = [] #neV
585 s_E_trans_mittel_gauss = []
586
587 for i in range(0,3):
588     E_trans_mittel_gauss.append((abs(E_trans_gauss[5-i])+abs(E_trans_gauss[
            ↪ i]))/2)
589     s_E_trans_mittel_gauss.append(np.sqrt(s_E_trans_gauss[5-i]**2+
            ↪ s_E_trans_gauss[i]**2)/2)
590 print('')
591 for i in range(0,3):
592     print(f'E-trans_mittel_{i+1}: {round(E_trans_mittel_gauss[i],1)}+-{
            ↪ round(s_E_trans_mittel_gauss[i],1)} neV')
593 print('\n')
594
595 ############ Lorentz ###########
596 print('----------Lorentz----------')
597 E_trans_lorentz = [] #neV
598 s_E_trans_lorentz = []
599 for i in range(0,6):
600     E_trans_lorentz.append(mu_lorentz[i]*reee*10**(9) - iso_shift[1])
601     s_E_trans_lorentz.append(np.sqrt((s_mu_lorentz[i]*reee*10**(9))**2 +(
            ↪ s_iso_shift[1])**2))
602
603 for i in range(0,6):
604     print(f'E-trans_{i+1}: {round(E_trans_lorentz[i],1)}+-{round(
            ↪ s_E_trans_lorentz[i],1)} neV')
605
606
607 E_trans_mittel_lorentz = [] #neV
608 s_E_trans_mittel_lorentz = []
609
610 for i in range(0,3):
611     E_trans_mittel_lorentz.append((abs(E_trans_lorentz[5-i])+abs(
            ↪ E_trans_lorentz[i]))/2)
612     s_E_trans_mittel_lorentz.append(np.sqrt(s_E_trans_lorentz[5-i]**2+
            ↪ s_E_trans_lorentz[i]**2)/2)
613 print('')
614 for i in range(0,3):
615     print(f'E-trans_mittel_{i+1}: {round(E_trans_mittel_lorentz[i],1)}+-{
            ↪ round(s_E_trans_mittel_lorentz[i],1)} neV')
616 print('\n')
617
618
619 ############ Voigt ###########
620 print('----------Voigt----------')
621 E_trans_voigt = [] #neV
622 s_E_trans_voigt = []
623 for i in range(0,6):
624     E_trans_voigt.append(mu_voigt[i]*reee*10**(9) - iso_shift[2])
625     s_E_trans_voigt.append(np.sqrt((s_mu_voigt[i]*reee*10**(9))**2 +(
            ↪ s_iso_shift[2])**2))
626
627 for i in range(0,6):
```

```
628     print(f'E-trans_{i+1}: {round(E_trans_voigt[i],1)}+-{round(
            ↪ s_E_trans_voigt[i],1)} neV')
629
630
631 E_trans_mittel_voigt = [] #neV
632 s_E_trans_mittel_voigt = []
633
634 for i in range(0,3):
635     E_trans_mittel_voigt.append((abs(E_trans_voigt[5-i])+abs(E_trans_voigt[
            ↪ i]))/2)
636     s_E_trans_mittel_voigt.append(np.sqrt(s_E_trans_voigt[5-i]**2+
            ↪ s_E_trans_voigt[i]**2)/2)
637 print('')
638 for i in range(0,3):
639     print(f'E-trans_mittel_{i+1}: {round(E_trans_mittel_voigt[i],1)}+-{
            ↪ round(s_E_trans_mittel_voigt[i],1)} neV')
640 print('\n')
641
642
643 print('------------------------ B-Feld ------------------------')
644 m_N = 3.15245*10**(-8)*10**(9) #nev/T
645 m_g = 0.09044*m_N
646
647
648 B_gauss = (E_trans_mittel_gauss[1]+E_trans_mittel_gauss[2])/(2*m_g)
649 s_B_gauss = np.sqrt((s_E_trans_mittel_gauss[1])**2 + (
        ↪ s_E_trans_mittel_gauss[2])**2)/(2*m_g)
650
651 B_lorentz = (E_trans_mittel_lorentz[1]+E_trans_mittel_lorentz[2])/(2*m_g)
652 s_B_lorentz = np.sqrt((s_E_trans_mittel_lorentz[1])**2 + (
        ↪ s_E_trans_mittel_lorentz[2])**2)/(2*m_g)
653
654 B_voigt = (E_trans_mittel_voigt[1]+E_trans_mittel_voigt[2])/(2*m_g)
655 s_B_voigt = np.sqrt((s_E_trans_mittel_voigt[1])**2 + (
        ↪ s_E_trans_mittel_voigt[2])**2)/(2*m_g)
656
657 print(f'B_gauss: {round(B_gauss,2)}+-{round(s_B_gauss,2)} T')
658 print(f'B_lorentz: {round(B_lorentz,2)}+-{round(s_B_lorentz,2)} T')
659 print(f'B_voigt: {round(B_voigt,2)}+-{round(s_B_voigt,2)} T\n')
660
661 print('---------------- magnetisches Moment mu_e ----------------')
662
663 m_e_gauss = (m_g - E_trans_mittel_gauss[0]/B_gauss)/m_N
664 s_m_e_gauss = np.sqrt( (s_E_trans_mittel_gauss[0]/B_gauss)**2 + (
        ↪ E_trans_mittel_gauss[0]*s_B_gauss/B_gauss**2)**2)/m_N
665
666 m_e_lorentz = (m_g - E_trans_mittel_lorentz[0]/B_lorentz)/m_N
667 s_m_e_lorentz = np.sqrt( (s_E_trans_mittel_lorentz[0]/B_lorentz)**2 + (
        ↪ E_trans_mittel_lorentz[0]*s_B_lorentz/B_lorentz**2)**2)/m_N
668
669 m_e_voigt = (m_g - E_trans_mittel_voigt[0]/B_voigt)/m_N
670 s_m_e_voigt = np.sqrt( (s_E_trans_mittel_voigt[0]/B_voigt)**2 + (
        ↪ E_trans_mittel_voigt[0]*s_B_voigt/B_voigt**2)**2)/m_N
671
672 print(f'magn. moment e_gauss: {round(m_e_gauss,5)}+-{round(s_m_e_gauss,5)}
        ↪ 1/m_N')
673 print(f'magn. moment e_lorentz: {round(m_e_lorentz,5)}+-{round(
        ↪ s_m_e_lorentz,5)} 1/m_N')
```

```
674 print(f'magn. moment e_voigt: {round(m_e_voigt,5)}+-{round(s_m_e_voigt,5)}
        ↪ 1/m_N\n')
675
676
677
678 ############ effective absorber thickness ##############
679 print('------------------------- effecitve absorber thickness
        ↪ -------------------------')
680
681 f_A      = 0.8 #debye-waller aus anleitung
682 d_A      = 25*10**(-6) #meter
683 beta     = 0.022 # anteil von 57^Fe in Probe
684 f        = 0.98 #%, iron content in absorber
685 s_f      = 0.02 #%
686
687
688 ## sigma_0 berechnen
689 lambdaa = 0.0861*10**(-9)#meter
690 I_e      = 3/2 #spin excited state
691 I_g      = 1/2 #sping ground state
692 alpha    = 8.58
693 s_alpha = 0.18
694
695 sigma_0      = (lambdaa**2/(2*np.pi)) * (2*I_e+1)/(2*I_g+1) * 1/(1+alpha)# m
        ↪ ^2
696 s_sigma_0    = sigma_0*s_alpha/(1+alpha)
697 print(f'sigma={round(sigma_0*10**(24),0)}+-{round(s_sigma_0*10**(24),0)}
        ↪ 10^-24 m^2')
698
699 rho = 7874 #kg/m^3
700 M   = 55.845*10**(-3) # kg/mol
701 s_M = 0.002*10**(-3) #kg/mol
702 N_A = 6.02214076 * 10**(23) #1/mol avogadro
703
704 n_A      = rho*(N_A/M)*f
705 s_n_A    = n_A*np.sqrt((s_f/f)**2 + (s_M/M)**2)
706 print(f'n_A={round(n_A*10**(-28),1)}+-{round(s_n_A*10**(-28),1)} *10^28 m
        ↪ ^-3')
707
708 T_A    = f_A*n_A*beta*sigma_0*d_A
709 s_T_A = T_A*np.sqrt((s_n_A/n_A)**2 + (s_sigma_0/sigma_0)**2)
710 print(f'T_A={round(T_A,1)}+-{round(s_T_A,1)}\n')
711
712 ## each line has its own effecitve absorber thickness
713
714
715 N_infty_gauss    = fitPara_gauss[-1]
716 N_infty_lorentz = fitPara_lorentz[-1]
717 N_infty_voigt    = fitPara_voigt[-1]
718
719 s_N_infty_gauss    = np.sqrt(fitCova_gauss[-1][-1])
720 s_N_infty_lorentz = np.sqrt(fitCova_lorentz[-1][-1])
721 s_N_infty_voigt    = np.sqrt(fitCova_voigt[-1][-1])
722
723 N_mu_gauss = []
724 s_N_mu_gauss = []
725 N_mu_lorentz = []
726 s_N_mu_lorentz = []
```

```
727 N_mu_voigt = []
728 s_N_mu_voigt = []
729 for i in range(0,6):
730     N_mu_gauss.append(multi_gauss(mu_gauss[i],*fitPara_gauss))
731     s_N_mu_gauss.append(multi_gauss(mu_gauss[i]-s_mu_gauss[i],*
        ↪ fitPara_gauss)- N_mu_gauss[i])
732     N_mu_lorentz.append(multi_lorentz(mu_lorentz[i],*fitPara_lorentz))
733     s_N_mu_lorentz.append(multi_lorentz(mu_lorentz[i]-s_mu_lorentz[i],*
        ↪ fitPara_lorentz) - N_mu_lorentz[i])
734     N_mu_voigt.append(multi_voigt(mu_voigt[i],*fitPara_voigt))
735     s_N_mu_voigt.append(multi_voigt(mu_voigt[i]-s_mu_voigt[i],*
        ↪ fitPara_voigt)- N_mu_voigt[i])
736
737
738 I_j_gauss = []
739 s_I_j_gauss = []
740 I_j_lorentz = []
741 s_I_j_lorentz = []
742 I_j_voigt = []
743 s_I_j_voigt = []
744 for i in range(0,6):
745     I_j_gauss.append(N_infty_gauss - N_mu_gauss[i])
746     s_I_j_gauss.append(np.sqrt((s_N_infty_gauss)**2 + (s_N_mu_gauss[i])**2)
        ↪ )
747     I_j_lorentz.append(N_infty_lorentz - N_mu_lorentz[i])
748     s_I_j_lorentz.append(np.sqrt((s_N_infty_lorentz)**2 + (s_N_mu_lorentz[i
        ↪ ])**2))
749     I_j_voigt.append(N_infty_voigt - N_mu_voigt[i])
750     s_I_j_voigt.append(np.sqrt((s_N_infty_voigt)**2 + (s_N_mu_voigt[i])**2)
        ↪ )
751
752
753 norm_gauss = sum(I_j_gauss)
754 norm_lorentz = sum(I_j_lorentz)
755 norm_voigt = sum(I_j_voigt)
756
757 liste = []
758 for i in range(0,6):
759     liste.append(s_I_j_gauss[i]**2)
760 s_norm_gauss=np.sqrt(sum(liste))
761
762 liste = []
763 for i in range(0,6):
764     liste.append(s_I_j_lorentz[i]**2)
765 s_norm_lorentz=np.sqrt(sum(liste))
766
767 liste = []
768 for i in range(0,6):
769     liste.append(s_I_j_voigt[i]**2)
770 s_norm_voigt=np.sqrt(sum(liste))
771
772
773 w_j_gauss = []
774 s_w_j_gauss = []
775 w_j_lorentz = []
776 s_w_j_lorentz = []
777 w_j_voigt = []
778 s_w_j_voigt = []
```

```
779 for i in range(0,6):
780     w_j_gauss.append(I_j_gauss[i]/norm_gauss)
781     s_w_j_gauss.append(w_j_gauss[i]*np.sqrt((s_I_j_gauss[i]/I_j_gauss[i])
        ↪ **2 + (2*s_norm_gauss/norm_gauss)**2))
782     w_j_lorentz.append(I_j_lorentz[i]/norm_lorentz)
783     s_w_j_lorentz.append(w_j_lorentz[i]*np.sqrt((s_I_j_lorentz[i]/
        ↪ I_j_lorentz[i])**2 + (2*s_norm_lorentz/norm_lorentz)**2))
784     w_j_voigt.append(I_j_voigt[i]/norm_voigt)
785     s_w_j_voigt.append(w_j_voigt[i]*np.sqrt((s_I_j_voigt[i]/I_j_voigt[i])
        ↪ **2 + (2*s_norm_voigt/norm_voigt)**2))
786
787
788
789
790 T_A_j_gauss = []
791 s_T_A_j_gauss = []
792 T_A_j_lorentz = []
793 s_T_A_j_lorentz = []
794 T_A_j_voigt = []
795 s_T_A_j_voigt = []
796 for i in range(0,6):
797     T_A_j_gauss.append(T_A*w_j_gauss[i])
798     s_T_A_j_gauss.append(T_A_j_gauss[i]*np.sqrt((s_T_A/T_A)**2 + (2*
        ↪ s_w_j_gauss[i]/w_j_gauss[i])**2))
799     T_A_j_lorentz.append(T_A*w_j_lorentz[i])
800     s_T_A_j_lorentz.append(T_A_j_lorentz[i]*np.sqrt((s_T_A/T_A)**2 + (2*
        ↪ s_w_j_lorentz[i]/w_j_lorentz[i])**2))
801     T_A_j_voigt.append(T_A*w_j_voigt[i])
802     s_T_A_j_voigt.append(T_A_j_voigt[i]*np.sqrt((s_T_A/T_A)**2 + (2*
        ↪ s_w_j_voigt[i]/w_j_voigt[i])**2))
803
804 print('---------- gewichte und T_A_j ----------')
805
806 print('\n----------Gauss----------')
807 for i in range(0,6):
808     print(f'w_{i+1}: {round(w_j_gauss[i],3)}+-{round(s_w_j_gauss[i],3)}')
809 for i in range(0,6):
810     print(f'T_A_{i+1}: {round(T_A_j_gauss[i],2)}+-{round(s_T_A_j_gauss[i
        ↪ ],2)}')
811
812 print('\n----------Lorentz----------')
813 for i in range(0,6):
814     print(f'w_{i+1}: {round(w_j_lorentz[i],3)}+-{round(s_w_j_lorentz[i],3)}
        ↪ ')
815 for i in range(0,6):
816     print(f'T_A_{i+1}: {round(T_A_j_lorentz[i],2)}+-{round(s_T_A_j_lorentz[
        ↪ i],2)}')
817
818 print('\n----------Voigt----------')
819 for i in range(0,6):
820     print(f'w_{i+1}: {round(w_j_voigt[i],3)}+-{round(s_w_j_voigt[i],3)}')
821 for i in range(0,6):
822     print(f'T_A_{i+1}: {round(T_A_j_voigt[i],2)}+-{round(s_T_A_j_voigt[i
        ↪ ],2)}')
823
824
825 T_A_alle = []
826 s_T_A_alle = []
```

```
827 for i in range(0,6):
828     T_A_alle.append(T_A_j_gauss[i])
829     s_T_A_alle.append(s_T_A_j_gauss[i])
830 for i in range(0,6):
831     T_A_alle.append(T_A_j_lorentz[i])
832     s_T_A_alle.append(s_T_A_j_lorentz[i])
833 for i in range(0,6):
834     T_A_alle.append(T_A_j_voigt[i])
835     s_T_A_alle.append(s_T_A_j_voigt[i])
836
837 a = []
838 b = []
839 for i in range(0,len(T_A_alle)):
840     a.append(T_A_alle[i]/s_T_A_alle[i]**2)
841     b.append(1/s_T_A_alle[i]**2)
842
843 T_A_alle_wert = sum(a)/sum(b)
844 s_T_A_alle_wert = np.sqrt(1/sum(b))
845
846
847 T_A_j_lorentz = [9,9,9,9,9,9]
848 s_T_A_j_lorentz = [0.3,0.3,0.3,0.3,0.3,0.3]
849
850 print('\n---------- Debye_Waller factor f_Q_j ----------')
851
852 f_Q_gauss = []
853 f_Q_lorentz = []
854 f_Q_voigt = []
855
856 s_f_Q_gauss = []
857 s_f_Q_lorentz = []
858 s_f_Q_voigt = []
859
860
861 for i in range(0,6):
862     f_Q_gauss.append(np.real((N_infty_gauss-N_mu_gauss[i])/(N_infty_gauss
        ↪ *(1-np.exp(-T_A_j_gauss[i]/2))*jv(0,1j* T_A_j_gauss[i]/2))))
863     f_Q_lorentz.append(np.real((N_infty_lorentz-N_mu_lorentz[i])/(
        ↪ N_infty_lorentz*(1-np.exp(-T_A_j_lorentz[i]/2))*jv(0,1j*
        ↪ T_A_j_lorentz[i]/2))))
864     f_Q_voigt.append(np.real((N_infty_voigt-N_mu_voigt[i])/(N_infty_voigt
        ↪ *(1-np.exp(-T_A_j_voigt[i]/2))*jv(0,1j* T_A_j_voigt[i]/2))))
865
866     s_f_Q_gauss.append(np.sqrt(np.real((N_mu_gauss[i]*s_N_infty_gauss/(
        ↪ N_infty_gauss**2*(1-np.exp(-T_A_j_gauss[i]/2)*jv(0,1j*T_A_j_gauss
        ↪ [i]/2))))**2+(-s_N_mu_gauss[i]/(N_infty_gauss*(1-np.exp(-
        ↪ T_A_j_gauss[i]/2)*jv(0,1j*T_A_j_gauss[i]/2))))**2+(-np.exp(-
        ↪ T_A_j_gauss[i]/2)*(jv(0,1j* T_A_j_gauss[i]/2)+1j*jv(1,1j*
        ↪ T_A_j_gauss[i]/2))*s_T_A_j_gauss[i]/(2*(np.exp(-T_A_j_gauss[i]/2)
        ↪ -jv(0,1j*T_A_j_gauss[i]/2))**2))**2)))
867     s_f_Q_lorentz.append(np.sqrt(np.real((N_mu_lorentz[i]*s_N_infty_lorentz
        ↪ /(N_infty_lorentz**2*(1-np.exp(-T_A_j_lorentz[i]/2)*jv(0,1j*
        ↪ T_A_j_lorentz[i]/2))))**2+(-s_N_mu_lorentz[i]/(N_infty_lorentz
        ↪ *(1-np.exp(-T_A_j_lorentz[i]/2)*jv(0,1j*T_A_j_lorentz[i]/2))))
        ↪ **2+(-np.exp(-T_A_j_lorentz[i]/2)*(jv(0,1j* T_A_j_lorentz[i]/2)+1
        ↪ j*jv(1,1j*T_A_j_lorentz[i]/2))*s_T_A_j_lorentz[i]/(2*(np.exp(-
        ↪ T_A_j_lorentz[i]/2)-jv(0,1j*T_A_j_lorentz[i]/2))**2))**2)))
```

```
868     s_f_Q_voigt.append(np.sqrt(np.real((N_mu_voigt[i]*s_N_infty_voigt/(
            ↪ N_infty_voigt**2*(1-np.exp(-T_A_j_voigt[i]/2)*jv(0,1j*T_A_j_voigt
            ↪ [i]/2))))**2+(-s_N_mu_voigt[i]/(N_infty_voigt*(1-np.exp(-
            ↪ T_A_j_voigt[i]/2)*jv(0,1j*T_A_j_voigt[i]/2))))**2+(-np.exp(-
            ↪ T_A_j_voigt[i]/2)*(jv(0,1j* T_A_j_voigt[i]/2)+1j*jv(1,1j*
            ↪ T_A_j_voigt[i]/2))*s_T_A_j_voigt[i]/(2*(np.exp(-T_A_j_voigt[i]/2)
            ↪ -jv(0,1j*T_A_j_voigt[i]/2))**2))**2)))
869
870
871 print('\n----------Gauss----------')
872 for i in range(0,6):
873     print(f'f_Q_{i+1}: {round(f_Q_gauss[i],2)}+-{round(s_f_Q_gauss[i],2)}')
874
875 print('\n----------Lorentz----------')
876 for i in range(0,6):
877     print(f'f_Q_{i+1}: {round(f_Q_lorentz[i]/w_j_lorentz[i],5)}+-{round(
            ↪ s_f_Q_lorentz[i],2)}')
878
879 print('\n----------Voigt----------')
880 for i in range(0,6):
881     print(f'f_Q_{i+1}: {round(f_Q_voigt[i],2)}+-{round(s_f_Q_voigt[i],2)}')
882
883
884
885 ########## line width ############
886 print('\n---------- line width gamma ----------')
887
888 Gamma_gauss = []  #mm/s
889 s_Gamma_gauss = []
890 Gamma_lorentz = []
891 s_Gamma_lorentz = []
892 Gamma_voigt = []
893 s_Gamma_voigt = []
894 for i in range(0,6):
895     Gamma_gauss.append(2*np.sqrt(2*np.log(2))*fitPara_gauss[i+6])
896     s_Gamma_gauss.append(2*np.sqrt(2*np.log(2))*np.sqrt(fitCova_gauss[i+6][
            ↪ i+6]))
897     Gamma_lorentz.append(2*fitPara_lorentz[i+6])
898     s_Gamma_lorentz.append(2*np.sqrt(fitCova_lorentz[i+6][i+6]))
899     Gamma_voigt.append(2*fitPara_voigt[i+12])
900     s_Gamma_voigt.append(2*np.sqrt(fitCova_voigt[i+12][i+12]))
901
902 print('\n----------Gauss----------')
903 for i in range(0,6):
904     print(f'Gamma_{i+1}: {round(Gamma_gauss[i],2)}+-{round(s_Gamma_gauss[i
            ↪ ],2)} mm/s')
905
906 print('\n----------Lorentz----------')
907 for i in range(0,6):
908     print(f'Gamma_{i+1}: {round(Gamma_lorentz[i],2)}+-{round(
            ↪ s_Gamma_lorentz[i],2)} mm/s')
909
910 print('\n----------Voigt----------')
911 for i in range(0,6):
912     print(f'Gamma_{i+1}: {round(Gamma_voigt[i],2)}+-{round(s_Gamma_voigt[i
            ↪ ],2)} mm/s')
913
914
```

```
915
916 print('\n----------Gauss----------')
917 for i in range(0,6):
918     print(f'Gamma_{i+1}: {round(Gamma_gauss[i]*(reee*10**(9)),2)}+-{round(
        ↪ s_Gamma_gauss[i]*(reee*10**(9)),2)} neV')
919
920 print('\n----------Lorentz----------')
921 for i in range(0,6):
922     print(f'Gamma_{i+1}: {round(Gamma_lorentz[i]*(reee*10**(9)),2)}+-{round
        ↪ (s_Gamma_lorentz[i]*(reee*10**(9)),2)} neV')
923
924 print('\n----------Voigt----------')
925 for i in range(0,6):
926     print(f'Gamma_{i+1}: {round(Gamma_voigt[i]*(reee*10**(9)),2)}+-{round(
        ↪ s_Gamma_voigt[i]*(reee*10**(9)),2)} neV')
927
928
929 ########## lifetime ############
930 print('\n---------- life time tau
    ↪ ----------')
931 hquer = 6.582119569 *10**(-16) #eVs
932
933 tau_g = [] #ns
934 s_tau_g = []
935 tau_l = []
936 s_tau_l = []
937 tau_v = []
938 s_tau_v = []
939 for i in range(0,6):
940     tau_g.append(hquer/(Gamma_gauss[i]*reee)*10**9)
941     s_tau_g.append(tau_g[i]*s_Gamma_gauss[i]/Gamma_gauss[i])
942     tau_l.append(hquer/(Gamma_lorentz[i]*reee)*10**9)
943     s_tau_l.append(tau_l[i]*s_Gamma_lorentz[i]/Gamma_lorentz[i])
944     tau_v.append(hquer/(Gamma_voigt[i]*reee)*10**9)
945     s_tau_v.append(tau_v[i]*s_Gamma_voigt[i]/Gamma_voigt[i])
946
947 print('\n----------Gauss----------')
948 for i in range(0,6):
949     print(f'tau_{i+1}: {round(tau_g[i],2)}+-{round(s_tau_g[i],2)} ns')
950 print('\n----------Lorentz----------')
951 for i in range(0,6):
952     print(f'tau_{i+1}: {round(tau_l[i],2)}+-{round(s_tau_l[i],2)} ns')
953 print('\n----------Voigt----------')
954 for i in range(0,6):
955     print(f'tau_{i+1}: {round(tau_v[i],2)}+-{round(s_tau_v[i],2)} ns')
956
957
958 ############## lifetime korrektur #################
959 print('\n---------- life time tau korrigiert
    ↪ ----------')
960 rel_width = 2
961
962 print('\n----------Gauss----------')
963 for i in range(0,6):
964     print(f'tau_{i+1}: {round(tau_g[i]*rel_width,2)}+-{round(s_tau_g[i]*
        ↪ rel_width,2)} ns')
965 print('\n----------Lorentz----------')
966 for i in range(0,6):
```

```
967     print(f'tau_{i+1}: {round(tau_l[i]*rel_width,2)}+-{round(s_tau_l[i]*
            ↪ rel_width,2)} ns')
968 print('\n----------Voigt----------')
969 for i in range(0,6):
970     print(f'tau_{i+1}: {round(tau_v[i]*rel_width,2)}+-{round(s_tau_v[i]*
            ↪ rel_width,2)} ns')
```

H. Laboratory Journal

~~17.8.2020~~

Mößbauer

~~Signal analysis~~

Day 1

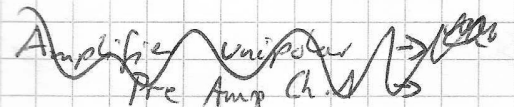
17.8.2020

Signal analysis

~~Preamplifier~~ → ~~file PREAMP.CSV~~

Analysis of the signal after ~~stop~~ ~~com~~ electronic components
signals averaged over 1024 signals
Used oscilloscope: ROHDE & SCHWARZ HMO 1022
Ludshidis removed from Co-source
Voltage at power supply: 847 V

Preamplifier → PREAMP.CSV
~~Amplifier~~
Amplifier → signal AMP_UNI2.CSV
→ signal of preamp (used for trigger)
AMP_UNI~~1~~.CSV



For this the amplification coarse gain ~~100~~ 50 fine gain ~~1~~ 1 is chosen. The shaping time is set to 2 µs

~~Ch.A.~~ PreAmp → Ch. 1 → AMP_UNIpo1.CSV
Amp Unipolar → Ch. 2 → AMP_UNI2.CSV

~~#~~ Now the bipolar output
Pre Amp Ch. 1 → AMP_BI1.CSV
Amp Bipolar Ch. 2 → ~~#~~AMP_BI2.CSV

Timing SCA (pos. output) → SCA~~#~~ 04.CSV
Linear gate DC inhibit → ~~SCA~~ LIN01.CSV
Linear gate enabled, no input → LIN02.CSV

31.8.20

Measurement delay amplifier

delay time: 2,75 µs

Signal not delayed (ch. 1) → DEL01.csv

Signal with delay (ch. 2) → DEL02.csv

Measurement signal path times

Signal: Delay: 4,75 µs
(after gate)
→ PATH01.csv

Logical signal
(after SCA)
→ PATH02.csv

Signals

voltage: 845 V

Preamplifier → file: 001.csv

Amplifier (unipol) → file: 002.csv
- Gain: 100
- Shaping time: 1 µs

Timing SCA → file: 004.csv ∈ used for trigger
~~• Trigger signal SCA: file: 02.csv~~
+ Amplifier → file: 003.csv ∈ for time reference

Delay t = 3,25 µs → file: 006.csv
+ Amplifier → file: 005.csv ∈ for time reference used for trigger

Linear Gate opening time (no signal input) → file: ~~009~~ 010.csv
+ SCA → file: 009.csv ∈ used for trigger

Linear Gate outputsignal → file: 012.csv
+ SCA → file: 011.csv ∈ used for trigger

Linear Gate outputsignal → file: 014.csv
+ Delay → file: 013.csv ∈ trigger

Calibration of the channels

→ pre liminary evaluation

→ lin. calibration fit $f(x) = ax + b$

$$f(\text{Energy}) = [\text{Channel}] = (18.75 \pm 0.41)\frac{1}{\text{keV}} \cdot X + (3.3 \pm 1.2)$$

=> 14.4keV peak is @ channel 273 ± 11.

Calibration of the channels

files:

Ag:	Ag_2_spec. TKA Ag_spec. TKA	measurement time for each data: 6 min
Ba:	Ba_spec. TKA	SCA-Window opened
Cu:	Cu_spec. TKA	
Mo:	Mo_spec. TKA	
Rb:	Rb_spec. TKA	
Tb:	Tb_spec. TKA	
Background:	Background. TKA	

Spe

Night-measurement: Spectrum of the 14,4keV source with open SCA (time: 52161 s)

-> file: spectrum_night-meas. TKA

18.8.2020 Day 2

background measurement for the 14,4keV ~~spectrum~~ source spectrum (time: 6420 s)

-> file: background-long. TKA

Set window of SCA according to preevaluation

→ cut spectrum such that the peak is fully in the window but as few surounding as possible so as to get less background

→ file: alu-FULL.TKA

Measurement of compton background

No absorber, 1mm Alu → $t = 2.89875 \cdot 10^6$ ms, $N = 75972$

Stainless steel absorber, $t = 5$ min for all measurements

Alu thickness [mm]	File name Stainless steel	natural iron
1 mm	alu-st-1.TKA	alu-ei-1.TKA
1.5	2	2
2	3	3
2.5	4	4
3	5	5
3.5 (2+1.5)	6	6
4	7	7
4.5 (2+2.5)	8	8
5 (2+3)	9	9
5.5 (3+2.5)	10	10
6 (4+2)	11	11
6.5 (4+2.5)	12	12
7 (4+3)	13	13
7.5 (4+2+1.5)	14	14
8 (4+2.5+1.5)	15	15
8.5 (4+3+1.5)	16	16
9 (4+3+2)	17	17
9.5 (4+3+2.5)	18	18
10 (4+3+2.5+1)	19	19
0	20	20
0,2 (p1)	21	21
0,4 (p1+p2)	22	22
0,6 (p1+p2+p3)	23	23
0,8 (p1+p2+p3+p4)	24	24
1,2 (1+p1)	25	25
1,4 (1+p1+p2)	26	26
1,6 (1+p1+p2+p3)	27	27
1,8 (1+p1+p2+p3+p4)		

Alu Thickness Measurements

1mm: 1.005mm, 1.00, 1.005, 1.005, 1.00, 1.00, 1.005, 1.00, 1.00, 1.07, 1.00, 1.005, 1.005

1.5mm: 1.46, 1.47, 1.46, 1.46, 1.46, 1.465, 1.455, 1.455, 1.46, 1.46

2.0mm: 1.95, 1.95, 1.955, 1.955, 1.955, 1.950, 1.945, 1.975, 1.955

2.5mm: 2.51, 2.51, 2.51, 2.51, 2.51, 2.505, 2.51, 2.51, 2.51, 2.51, 2.51

3.0mm: 3.01, 3.01, 3.01, 3.015, 3.015, 3.015, 3.01, 3.01, 3.01, 3.02, 3.01, 3.01, 3.005

4.00mm: 3.985, 3.990, 3.99, 3.985, 3.98, 3.98, 4.000, 3.985, 3.980

platte 1: 0,28mm, 0,215mm, 0,210mm, 0,210mm, ~~0,305mm~~
0,21mm, 0,21mm, 0,26mm, 0,225mm

platte 2: 0,245mm, 0,215mm, 0,22mm, 0,21mm
0,305mm, 0,205mm, 0,21mm, 0,22mm

platte 3: 0,21mm, 0,22mm, 0,21mm, 0,225mm
0,215mm, 0,225mm, 0,21mm, 0,22mm

platte 4: 0,215mm, 0,22mm, 0,245mm, 0,22mm
0,23mm, 0,205mm, 0,22mm, 0,205mm

acrylic glass abbreviation

Thickness of the acrylic glass is measured with „milimeter schraube"

$d = 1,96\,mm$; $1,99\,mm$; $1,975\,mm$; $1,975\,mm$

$\Rightarrow \bar{d} = 1,975\,mm$

measurement with no plexiglass → file: no_plexi.TKA

with plexiglass → file: plexi.TKA

27.8.2020

testing the velocity shown by the PC.
The distance which is paised by the wagon is measured with a ruler. The distance set is 720mm. The time is measured with a smartphone.

v_{PC} [mm/s]	start [cm]	stop [cm]	t [s]	s_x
8	4	4	10,75	±0,3 mm
	5	4	11,25	
7	6	4	14,13	
	7	3	14,25	
6	6	5	18,39	
	7	3	16,86	
5	6	5	22,26	±0,2 mm
	7	4	22,28	
4	6	4	25,25	
	7	3	25,09	
3	6	4	33,54	±0,1 mm
	7	4	36,88	*
2	7	4	55,26	
	7	4	55,56	
1	7	4	1:50,72	
	7	4	1:51,04	

⇒ file: velo1.txt

8	7		
7	4		
	6	2	11,52
~~6~~			
8	6	3	11,24
	5	3	10,05
7	6	4	14,26
	7	4	16,06
~~6~~			

⇒ file: velo2.txt

31.8.20