Mößbauer effect

Advanced Practical II

Tim Butcher - Florian Tönnies

Faculty for Mathematics and Physics
Albert-Ludwigs-Universität Freiburg

7th March 2016

Abstract

In the experiment the line of the 14.4 keV transition of ^{57}Fe was recorded with a one line absorber (stainless steel) and a six line absorber (natural iron) using Mößbauer spectroscopy. The absorber was exposed to the gamma radiation emitted by the decay of ^{57}Co to an excited state of ^{57}Fe, while moving at different velocities. The resulting Mößbauer spectrum was measured with a scintillation counter. The lifetime of the excited state and isomeric shift were extracted from the width and shift (from $v = 0$) of the peak. The established values from the one line absorber are:

	Value	Literature
Isometric shift: E_{iso}	(8.9 ± 0.2) neV	
Excited state lifetime: τ	(150 ± 10) ns	141 ns
Excited state half life: $T_{1/2}$	(104 ± 8) ns	98 ns
Excited state lifetime (from fit): τ_{fit}	(83 ± 5) ns	141 ns
Effective absorber thickness: T_A	6.2 ± 0.6	
Debye-Waller factor of the source: f_S	0.56 ± 0.02	

With the six line absorber the hyperfine splitting and magnetic field at the position of the nucleus were additionally determined:

	Value	literature
Isomeric shift: E_{iso}	(5.0 ± 0.5)neV	5.3 neV
	Value / μ_N	
nuclear magnetic moment $\mu_{e,ab}$	-0.153 $\pm$ 0.004	
nuclear magnetic moment $\mu_{e,ac}$	-0.159 $\pm$ 0.002	(-0.1549 $\pm$ 0.0002) μ_N
nuclear magnetic moment $\mu_{e,bc}$	-0.159 $\pm$ 0.003	
	Value / T	
magnetic field B_a	33.1 ± 1.9	
magnetic field B_b	32.7 ± 0.6	≈ 33 T
magnetic field B_c	32.6 ± 2.7	

The majority of the experimental values are compatible with the corresponding literature values.

Table of Contents

1 Introduction

1.1 Objective

The line pertaining to the 14.4 keV transition of ^{57}Fe is to be recorded with a one line absorber (stainless steel) and a six line absorber (natural iron) using the Mößbauer effect. The isomeric shift, effective absorber thickness, Debye-Waller factor of the source, lifetime of the excited state, magnetic moment and magnetic field at the position of the nucleus are to be extracted from the absorption spectra.

1.2 Theoretical Background

1.2.1 Gamma radiation

The decay of an excited nuclear state (as a consequence of α- or β decay) into its ground state leads to the emission of a photon of very high frequency (γ-ray). In certain circumstances no photon is emitted and the energy is directly transferred to an electron in the atomic shell. This electron leaves the atom with an energy reduced by its bonding energy. This process is called *inner conversion*. For heavier elements the energy difference gives rise to the emission of X-radiation. The *Auger-effect* occurs for lighter elements. Here the remaining energy is transferred to another electron (*Auger-electron*), which is also emitted.

1.2.2 Interaction of Gamma radiation with matter

1. Photoelectric effect
   The inbound photon knocks an electron out of the atom shell. This occurs at low photon energies of up to 100 keV.

2. Compton effect
   The photon is scattered by a weakly bound electron, transferring a part of its energy. This process mainly happens at energies of 100 keV to several MeV (and is therefore the main disturbing effect during the course of the experiment see section 3.2).

3. Pair production
   The minimum energy for pair production is 1.022 MeV. This is twice the rest energy of an electron. A positron-electron pair is created, which annihilates to a pair of photons.

1.2.3 Nuclear resonance absorption

A photon that is emitted from the decay of an excited state of the nucleus can lead to the excitation of a nucleus of the same kind into the same excited state. This phenomenon is known as *nuclear resonance absorption*. Under normal circumstances the observation of this process is impossible. The reason for this is the recoil energy that is transferred to the nucleus. The momentum of the photon is:

$$p = \hbar k = \frac{\hbar\omega}{c} = \frac{E_\gamma}{c} \tag{1}$$

So the recoil energy is:

$$E_r = \frac{p^2}{2m} = \frac{E_\gamma^2}{2mc^2} \tag{2}$$

The photon loses twice this energy as energy is also transferred during the absorption process. So resonance absorption is not observable for a free atom and a very thin spectral line.
The width of a spectral line (Γ) is governed by the *time-energy uncertainty*. For an excited state with a lifetime of τ the line width is:

$$\Gamma = \frac{\hbar}{\tau} \tag{3}$$

The thermal motion of the nuclei further complicates the situation. The thermal motion leads to a momentum component in the direction of emission p_t. The shift in energy is (with the recoil momentum p and excitation energy E_0):

$$\Delta E = \frac{p^2}{2m} - pv_t = \frac{E_\gamma^2}{2mc^2} - E_\gamma \frac{v_t}{c} \tag{4}$$

The first term is the recoil energy. The second term is the energy shift due to the *Doppler-effect*. Because of the thermal velocity distribution the Doppler-shift does not have a fixed value but is of a similar magnitude as the recoil energy. From this it follows, that if the velocity are sufficiently broadly distributed ΔE can become exactly zero. If this is the case resonance absorption is possible.

1.2.4 The Mößbauer effect

The above paragraph only concerned free atoms. The situation is vastly different when the atoms are in a crystal lattice. The *Mössbauer effect* is the name of the process of *recoilless resonance absorption* of atoms in a crystal. Recoilless absorption simply means momentum transfer without energy transfer (as in the mechanical example of the reflection of a ball by a wall). An atom in a lattice has three degrees of freedom in which harmonic oscillations can take place. The energy of the oscillation is quantised in factors of $\hbar\omega$. The oscillator quantum $\hbar\omega$ is a quasiparticle by the name of *phonon*. If the recoil energy of the gamma emission is larger than $\hbar\omega$ it is transformed into oscillation energy and eventually thermal energy. If on the hand $E_r < \hbar\omega$ the situation can be analysed with two theories of solids:

I. Einstein model

The Einstein model places every individual atom in a three dimensional harmonic oscillator ($3N$ oscillators)and assumes that all of the atoms oscillate with the same frequency ω_E. Thus the vibrational spectrum $Z(\omega)$ becomes (see Fig. 1.1):

$$Z_E = 3N \cdot \delta(\omega - \omega_E) \tag{5}$$

The direct consequence of this is that the lattice can only change energy in integer values of $\hbar\omega$. So either no energy is transferred or at least $\hbar\omega$. The case of no energy transfer corresponds to *recoilless emission*. The Einstein model predicts a non shifted line of natural width accompanied by a line shifted by $\hbar\omega$. The shifted line is rather wide because of the high natural line width of a Phonon-line and the phonon spectrum is more complicated than the Einstein model suggests.

II. Debye model

Unlike the Einstein model the Debye model allows different frequencies. Three branches following a linear dispersion relation are introduced for the vibrational spectrum:

$$\omega = ck \tag{6}$$

With the speed of sound in the solid (c) and the wave vector k (for high wavelengths $\Leftrightarrow$ small wave vectors). k_D is an indicator of the reciprocal particle spacing. Instead of integrating over the first *Brillouin zone* one integrates over a sphere with radius k_D (*Debye wave vector*). The value of k_D is chosen that the integral equals the number of ions in the crystal (N). From this it follows that k_D satisfies this relation:

$$n = \frac{k_D^3}{6\pi^2} \tag{7}$$

n being the particle density $n = \frac{N}{V}$. The Debye wave vector is used to define the *Debye frequency* (ω_D) and the *Debye temperature* (Θ_D):

$$\omega_D = k_D c \tag{8}$$

$$k\Theta_D = \hbar\omega = \hbar c k_D \tag{9}$$

The maximum phonon frequency is ω_D and Θ_D is the temperature at which all modes begin to be excited. At temperatures lower than Θ_D modes are "frozen out". Both quantities can also be interpreted as the "stiffness" of the crystal. The vibrational spectrum becomes (see Fig. 1.1):

$$Z_D(\omega) = \begin{cases} \frac{9N}{\omega_D^3} \cdot \omega^2 & \text{for} \quad \omega \leq \omega_D \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The Debye model holds at low frequencies where its predictions can be experimentally reproduced.

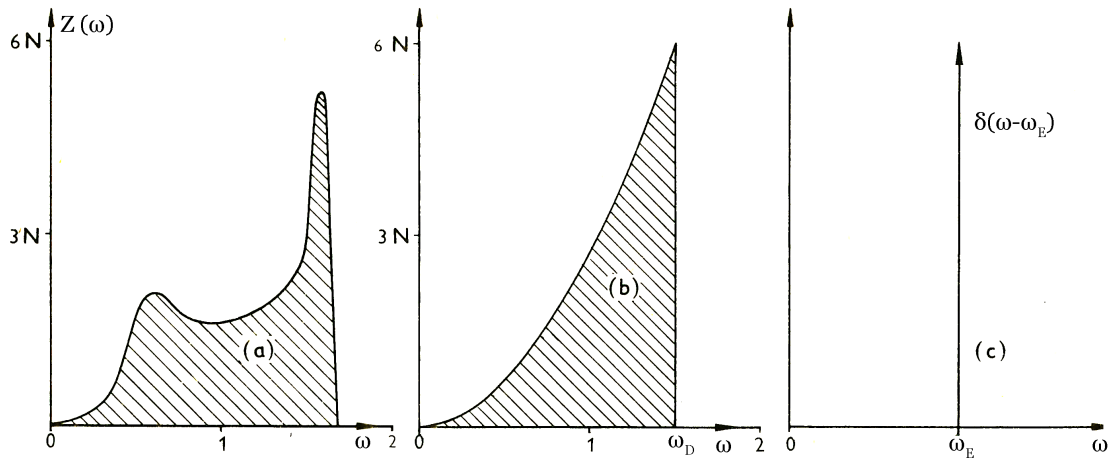


Figure 1.1: Frequency spectrum of lattice oscillations; left: real spectrum; middle: Debye model; right: Einstein model (from [7])

1.2.5 Experimental observation of the recoilless resonance absorption for the 14.4 keV transition of ^{57}Fe

The radioactive source used in the experiment is ^{57}Co and its decay can be seen in Fig. (**1.2**). 99.8% of it decays to ^{57}Fe* (by electron capture) with a half life of 270 days:

$$^{57}_{27}\mathrm{Co} + \mathrm{e}^- \longrightarrow {}^{57}_{26}\mathrm{Fe}^* + \nu_e \tag{11}$$

The excited state of ^{57}Fe has a 14.4 keV transition to the ground state (with a half life of 98 ns).

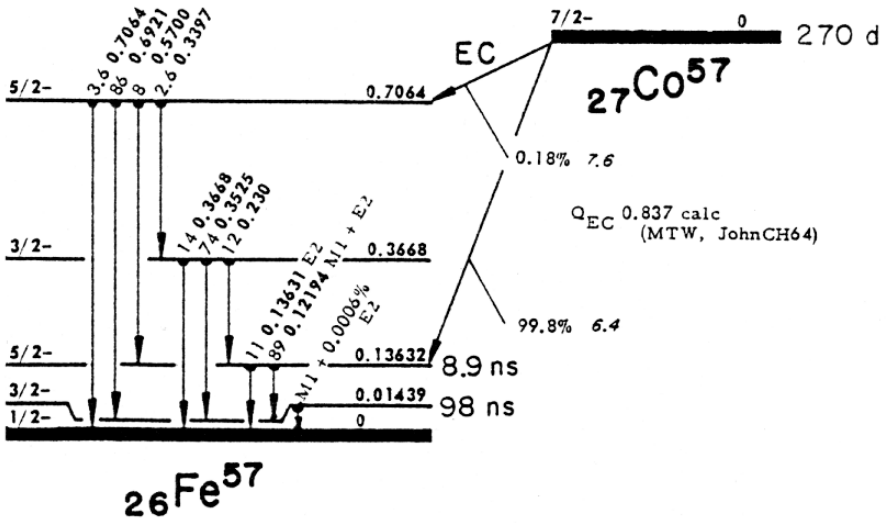


Figure 1.2: Energy scheme for the decay of ^{57}Cobalt

The lifetime of the excited state is $\tau = 140\,\mathrm{ns}$ which means a line width of $\Gamma = 4.7\,\mathrm{neV}$ ([9]). The relative line width is $\frac{\Gamma}{E_\gamma} = 3 \cdot 10^{-13}$, which is very thin. The energy resolution of conventional scintillation counters is not sufficiently high and the line can not be directly measured. The problem is solved by the Mößbauer effect. Moving the absorber leads to a Doppler-shift of the photons energy ($\Delta E = E_0(\frac{v}{c})$). Measuring the transmitted photons with a scintillation counter for varying velocities leads to the profile of the γ-line.

1.2.6 Debye-Waller factor

The Debye-Waller factor f is the fraction of recoilless absorption and follows from the Debye theory. The mean occupation number of phonons is required for the calculation. The Debye-Waller factor is given by:

$$f = exp\left[-\frac{3E_R}{2k\Theta_D}\left(1 + \frac{4T^2}{\Theta_D^2}\int_0^{\Theta_D/T}\frac{xdx}{e^x - 1}\right)\right] \tag{12}$$

$$\overset{T \ll \Theta_D}{\approx} exp\left[-\frac{E_r}{k\Theta_D}\left(\frac{3}{2} + \frac{\pi^2 T^2}{\Theta_D^2}\right)\right] \tag{13}$$

With the absolute temperature T, Boltzmann-constant k, Θ_D Debye temperature and the recoil energy E_R. The value of f in respect to different temperatures can be seen in Fig. **1.3**. The examined

14.4 keV transition of ^{53}Fe has the speciality that $f = 0.91$ at room temperature. Thus cooling the source and the absorber (as was done in the original Mößbauer experiment) is avoided.

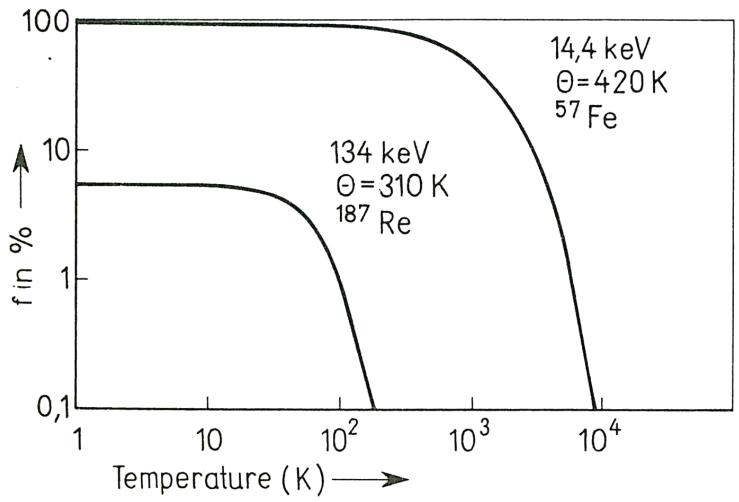


Figure 1.3: Debye-Waller factor for two typical transitions (from [9])

1.2.7 Isomeric shift (chemical shift)

The electron shell of the examined atom is (in good approximation) spherically symmetric and causes a slight change of the Coulomb energy of the nucleus, which alters the energy of the nuclear states. As excited and ground state have a different atomic radius and the electron density at the position of the nucleus differs, the energy of the transition is changed. This is called *isomeric shift* and its value is of the same magnitude as the natural line width. Isomeric shift only occurs when the electron densities in source and absorber are different. This is the reason why the line of ^{57}Fe is not symmetric to $v = 0$ (see section 3.4).

1.2.8 Hyperfine splitting

A magnetic field at the position of the nucleus can remove the degeneracy of the nuclear spins.

The coupling of magnetic field (B) and the magnetic moment of the nucleus (μ_I) makes itself noticeable in an interaction energy of:

$$E_{HF} = \frac{\mu_I m_I B}{I} \tag{14}$$

I being the nuclear spin and m_I its projection onto the axes of the magnetic field. The number of values for m_I is given by the *multiplicity* $2I + 1$ and they are all integers.

The hyperfine splitting of ^{57}Fe (between the $I_G = \frac{1}{2}$ and $I_E = \frac{3}{2}$ state) can be seen in Fig. **1.4** (in the experiment this occurs for the natural iron probe). Because of the selection rule only six transitions are allowed. The energy of one of the transition is:

$$E_\gamma = E_0 + E_{iso} - \left(\frac{\mu_E m_e}{I_e} - \frac{\mu_g m_g}{I_g} \right) B \tag{15}$$

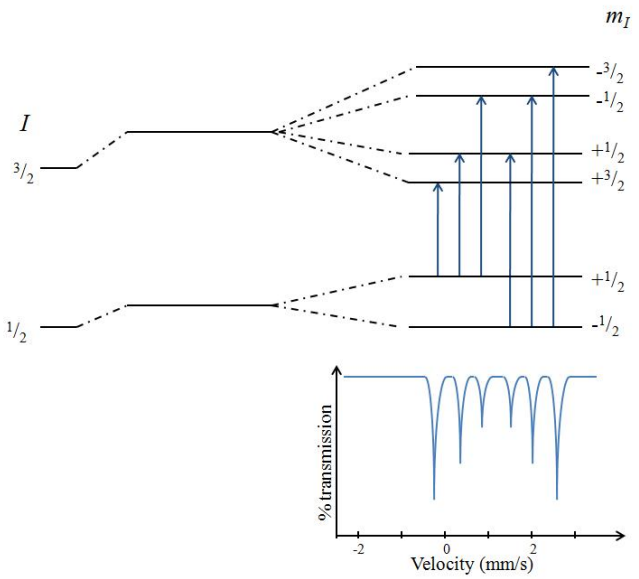


Figure 1.4: Hyperfine splitting for ^{57}Fe and resulting Mößbauer spectrum [1]

1.2.9 Scintillation counter

Scintillation counters are special measuring instruments designed to convert high-energy photons into radiation of lower energy but higher intensity. The intensity of the scintillator radiation is proportional to the energy of the detected radiation. The lower energy photons can permeate the medium and are amplified and detected with a *photomultiplier*. There are different scintillator materials. In this experiment an inorganic sodium-chloride (NaI) scintillator is used due to its superior energy resolution. It works as follows:
In the crystal electrons are raised from the valence- into the conduction band (band gap 6-8 eV [3]) by inbound γ-photons. A photon of characteristic energy is set free when the electrons relax into the ground state. However these will be instantaneously reabsorbed. This does not happen if a difference in energy is present. Doping with Thallium slightly deforms the conduction band, resulting in a reduction of the band gap to 3 eV (see Fig. 1.5). Excitons (bound electron-hole pairs) or free charge carriers can be captured at these *activator centres*. They diffuse through the crystal until they recombine at an activator centre and emit a lower energy photon. These can leave the crystal and reach the photomultiplier.

[1] https://en.wikipedia.org/wiki/Mössbauer_spectroscopy

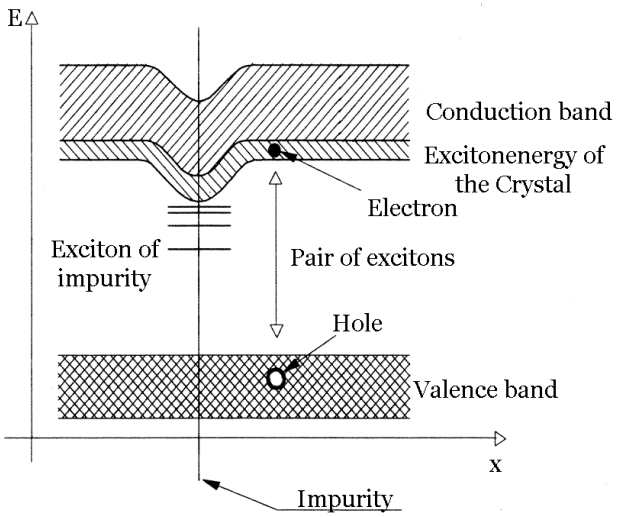


Figure 1.5: Energy levels of an inorganic scintillator (from [3])

A photomultiplier is required for the detection of the radiation set free by the scintillator. Upon arrival of photons at the photocathode electrons are set free, which are then accelerated towards the dynodes by an applied voltage. The electrons hit the dynode causing a cascade of electrons which are in turn accelerated towards the next dynode (see Fig. 1.6). The potential of the dynodes is gradually increased with a voltage divider. The gain in energy from the acceleration is converted into the release of further secondary electrons. The photomultiplier in the experiment includes a pre-amplifier which produces a voltage signal proportional to the input current. This is done with a reference capacitor, meaning that the voltage signal will resemble that of a discharging capacitor. This is how the low number of inbound photons can be transformed into a measurable electrical signal.

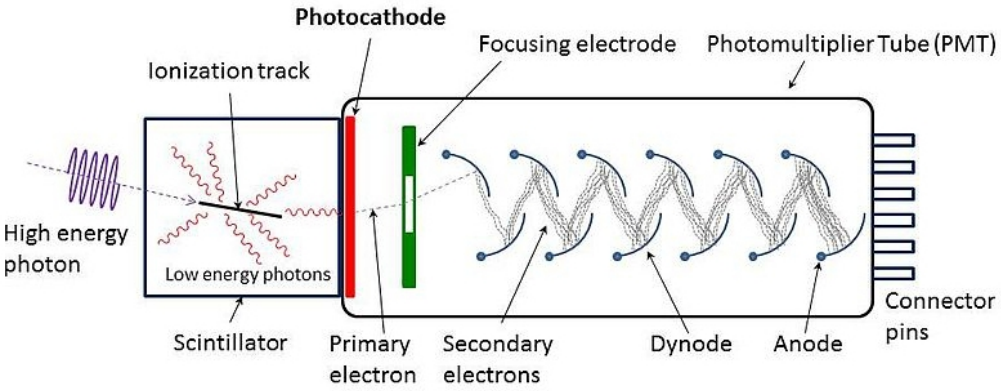


Figure 1.6: Apparatus with a scintillating crystal and photomultiplier[2]

[2] https://en.wikipedia.org/wiki/Scintillation_counter

1.3 Electronics

1.3.1 Amplifier

The amplifier further amplifies the signal of the pre-amplifier and forms the signals for further processing. This is done by adjusting the *shaping time*. The amplifier determines the maximum amplitude of the input pulse during the shaping time and generates an output signal after the end of the shaping time. A delay in time between the in- and output signal is caused by this. The shaping time should be chosen that the maximum amplitude is reached and idle time is minimised. The amplifier has a bipolar (which is not used in the experiment) and a unipolar output.

1.3.2 Multi Channel Analyser (MCA)

The MCA distributes the pulses according the voltage onto channels. For this purpose every pulse is broadened and compared with a sawtooth voltage which sets in at the same time. The time difference between the surge of the pulse and the reaching of the hight of the saw-tooth indicates the amplitude pertaining to the pulse. The channels are merely numbered and must be assigned their corresponding energy by calibration with known spectral lines (see section 3.1).

1.3.3 Single Channel Analyser (SCA)

The SCA serves the purpose of filtering a specific range of energy. The range is set by the *lower-* and *upper level*. If the energy of an inbound pulse lies in this energy window a rectangular pulse is measurable at the output of the SCA (see Fig. 1.7).

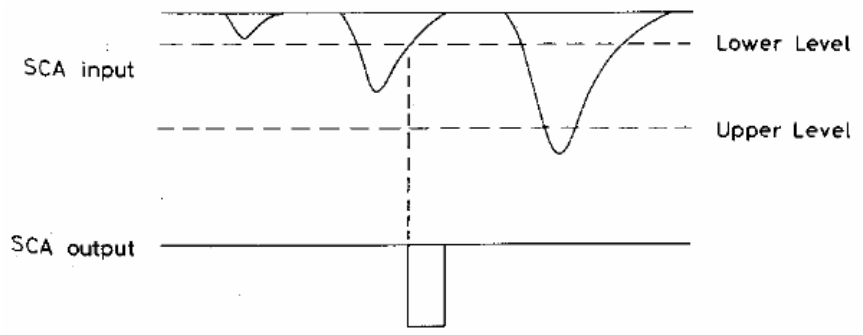


Figure 1.7: Output of the SCA (from [2])

1.3.4 Delay and Gate

The run-time of the signal can be altered by adjusting the cable length with a *delay*.
The gate is used in order to set a stretch of time in which the MCA registers a signal. The gate has to be wide enough for the pulse to reach its maximum. Otherwise the energies will be wrongly assigned leading to the warping of the spectrum.

2 Experimental Procedure

2.1 Tasks

1. Calibration of the MCA
2. Determination of the Compton-background
3. Determination of the attenuation caused by the acrylic glass window
4. Determination of (from the absorption spectrum of stainless steel (S) and natural iron (N)):
   (a) the isomeric shift (N+S)
   (b) the effective absorber thickness (S)
   (c) the Debye-Waller-factor f_S of the source (S)
   (d) the lifetime of the 14.4 keV of the ^{57}Fe (S)
   (e) the magnetic field and the magnetic moment of the 14.4 keV state (N)

2.2 Experimental Set-up

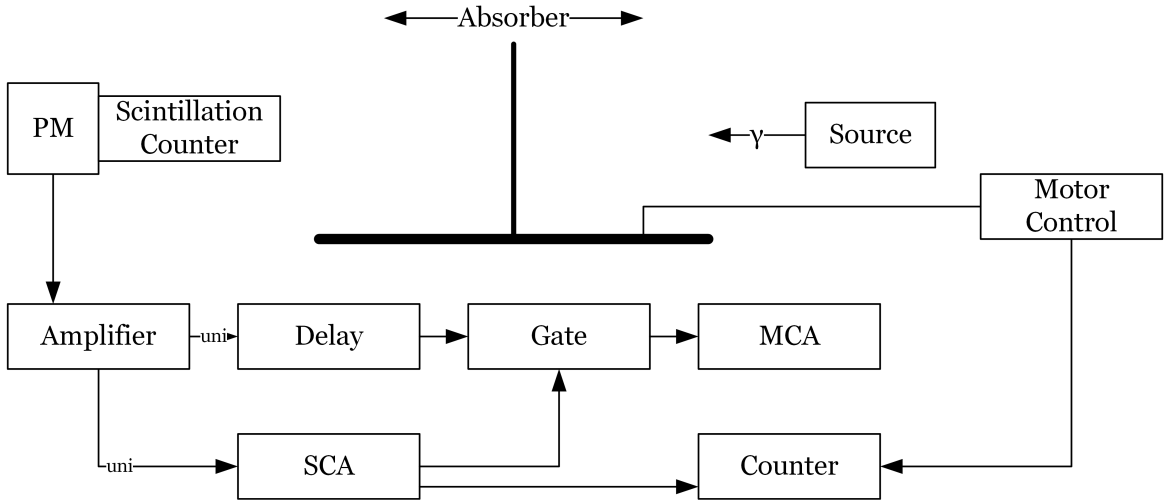


Figure 2.1: Experimental set-up

The experimental set-up (see Fig. 2.1) consisted of a radioactive source (^{57}Co), a moveable absorber and Sodium chloride scintillation counter. The scintillation counter was connected to the electronics described in section 1.3. The counters pre-amplifier was connected to the regular amplifier. The signal was then split into the SCA and the delay to counteract the difference in run time. The signals were reunited at the linear gate, which was set that only pulses with energies corresponding to the 14.4 keV transition could pass. The signal finally ended up at a counter or the MCA. The absorber was movable with an electric motor connected to the counter (velocities of 0.01-10.0 $\frac{\text{mm}}{\text{s}}$). The measurements took place when the absorber had reached a constant velocity. By measuring at different velocities the absorption spectra of stainless steel and natural iron could be recorded with the scintillation counter.

2.2.1 Calibration of the MCA

First the MCA (with 2048 channels) was calibrated by measuring the spectra of rubidium, molybdenum, silver and terbium. Before recording the spectra all the signals were visualised on an oscilloscope. The following settings were chosen for the amplifier:

- Amplification: 1.15
- Course gain: 50
- Shaping time: 2 µs

As the entire spectrum was of interest the gate was set to "DC-inhibit", letting all incoming signals pass through unhindered. Then the spectra of the probes were recorded for 600 s. After this the gate and delay were connected and the SCA adjusted for the measurement 14.4 keV peak. The settings of SCA and delay were:

- Delay: 3 µs
- Upper level: 1.1
- Lower level: 0.9

2.2.2 Determination of the Compton-background

The Compton-background was measured by gradually increasing the shielding of the scintillation counter with aluminium plates. The absorber was stationary for the entire duration of this measurement. The measurement time was $t = 300\,\mathrm{s}$ for thirty different shielding thicknesses. The number of counts was obtained by summing over the channels of the MCA.

2.2.3 Attenuation by the acrylic glass

To determine the attenuation caused by the acrylic glass, the counting rate was measured with a strip of acrylic glass and without an absorber for 15 minutes.

2.2.4 One line absorber (stainless steel)

The one line absorber was measured at velocities from 0.1-2.0 $\frac{\mathrm{mm}}{\mathrm{s}}$ in steps of 0.1 $\frac{\mathrm{mm}}{\mathrm{s}}$ for 300 s. Subsequently a finer measurement with a step size of 0.01 $\frac{\mathrm{mm}}{\mathrm{s}}$ from 0.01-1.19 $\frac{\mathrm{mm}}{\mathrm{s}}$ was performed.

2.2.5 Six line absorber (natural iron)

The six line absorber was measured at velocities from 0.1-8.0 $\frac{\mathrm{mm}}{\mathrm{s}}$ in steps of 0.1 $\frac{\mathrm{mm}}{\mathrm{s}}$ for 600 s.

3 Evaluation

3.1 Calibration of the MCA

The MCA was calibrated with the K_α lines of rubidium, molybdenum, silver and terbium. The channels were assorted their corresponding energy by fitting the K_α peaks with Gaussian functions (see Fig. 3.1-3.5):

$$n(ch) = A \cdot e^{-\frac{(ch-\mu)^2}{2\cdot\sigma^2}} + C + D \cdot ch \tag{16}$$

The fit parameters are listed in table 1.

Name	A	C	D	σ	μ	χ^2_{red}
Barium	89700 ± 500	3090 ± 200	21 ± 1	46.89 ± 0.26	300.80 ± 0.26	13.18
Molybdaenum	16970 ± 120	-1186 ± 152	26 ± 1	23.42 ± 0.21	200.06 ± 0.16	3.18
Rubidium	7885 ± 46	-773 ± 145	17 ± 1	20.32 ± 0.13	135.90 ± 0.10	0.98
Silver	68090 ± 660	-66 ± 500	14 ± 1	39.90 ± 0.40	212.68 ± 0.37	19.04
Terbium	94300 ± 200	12127 ± 121	0.43 ± 0.18	59.11 ± 0.13	$411.00 \pm .018$	2.17

Table 1: Overview of channels and energy for calibration

The χ^2 values are (with the exception of rubidium) too high, indicating that the model of one Gaussian function does not fully describe the data. As only the positions of the peaks were of interest, the Gaussian functions were still used. The µ values allow the position of the K_α line to be identified. Table 2 gives a summary of the channels and corresponding energies:

Element	Channel	Energy / keV
Barium	301 ± 23	32.06
Molybdenum	200 ± 12	17.44
Rubidium	136 ± 10	13.37
Silver	213 ± 20	22.10
Terbium	411 ± 30	22.10

Table 2: Overview of channels and energy for calibration

The uncertainties of the µ values (resulting from the fits) are unrealistically small. So these were not used the uncertainty of the channel position. Instead the uncertainties were chosen to be:

$$s_{ch} = \frac{\sigma}{2} \tag{17}$$

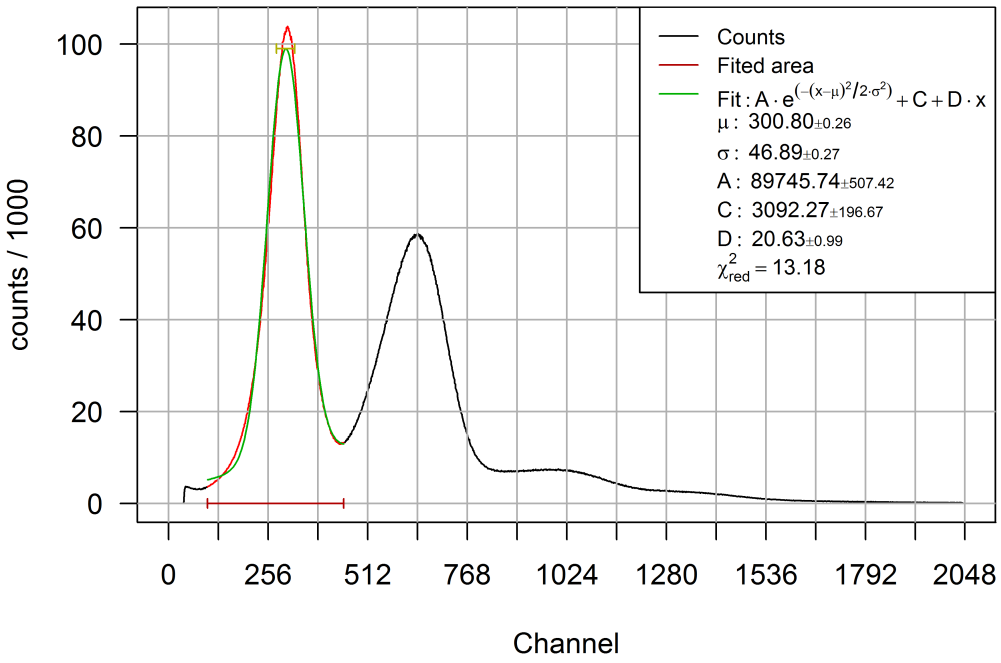


Figure 3.1: Gaussian fit of the barium K_α line

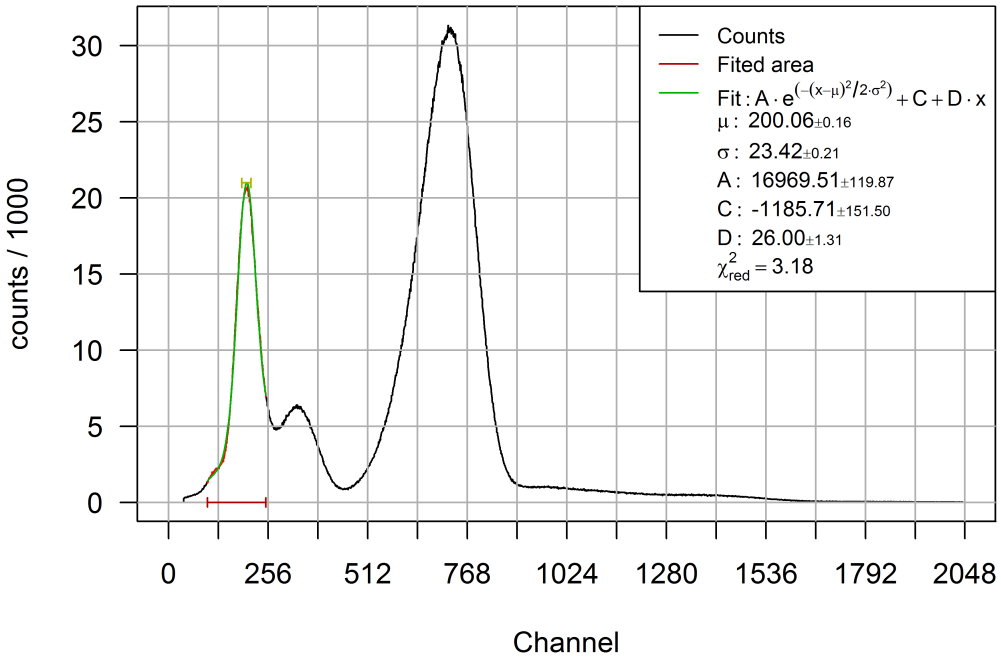


Figure 3.2: Gaussian fit of the molybdenum K_α line

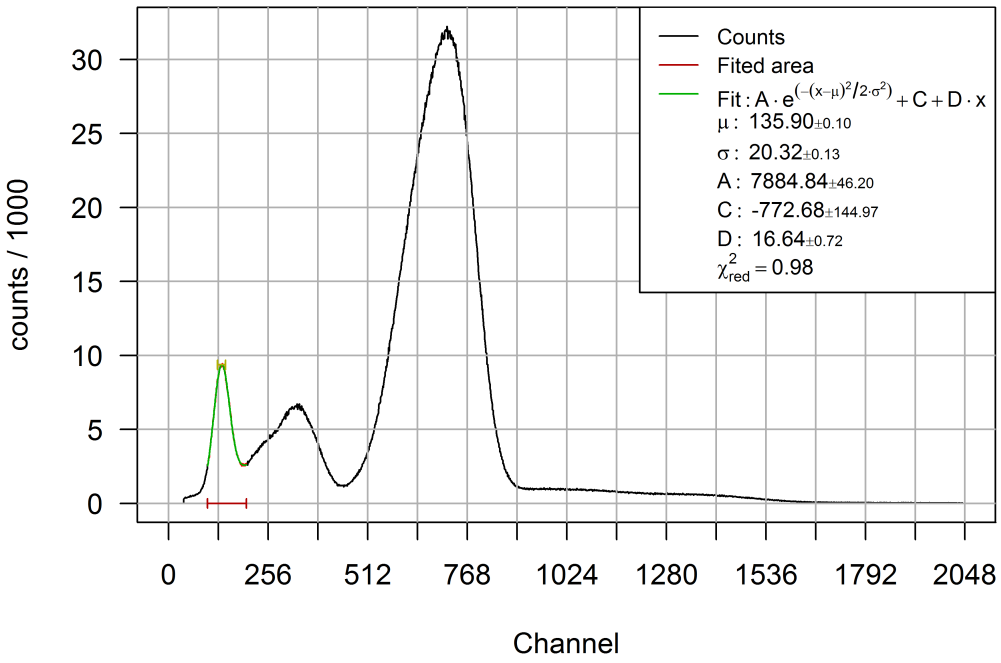


Figure 3.3: Gaussian fit of the rubidium K_α line

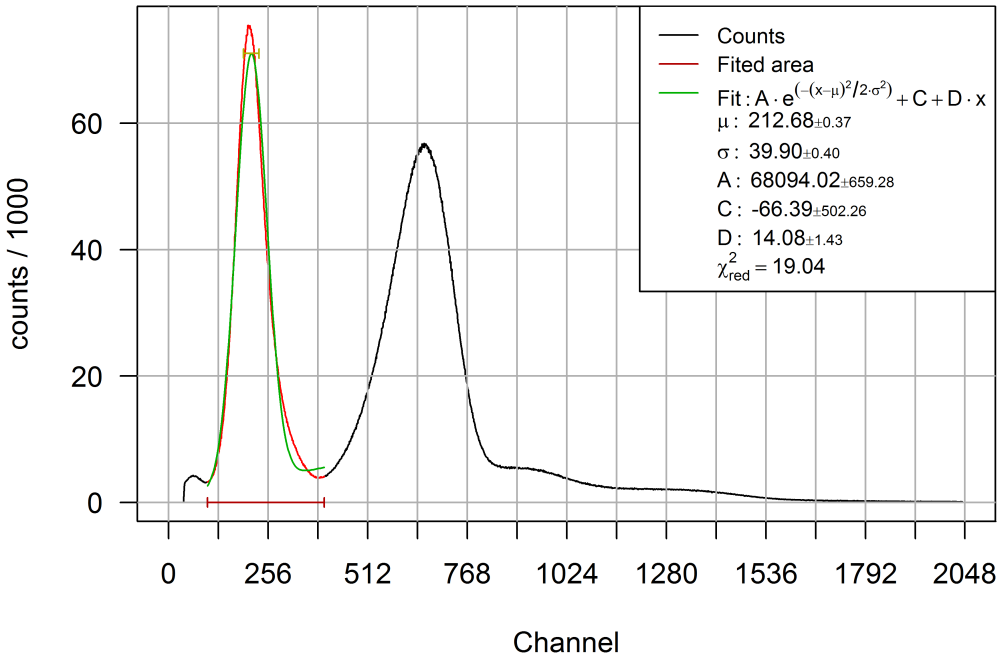


Figure 3.4: Gaussian fit of the silver K_α line

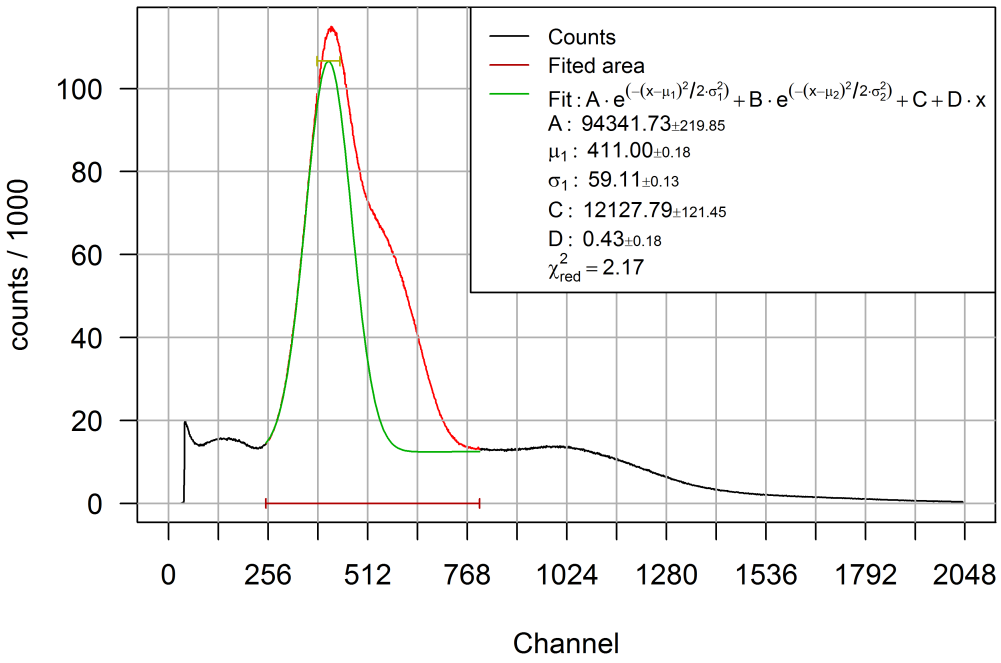


Figure 3.5: Gaussian fit of the terbium K_α line

The obtained channels were then plotted against their respective energy and a linear fit was performed to obtain a relationship between channel and energy (see figure 3.6).

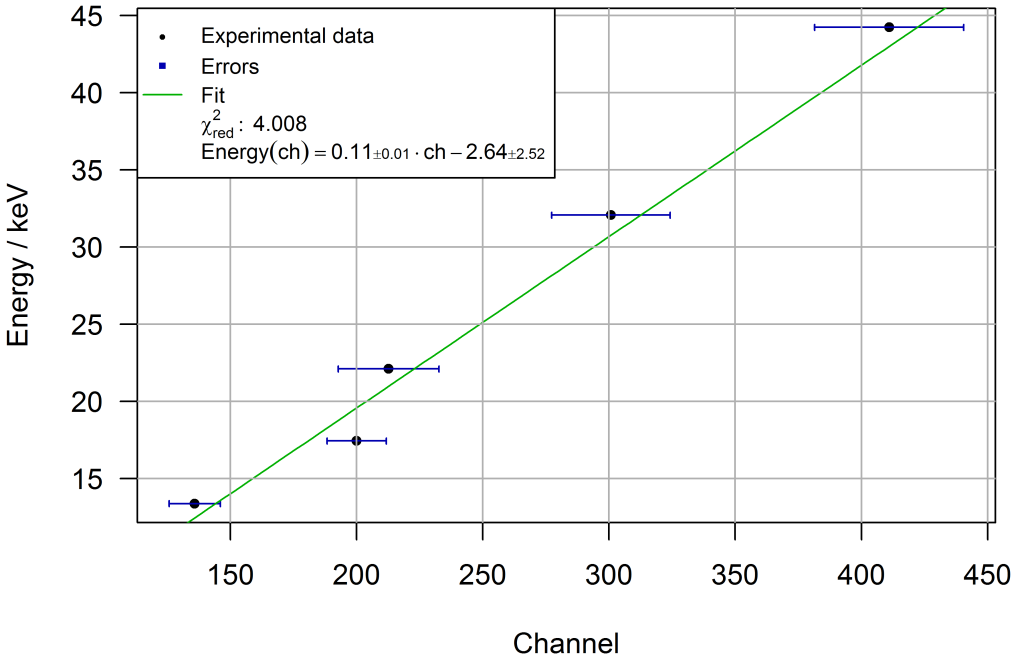


Figure 3.6: Linear fit for the energy calibration

The χ^2 value of 4 implies that the present relationship is not of a linear kind and higher order polynomials should be introduced. Additionally the error of the MCA (on which no information is present) may be far larger than the approximated values. The non-linearity was probably caused by

the scintillator or the photomultiplier.

The uncertainty of the energy was calculated with Gaussian error propagation:

$$E(ch) = A + B \cdot ch \tag{18}$$

$$s_E = \sqrt{s_A^2 + (s_B \cdot ch)^2} \tag{19}$$

With this energy calibration 14.4 keV peak in the cobalt spectrum was fitted with Gaussian function (see Fig. 3.7). The error bar represents the $\frac{\sigma}{2}$ range that was chosen as uncertainty.

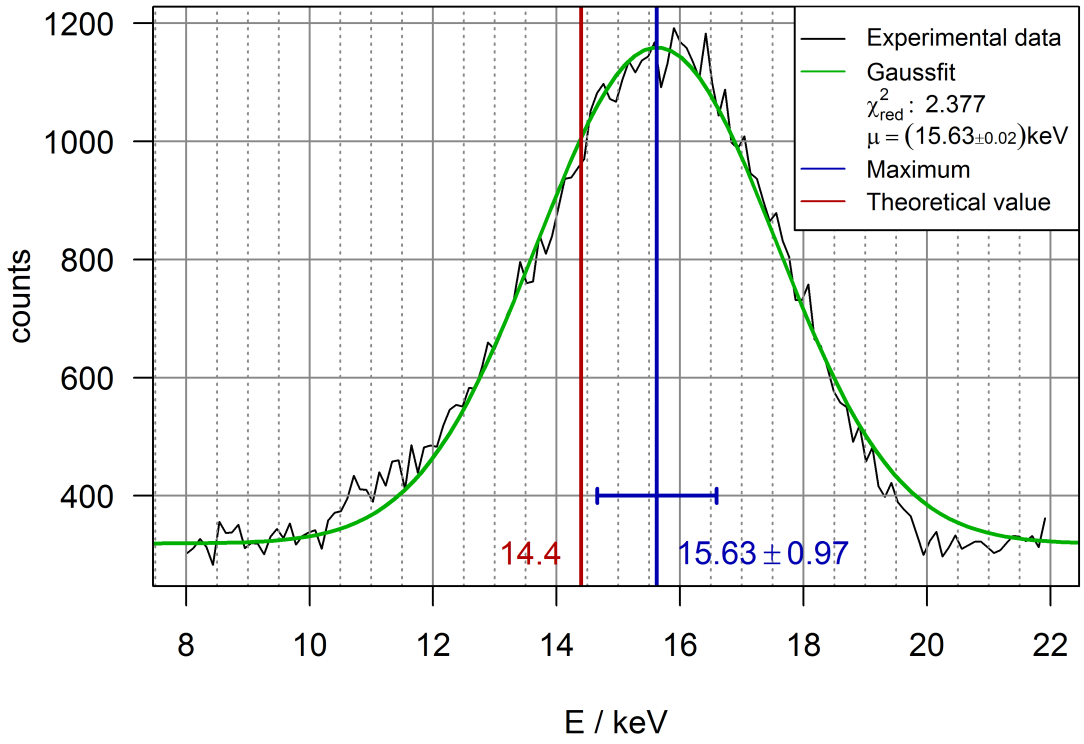


Figure 3.7: Gaussian fit of the 14.4 keV peak with energy resulting from the calibration

As can be seen from the figure the calibration has failed to assign the correct energy value to the peak. The MCA allocates the peak an energy of 15.63 keV, which is more than 1 keV higher than the expected value of 14.4 keV. This indicates the alarmingly high uncertainty associated with the used electronics, which are discussed in detail in section 4. Due to the fact that the energy calibration is irrelevant for the further measurements, it's inaccuracy has no effect on their outcome.

3.2 Determination of the Compton-background

High energy photons can lose energy by Compton-scattering with the electrons of the absorber. If they have lost sufficient energy to lie within the 14.4 keV SCA window, they are detected by the apparatus. Most of the Compton scattering is caused by the 122 keV and 136 keV transitions ([4]). This background must be subtracted from the measured Mößbauer spectra. Measuring it is accomplished by inserting aluminium plates in front of the scintillation counter. The counting rate ($\dot{N} = \frac{N}{\Delta t}$) decreases exponentially with the thickness of the aluminium plates (d):

$$\dot{N} = \dot{N}_0 e^{-\mu d} \tag{20}$$

μ is the mass attenuation coefficient which depends on the photon energy. The assumption was made that the mass attenuation coefficients for the two most probable transitions (14.4 keV and 122 keV) are the same. The thickness of the aluminium plates was measured with a measuring calliper and an uncertainty of $s_x = 0.002\,\text{mm}$. The counting rate was plotted against the thickness of the plate and fitted with the sum of two exponential functions (see Fig. 3.8):

$$\dot{N} = Ae^{ad} + Be^{bd} \tag{21}$$

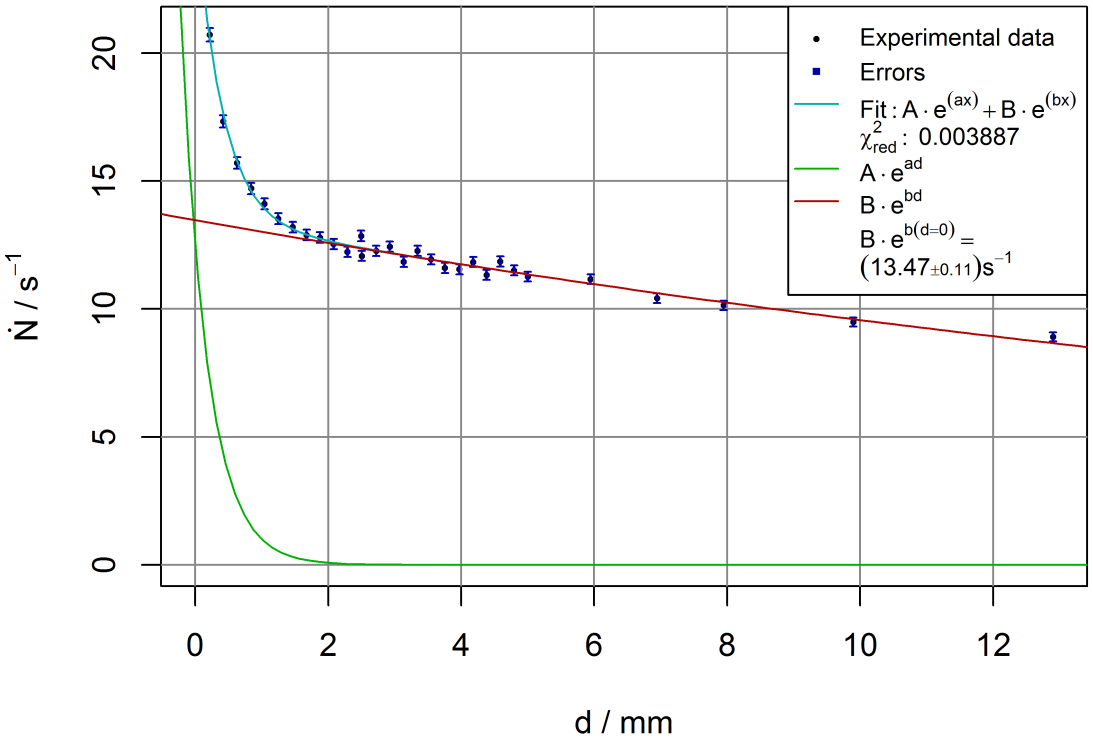


Figure 3.8: Sum of two exponential fits for the determination Compton-background

The counts are Poisson distributed, which leads to the uncertainty of the counting rate (with a measurement time of $t = 300\,\text{s}$):

$$s_{\dot{N}} = \frac{\sqrt{N}}{t} \tag{22}$$

The uncertainty of the thickness of the aluminium shielding was determined with error propagation:

$$s_d = \sqrt{n} s_x \tag{23}$$

n is the number of aluminium plates. This uncertainty is so small that the error bars are not noticeable. The exponential term pertaining to the Compton-scattering was extracted from the fit:

$$\dot{N}_B = (13.47 \pm 0.11) e^{(0.034 \pm 0.002)d} \tag{24}$$

Extrapolating this part to $d = 0$ gives the counting rate caused by the Compton-background, which was subtracted from all subsequent measurements:

$$\dot{N}_{Compton} = (13.47 \pm 0.11)\,\text{cps} \tag{25}$$

The relative uncertainty is less than 1%, meaning that the correction can be performed without hesitation. The uncertainty of the corrected counting rate ($\dot{N}_{cor}$) was calculated with error propagation:

$$s_{\dot{N}_{cor}} = \sqrt{s^2_{\dot{N}_{Compton}} + s^2_{\dot{N}_{uncor}}} \tag{26}$$

3.3 Attenuation by the acrylic glass

The counting rate was measured with and without a strip of acrylic glass. The thickness of the acrylic glass was determined to be:

$$d = (1.985 \pm 0.005)\,\text{mm}$$

This is roughly the thickness of the acrylic glass in the absorber casing. The counting rates were:

$$\dot{N}_0 = (66.9 \pm 0.3)\text{cps}$$
$$\dot{N}_{acryl} = (51.9 \pm 0.2)\text{cps}$$

The acrylic glass has led to a noticeable attenuation of the gamma rays. The attenuation coefficients of acrylic glass for a plethora of photon energies can be found in a table in [1] Intense studying of this table leads to the suspicion that there may be an exponential relationship between energy and mass attenuation coefficient. As 14.4 keV has not been assorted a value it was instructive to perform an exponential fit of the surrounding data points to determine μ/ρ T 14.4 keV. The resulting fit can be marvelled at in Fig. 3.9.

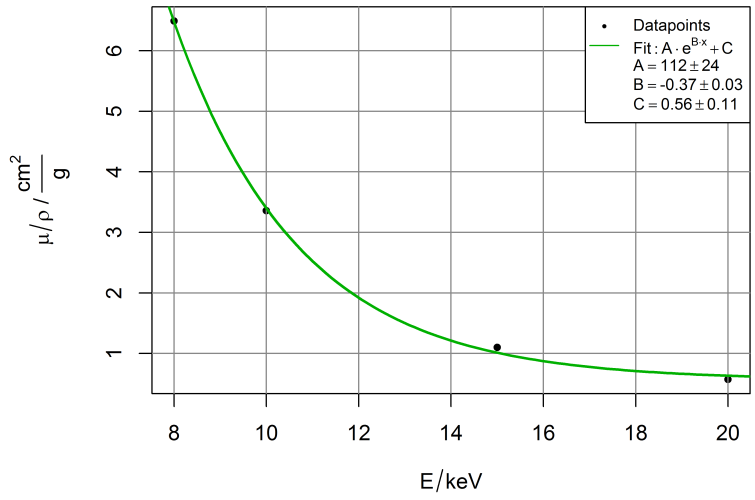


Figure 3.9: Exponential fit of the attenuation coefficients in the neighbourhood of 4.4keV

The attenuation coefficient at 14.4 keV can be extracted from the fit:

$$\mu/\rho = (1.1 \pm 0.3)\,\frac{\mathrm{cm}^2}{\mathrm{g}} \tag{27}$$

The density of the acrylic glass is $\rho = 1.19\frac{\mathrm{g}}{\mathrm{cm}^3}$ ([1]). With this density the mass attenuation coefficient is:

$$\mu = (1.3 \pm 0.3)\,\frac{1}{\mathrm{cm}}$$

With this it is possible to calculate the theoretical counting rate:

$$\dot{N}_{acryl,theo} = N_0 e^{-\mu d} = (51 \pm 3)\,\mathrm{cps}$$

The theoretical value matches the measured one in a range of 1σ. This implies that the determined mass attenuation coefficient describes the shielding of the scintillation counter by the acrylic glass. The astute reader may have noticed that the attenuation of the radiation by the acrylic glass only results in a shift of the counting rate (as it is velocity independent) and the attenuation is not very high. As this is irrelevant for the upcoming measurements the ground breaking value of the mass attenuation coefficient is of no practical use whatsoever.

3.4 Single line absorber (stainless steel)

The recorded spectrum of the single line absorber (stainless steel) can be seen in Fig. 3.10. After removing the Compton-background from the counting rate, the experimental data were fitted with a Gaussian-, Lorentz- and a Voigt-function (see Fig. 3.10):

- Gaussian: $f(x) = A \cdot e^{-\frac{(x-\mu)^2}{2\cdot\sigma^2}} + C$
- Lorentz: $\frac{\Gamma}{\pi(x-\mu)^2+\Gamma^2}$ (This is the theoretical prediction)
- Voigt: $f(x) = A \cdot Voigt(x, \Gamma, \sigma, \mu) + C$
  The Voigt function is a convolution of Lorentz- and Gaussian functions (computed numerically). The Gaussian function accounts for inaccuracies of the experimental set-up (unreliable motor and broadening by the absorber).

The resulting fit parameters are listed in table 3.

The Gaussian fit represents the data poorly as it is far too wide at the position of the minimum. The Lorentz- and Voigt fit are practically identical and match the data rather well. However the Voigt fit was chosen because it reflects the experimental conditions better from a theoretical point of view.

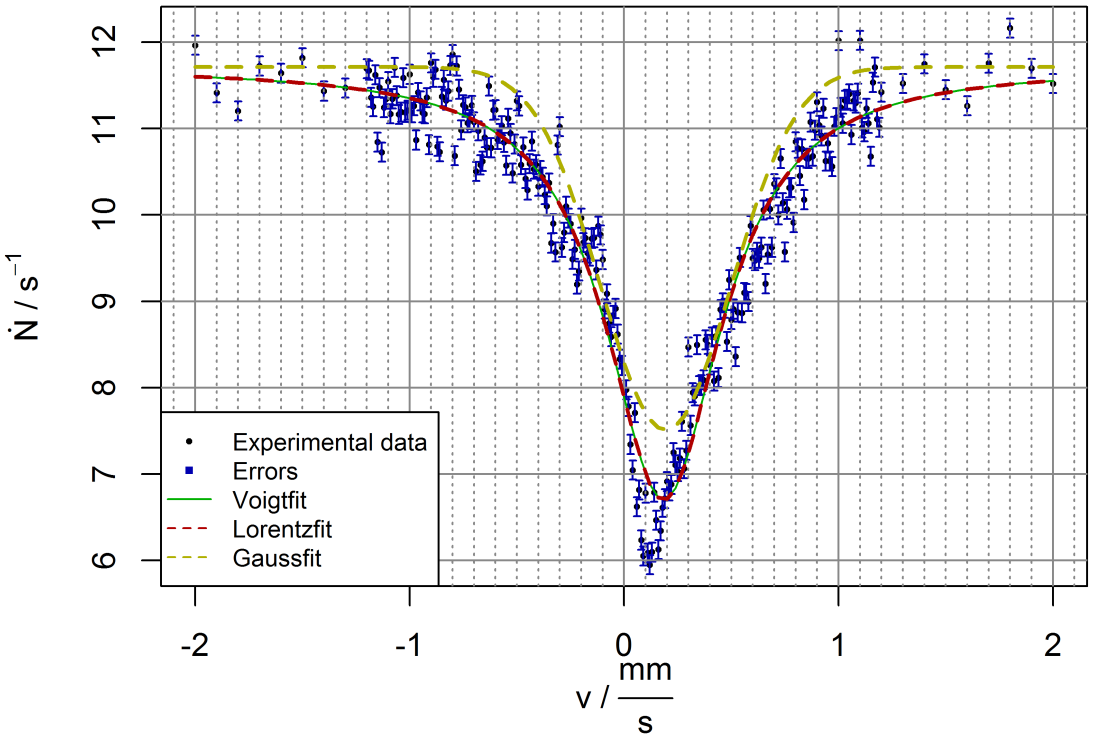


Figure 3.10: Fits of the one line absorber spectrum

Function	A	C	μ	σ	Γ	χ^2_{red}
Voigt	-5.21 ± 0.22	11.71 ± 0.06	0.185 ± 0.005	$6.6 \cdot 10^{-5} \pm 29$	0.331 ± 0.021	11.94
Gauss	-3.19 ± 0.09	11.27 ± 0.04	0.195 ± 0.006	0.302 ± 0.007		15.30
Lorentz	-5.20 ± 0.18	11.71 ± 0.05	0.185 ± 0.005		0.331 ± 0.011	11.89

Table 3: Fit parameters for the one line absorber

3.4.1 Isomeric shift

The isomeric shift is simply the shift of the minimum from zero. The shift of the curve is given by the fit parameter μ. The results are (as velocity and energy):

$$\mathbf{v_{iso}} = (0.185 \pm 0.005)\ \tfrac{\text{mm}}{\text{s}}$$
$$\mathbf{E_{iso}} = (8.9 \pm 0.2)\text{neV}$$

3.4.2 Effective absorber thickness

The calculation of the effective absorber thickness requires several values:

The angular frequency (ω_0) of the 14.4 keV photon is obtained by dividing the energy by the reduced Planck constant:

$$\omega_0 = \frac{14.4\text{keV}}{\hbar} = \frac{14.4\,\text{keV}}{6.582 \cdot 10^{-16}\,\text{eVs}} = 2.19 \cdot 10^{-1}\,\text{s}^{-1} \tag{28}$$

The second quantity is the number of ^{57}Fe atoms in one cubic meter ($\mathbf{n_a}$). This is done as follows:

$$\begin{aligned}
\rho_{Fe} &= 7.87\ \tfrac{\mathrm{g}}{\mathrm{cm}^3} && \text{(Density of Iron)}\\
M_{Fe} &= 55.85\ \tfrac{\mathrm{g}}{\mathrm{mol}} && \text{(Molar mass)}\\
\alpha_{Fe} &= (70 \pm 5)\,\% && \text{(Percentage of iron in absorber [1])}\\
N_A &= 6.022 \cdot 10^{23}\ \tfrac{1}{\mathrm{mol}} && \text{(Avogadro constant)}\\
\mathbf{n_a} &= \frac{\rho_{Fe}}{M_{Fe}} N_A \alpha_{Fe}\\
&= (\mathbf{59.4 \pm 4.2}) \cdot \mathbf{10^{27}}\ \tfrac{\mathbf{1}}{\mathbf{m^3}}
\end{aligned}$$

The last value that is necessary for the calculation is the *cross section* of the transition (σ_0). The formula is from [4] and the uncertainty was calculated with error propagation:

$$\begin{aligned}
c &= 3 \cdot 10^8\ \tfrac{\mathrm{m}}{\mathrm{s}} && \text{(Speed of light)}\\
\alpha &= 8.9 \pm 0.6 && \text{(Conversion coefficient [6])}\\
I &= \frac{1}{2} && \text{(Nuclear spin of ground state)}\\
I^* &= \frac{3}{2} && \text{(Nuclear spin of excited state)}\\
\sigma_\mathbf{0} &= 2\pi \left(\frac{c}{\omega_0}\right)^2 \left(\frac{1}{1+\alpha}\right) \left(\frac{2I^*+1}{2I\ \ +1}\right)\\
&= (\mathbf{2.4 \pm 0.2}) \cdot \mathbf{10^{22}}\ \tfrac{\mathbf{1}}{\mathbf{m^2}}
\end{aligned}$$

The calculation of the (dimensionless) *effective absorber thickness* T_A is now possible with following formula (uncertainty from error propagation):

$$\begin{aligned}
f_a &= .8 && \text{(Debye-Waller-factor)}\\
\beta &= 0.022 && \text{(proportion of } ^{57}\text{Fe isotopes in the sample)}\\
\sigma_0 &= (2.39 \pm 0.17) \cdot 10^{22}\ \tfrac{1}{\mathrm{m}^2}\\
d_a &= 25 \cdot 10^{-6}\mathrm{m} && \text{(thickness of absorber)}\\
n_a &= (59.4 \pm 4.2) \cdot 10^{27}\ \tfrac{1}{\mathrm{m}^3}\\
\mathbf{T_A} &= f_a \beta \sigma_0 d_a n_a\\
&= \mathbf{6.2 \pm 0.6}
\end{aligned}$$

3.4.3 Determination of the lifetime from the effective absorber thickness

With the calculated absorber thickness one can obtain the relative broadening of the peak. To do so the calculated value is located on the x-axis of the graph in Fig. 3.11. Then the value of the *unsplit* line with it's corresponding value on the y-axis is graphically determined. This was accomplished with high accuracy by editing a high resolution picture in Adobe Photoshop (see Fig. 3.11).

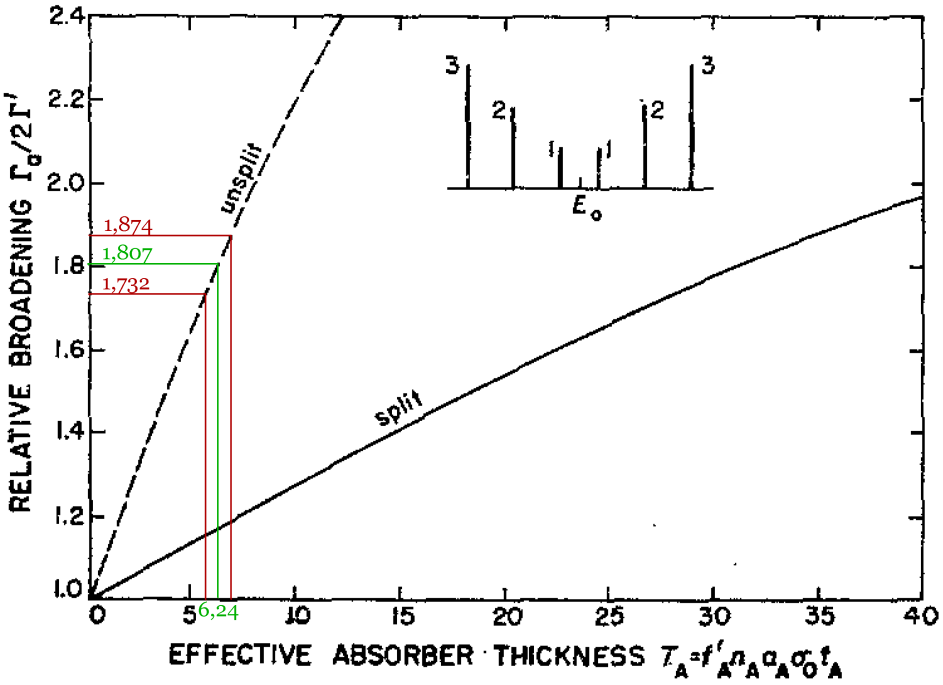


Figure 3.11: Graphically obtaining the relative broadening from the effective absorber thickness (from [4])

The error was determined by also reading off the relative broadening for $T_a \pm \sigma_{T_a}$. The difference between main value T_a and those of $T_a \pm \sigma_{T_a}$ was set as the uncertainty of the relative broadening. As the differences slightly differ the greater one was chosen. The final results are listed table 4.

	T_A	$\frac{\Gamma_0}{2\Gamma_{nat}}$
$\mathbf{T_A}$	**6.24**	**1.807**
$T_A - \sigma_{T_A}$	5.61	1.732
$T_A + \sigma_{T_A}$	6.87	1.874

Table 4: from absorber thickness to broadening

The resulting value for the relative broadening with its uncertainty is:

$$\frac{\Gamma_0}{2\Gamma_{nat}} = \frac{W}{2} = 1.81 \pm 0.07$$

The natural line width was then calculated with the width of the fitted Voigt-function Γ (see table 3). The uncertainty was calculated with error propagation:

$$\Gamma = \frac{\Gamma}{W/2} \qquad s_\Gamma = \sqrt{\left(\frac{1}{W/2}\right)^2 \cdot s_\Gamma^2 + \left(\frac{\Gamma}{(W/2)^2}\right)^2 \cdot s_{W/2}^2}$$

$$\mathbf{\Gamma} = (\mathbf{4.39 \pm 0.33})\,\text{neV}$$

The lifetime of the excited state follows from the energy-time uncertainty. It states that $\Gamma\tau \geq \hbar$. With this formula it is possible to receive a minimum lifetime τ and the half life $T_{1/2}$ of the excited state.

$$\tau = \frac{\hbar}{\Gamma} \tag{29}$$

$$T_{1/2} = ln(2) \cdot \tau \tag{30}$$

The calculations resulted in following values. The errors were calculated using Gaussian error propagation.

$$\tau = (\mathbf{150 \pm 10})\,\mathrm{ns} \tag{31}$$

$$\mathbf{T_{1/2}} = (\mathbf{104 \pm 8})\,\mathrm{ns} \tag{32}$$

The literature values are (from [1]):

$$\tau_{\mathbf{Lit}} = 141\,\mathrm{ns} \tag{33}$$

$$\mathbf{T_{1/2,Lit}} = 98\,\mathrm{ns} \tag{34}$$

The experimentally determined values boast rather large uncertainties but are compatible with the literature values in a range of 1σ.

3.4.4 Debye-Waller-factor of the source

The calculation of the Debye-Waller factor of the source (f_S) is explained in [4]. Formula 11 from this publication states that

$$\frac{P(\infty) - P(\vartheta_1)}{P(\infty)} = fW_i^{(S)} \left[1 - \exp\left(-\frac{1}{2}T_A\right) J_0\left(\frac{1}{2}iT_A\right)\right] \tag{35}$$

Here $fW_i^{(S)}$ is the Debye-Waller factor f_S and $P(\infty)$ is the counting rate at an infinite velocity (no resonance absorption) which is obtained from the fit in figure 3.10.

$P(\vartheta_1)$ is the minimal counting rate (maximum resonance absorption). It can be calculated by inserting the isomeric shift into the resulting Voigt-function from the fit. The error was calculated with Gaussian error propagation. The uncertainties of the shift along the counting rate axis (C) and the scaling factor (A) resulted directly from the fit.

The effective absorber thickness T_A was calculated in 3.4.2.
Finally J_0 is the Bessel-function of order zero (i is the imaginary unit).

The Debye-Waller factor is obtained by rearranging formula 35:

$$fW_i^{(S)} = f_S = \frac{P(\infty) - P(\vartheta_1)}{P(\infty)\left[1 - \exp\left(-\frac{1}{2}T_A\right) J_0\left(\frac{1}{2}iT_A\right)\right]}$$

The resulting value for the Debye-Waller factor is:

$$\mathbf{f_S = 0.56 \pm 0.02}$$

This means that slightly over half of the inbound photons are emitted without recoil from the source.

The uncertainty was calculated with Gaussian error propagation. The formula was computed with Mathematica:

$$s_{f_S} = \sqrt{\left(\frac{\partial f_S}{\partial P(\infty)}\right)^2 s^2_{P(\infty)} + \left(\frac{\partial f_S}{\partial P(\vartheta_1)}\right)^2 s^2_{P(\vartheta_1)} + \left(\frac{\partial f_S}{\partial T_A}\right)^2 s^2_{T_A}} \tag{36}$$

The contributing factors and uncertainties are listed in table 5:

contribution	value	unit
$P(\infty)$	11.71 ± 0.06	s^{-1}
$P(\vartheta_1)$	6.70 ± 0.22	s^{-1}
T_A	6.2 ± 0.6	

Table 5: Contributing factors for the calculation of f_S

3.4.5 Lifetime of the excited state (from the Voigt-fit)

The lifetime of the excited state can also be directly obtained from the width of absorption peak with energy-time uncertainty principle (as was done in section 3.4.3). The value for Γ ($(8.9 \pm 0.2)\,\mathrm{neV}$) was extracted from the above fit (see Fig. 3.10 and the lifetime was calculated. The uncertainty was determined with error propagation.

$$\tau_{\mathbf{fit}} = (\mathbf{83 \pm 5})\,\mathbf{ns}$$

The literature value was 141 s. The lifetime which was extracted directly from the Voigt-fit contradicts the literature value and the discrepancy is very large (11σ).

3.5 Six line absorber (natural iron)

After removing the Compton-background from the counting rate the experimental data was fitted with six Lorentz functions (see Fig. 3.12).

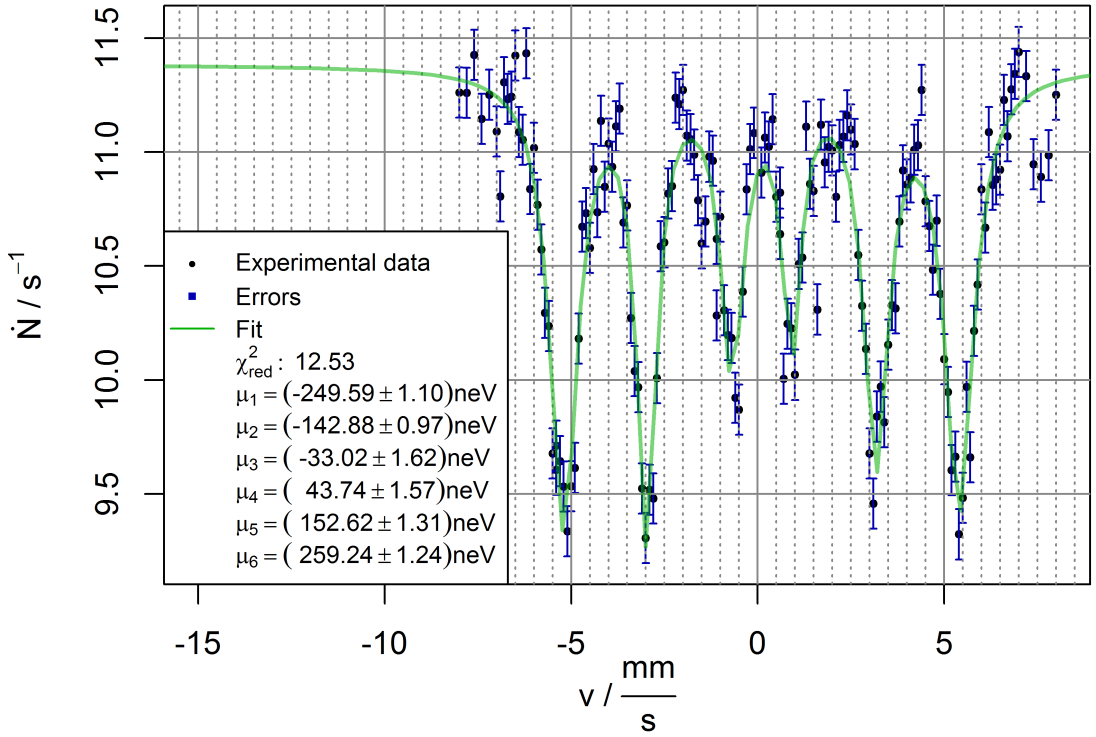


Figure 3.12: Lorentzfits of the six line absorber

The reason why the fit was performed with Lorentz functions is that Gaussian functions were too broad and the Voigt functions failed to converge. The χ^2 square of 12.58 seems too high but can be explained by statistical scattering of the data points and the complex nature of the function.

3.5.1 Isomeric shift

To obtain the isomeric shift one has to once again determine the offset from zero. This was done by calculating half the distance between every pair of symmetric peaks. A summary of the positions of the peaks is given in table 6. The calculated isomeric shifts are shown in table 7. A weighted mean of these values was calculated and also listed in the table along with the literature value.

peak	position / mm/s	position / neV
1	-5.20 ± 0.02	-250 ± 1
2	-2.98 ± 0.02	-143 ± 1
3	-0.69 ± 0.03	$-\ 33 \pm 2$
4	0.91 ± 0.03	44 ± 2
5	3.18 ± 0.03	153 ± 1
6	5.40 ± 0.03	259 ± 1

Table 6: Peaks in the six line spectra

peaks	v_{iso} / mm/s	E_{iso} / neV
1 & 6	0.10 ± 0.02	4.8 ± 0.7
2 & 5	0.10 ± 0.02	4.9 ± 1.1
3 & 4	0.11 ± 0.02	5.4 ± 0.9
weighted mean	0.10 ± 0.01	5.0 ± 0.5
literature[3]	0.11	5.3

Table 7: Isomeric shift in the six line absorber

The literature value is compatible with the experimental value in a range of 1σ.

3.5.2 Magnetic moment

The energies of the absorption peaks are given by Formula 15:

$$E_i = E_{iso} - E_{HFS} = E_{iso} - \left(\frac{\mu_e m_e}{I_e} - \frac{\mu_g m_g}{I_g} \right) B \tag{37}$$

In this case E_{iso} is the isomeric shift, μ_e, μ_g, I_e, I_g are the nuclear magnetic moments and the nuclear spins for the excited (e) or ground state (g). Substituting the energy with the corresponding velocity values simplifies later calculations:

$$v_i = v_{iso} - v_{HFS} = v_{iso} - \frac{c}{E_0} \left(\frac{\mu_e m_e}{I_e} - \frac{\mu_g m_g}{I_g} \right) B \tag{38}$$

The nuclear spins of the present ground and excited states are $I_g = 1/2$ and $I_e = 3/2$. The magnetic quantum numbers are shown in following table:

peak	m_e	m_g
1	3/2	1/2
2	1/2	1/2
3	-1/2	1/2
4	1/2	-1/2
5	-1/2	-1/2
6	-3/2	-1/2

Table 8: Peaks in the iron spectra

With these quantum numbers and formula 38 it is possible to calculate the difference in velocity for two corresponding peaks:

$$\Delta v_a = v_6 - v_1 = \frac{2c}{E_0} \left(-\mu_e + \mu_g \right) B \tag{39}$$

$$\Delta v_b = v_5 - v_2 = \frac{2c}{E_0} \left(-\frac{\mu_e}{3} + \mu_g \right) B \tag{40}$$

$$\Delta v_c = v_4 - v_3 = \frac{2c}{E_0} \left(\frac{\mu_e}{3} + \mu_g \right) B \tag{41}$$

[3] Datasheet of the sample

The uncertainties are as follows:

$$s_{\Delta v_a} = \sqrt{s_{v_6}^2 + s_{v_1}^2}$$
$$s_{\Delta v_b} = \sqrt{s_{v_5}^2 + s_{v_2}^2}$$
$$s_{\Delta v_c} = \sqrt{s_{v_4}^2 + s_{v_3}^2}$$

The uncertainties of the velocities are all extracted from the μ values of the corresponding Lorentz fit (see Fig. 3.12). Dividing the velocity differences by one another cancels out the unknown magnetic field (B). The literature value (from [5]) for the magnetic moment of the ground state is:

$$\mu_g = (0.09044 \pm 0.0007) \cdot \mu_N$$

With the nuclear magneton:

$$\mu_N = 3.152 \cdot 10^{-20} \frac{\text{eV}}{\text{T}}$$

With these values the calculation of the magnetic moment of the excited state is possible:

$$\frac{\Delta v_a}{\Delta v_b} = \frac{-\mu_e + \mu_g}{-\frac{\mu_e}{3} + \mu_g} \quad \Longrightarrow \quad \mu_{e,ab} = (-0.153 \pm 0.004)\mu_N$$
$$\frac{\Delta v_a}{\Delta v_c} = \frac{-\mu_e + \mu_g}{\frac{\mu_e}{3} + \mu_g} \quad \Longrightarrow \quad \mu_{e,ac} = (-0.159 \pm 0.002)\mu_N$$
$$\frac{\Delta v_b}{\Delta v_c} = \frac{-\frac{\mu_e}{3} + \mu_g}{\frac{\mu_e}{3} + \mu_g} \quad \Longrightarrow \quad \mu_{e,bc} = (-0.159 \pm 0.003)\mu_N$$

The errors were calculated with these formulas:

$$s_{\mu_{e,ab}} = \frac{1}{A^2} \cdot \sqrt{\left((\Delta v_a - \Delta v_b) \cdot A \cdot s_{\mu_g}\right)^2 + \left(-6 \cdot \mu_g \cdot \Delta v_b \cdot s_{\Delta v_a}\right)^2 + \left(6 \cdot \mu_g \cdot \Delta v_a \cdot s_{\Delta v_b}\right)^2}$$
$$s_{\mu_{e,ac}} = \frac{1}{B^2} \cdot \sqrt{\left((\Delta v_c - \Delta v_a) \cdot B \cdot s_{\mu_g}\right)^2 + \left(-12 \cdot \mu_g \cdot \Delta v_c \cdot s_{\Delta v_a}\right)^2 + \left(12 \cdot \mu_g \cdot \Delta v_a \cdot s_{\Delta v_c}\right)^2}$$
$$s_{\mu_{e,bc}} = \frac{1}{C^2} \cdot \sqrt{\left((\Delta v_c - \Delta v_b) \cdot C \cdot s_{\mu_g}\right)^2 + \left(-6 \cdot \mu_g \cdot \Delta v_c \cdot s_{\Delta v_b}\right)^2 + \left(6 \cdot \mu_g \cdot \Delta v_b \cdot s_{\Delta v_c}\right)^2}$$

with

$$A = \Delta v_a - 3 \cdot \Delta v_b$$
$$B = \Delta v_a + 3 \cdot \Delta v_c$$
$$C = \Delta v_b + \Delta v_c$$

A mean of these values was not calculated because the individual values are correlated by the used formulas. The three values of the magnetic moment are listed in 9 together with the literature value (from [5]).

	Value / μ_N
$\mu_{e,ab}$	-0.153 ± 0.004
$\mu_{e,ac}$	-0.159 ± 0.002
$\mu_{e,bc}$	-0.159 ± 0.003
literature	-0.1549 ± 0.0002

Table 9: Calculated values for the magnetic momentum and literature value

The experimental values of the magnetic moment match the literature value in a range of 2σ.

3.5.3 Magnetic field at the position of the nucleus

The magnetic field at the position of the nucleus can be calculated with the value for the magnetic moment of the excited state. This follows directly from formula 41. Gaussian error propagation yields the corresponding errors and is given below (formula 42). The resulting values for B and the literature values can be found in 10.

	Value / T
B_a	33.1 ± 1.9
B_b	32.7 ± 0.6
B_c	32.6 ± 2.7
literature	≈ 33

Table 10: Experimental magnetic field values and literature value (from [8])

$$s_{B_a} = |B_a| \cdot \sqrt{\left(\frac{s_{\Delta v_a}}{\Delta v_a}\right)^2 + \left(\frac{s_{\mu_{e,ab}}}{\mu_{e,ab} + \mu_g}\right)^2 + \left(\frac{s_{\mu_g}}{\mu_{e,ab} + \mu_g}\right)^2} \tag{42}$$

$$s_{B_b} = |B_b| \cdot \sqrt{\left(\frac{s_{\Delta v_b}}{\Delta v_b}\right)^2 + \left(\frac{s_{\mu_{e,ac}}}{-\frac{\mu_{e,ac}}{3} + \mu_g}\right)^2 + \left(\frac{s_{\mu_g}}{-\frac{\mu_{e,ac}}{3} + \mu_g}\right)^2} \tag{43}$$

$$s_{B_c} = |B_c| \cdot \sqrt{\left(\frac{s_{\Delta v_c}}{\Delta v_c}\right)^2 + \left(\frac{s_{\mu_{e,bc}}}{\frac{\mu_{e,bc}}{3} + \mu_g}\right)^2 + \left(\frac{s_{\mu_g}}{\frac{\mu_{e,bc}}{3} + \mu_g}\right)^2} \tag{44}$$

The experimental values have rather high uncertainties and are therefore compatible with the literature value in a range of 1σ.

4 Summary

4.1 Calibration of the MCA

After the energy calibration the 14.4 keV was located at (15.63 ± 0.97). The values are incompatible in a 1σ range, despite the fact that a rather large uncertainty of $\frac{\sigma}{2}$ resulting from the Gaussian fit parameter was chosen. This implies that there are serious sources of errors present.

One source of error which does not originate from the experimental set-up is the occurrence of Compton scattering, which may have shifted the positions of the peaks.

A number of sources of disturbance for the photomultiplier exist. One can certainly say that the energy resolution of the scintillation counter deteriorated due to dark noise. The dark noise is mainly of thermal origin. This leads to thermal excitation of photoelectrons (on the photocathode or the dynodes) which can cause avalanche-like discharges of their own. As there was no cooling unit this effect may have increased during the course of the experiment. The dark noise may also have been caused by electronic devices. The built in pre-amplifier in particular may have played a non negligible role in this. It follows that a constant amplification of the inbound radiation by the photomultiplier is highly unlikely.

The scintillator itself is also prone to similar thermal influences. In addition inhomogeneities and crystal defects further falsify the measurements.

In the face of these glaring flaws the MCA must have wrongly distributed a considerable amount of the incoming pulses, making an exact calibration impossible. The workings of the measurement software itself are of dubious nature. All the output text files were two channels short. It can be suspected that more problems may have occurred during the data output.

These sources of errors persisted for the entire experiment.

4.2 Compton-Background

The counting rate caused by the Compton background was determined to be:

$$\dot{N}_{Compton} = (13.47 \pm 0.11)\,\text{cps}$$

The counting rate has a relative error of under 1% and could therefore be subtracted from the measured counting rates for the Mößbauer spectra without hesitation.

4.3 Attenuation by the acrylic glass

The measured counting rates with and without the acrylic glass shielding were:

$$\dot{N}_0 = (66.9 \pm 0.3)\,\text{cps}$$
$$\dot{N}_{acryl} = (51.9 \pm 0.2)\,\text{cps}$$

So the acrylic glass in the casing of the absorber probe had an effect on the counting rate. The mass attenuation coefficient of the acrylic glass in the retainer of the probes was found (for 14.4 keV photons):

$$\mu = (1.3 \pm 0.3)\,\frac{1}{\mathrm{cm}}$$

With this value the theoretical value of the counting rate for the acrylic glass shielded scintillation counter could be calculated:

$$\dot{N}_{acryl,theo} = (51 \pm 3)\,\mathrm{cps}$$

This value is in accordance with the experimental observation. As the attenuation is independent of the absorber velocity it simply amounts to a constant shift of the counting rate and it is legitimately ignorable.

4.4 One line absorber

The one line absorption spectrum was fitted with a Voigt function from which the life time of the excited state (τ), the isomeric shift (E_{iso}) and the Debye-Waller factor of the source could be determined. The resulting values (with literature values) are summarised in table 11.

Result	value	literature
E_{iso}	(8.9 ± 0.2) neV	
T_A	6.2 ± 0.6	
τ	(150 ± 10) ns	141 ns
$T_{1/2}$	(104 ± 8) ns	98 ns
τ_{fit}	(83 ± 5) ns	141 ns
f_S	0.56 ± 0.02	

Table 11: Experimental values for the isomeric shift(E_{iso}), effective absorber thickness (T_A), lifetime (τ), half life ($T_{1/2}$) and the Debye-Waller factor of the source (f_S) with literature values

The value for the isomeric shift is realistic, but may have been falsified by the slight asymmetry of the curve.

The obtained value for the effective absorber thickness was used to find the lifetime of the excited state (τ). The final value is compatible with the literature value in a range of 1σ. Although it must be said that the uncertainty is rather high.

The lifetime was also extracted directly from the width of the fitted Voigt function (τ_{fit}). However the result contradicts the literature value as it is far too small. This means that the fitted function is far too wide. Along with the aforementioned poor quality of the used electronics the movement of the absorber is a considerable source of error. The velocity of the absorber was most certainly not constant and the step size of 0.01 $\frac{mm}{s}$ was overly optimistic. This can be seen in the scattering of the data points and the asymmetry of curve.

The Debye-Waller factor of the source did not have a literature value for comparison, but the value of 0.56 is adequate.

All in all the results of the one line spectrum are satisfactory, especially in face of the discussed shortcomings of the experimental set-up.

4.5 Six line absorber

The spectrum of the six line absorber was fitted with six Lorentz functions. The following values were extracted from the measured data:

	Value	literature
Isomeric shift: E_{iso}	(5.0 ± 0.5) neV	5.3 neV
	Value / μ_N	
nuclear magnetic moment $\mu_{e,ab}$	-0.153 ± 0.004	
nuclear magnetic moment $\mu_{e,ac}$	-0.159 ± 0.002	$(-0.1549 \pm 0.0002)\,\mu_N$
nuclear magnetic moment $\mu_{e,bc}$	-0.159 ± 0.003	
	Value / T	
magnetic field B_a	33.1 ± 1.9	
magnetic field B_b	32.7 ± 0.6	≈ 33 T
magnetic field B_c	32.6 ± 2.7	

Table 12: Experimental values for the isomeric shift,magnetic momentum and magnetic field at the position of the nucleus with literature values

The experimentally determined isometric shift is in accordance with the literature value. However the relative uncertainty of 10% impinges the validity of the data.

All values of for the magnetic moments and magnetic field at the position of the nucleus match the literature value in a range of 1σ (with one exception of 2σ).

In conclusion the experimental results for the six line spectrum are adequate, providing insight on the hyperfine structure of ^{57}Fe.

5 Appendix

5.1 Measured data

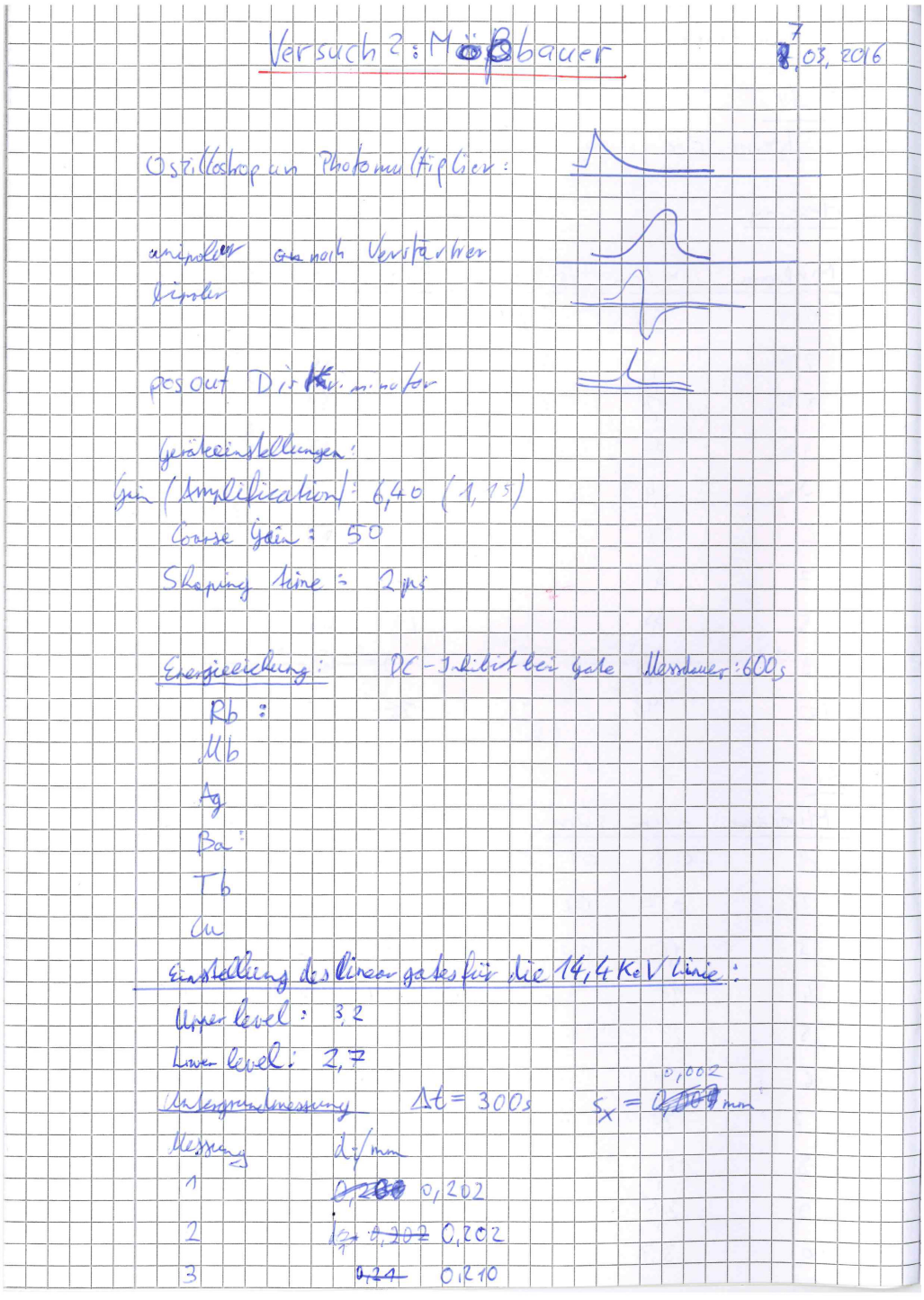


Figure 5.1: Laboratory journal entry page 1

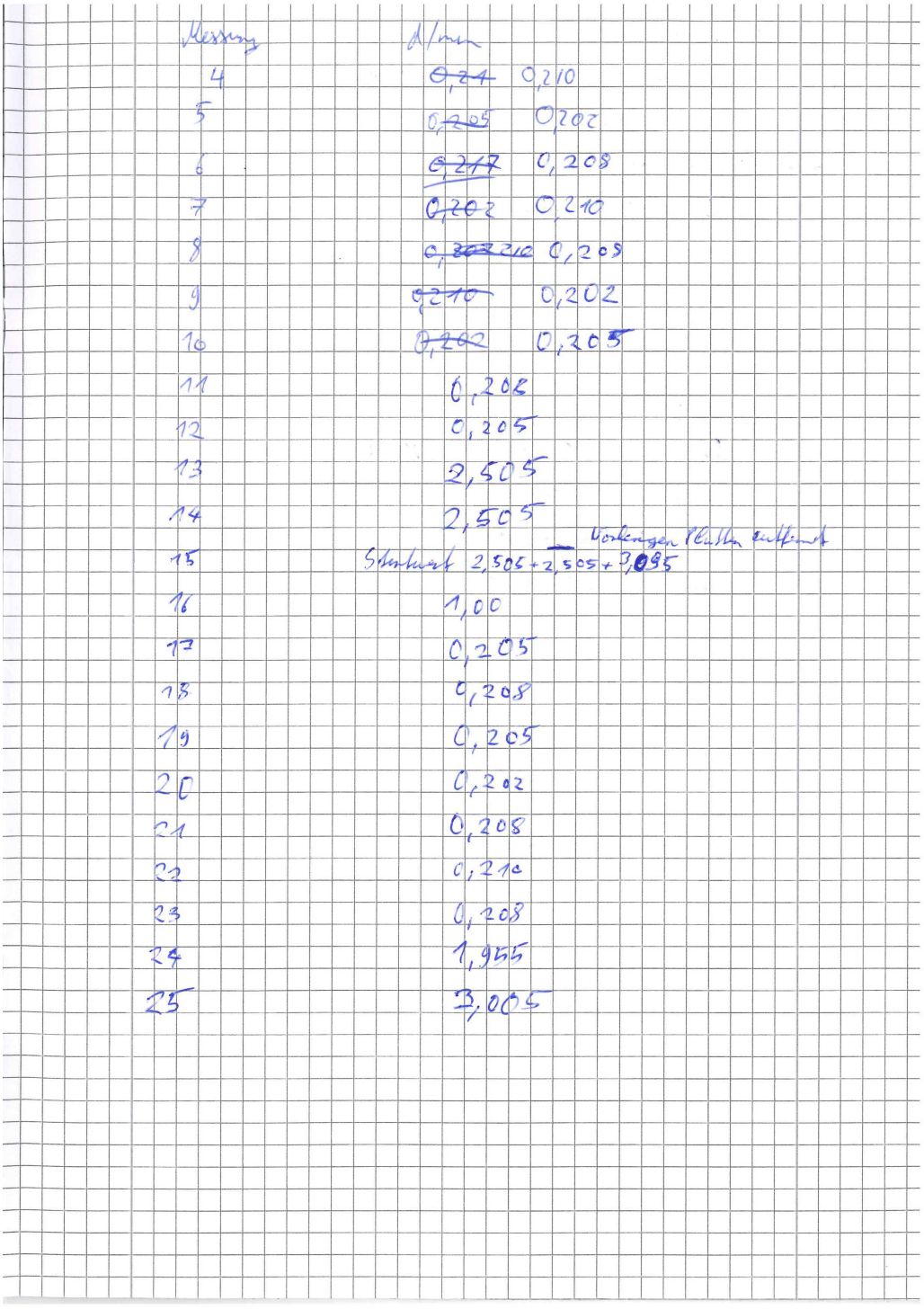


Figure 5.2: Laboratory journal entry page 2

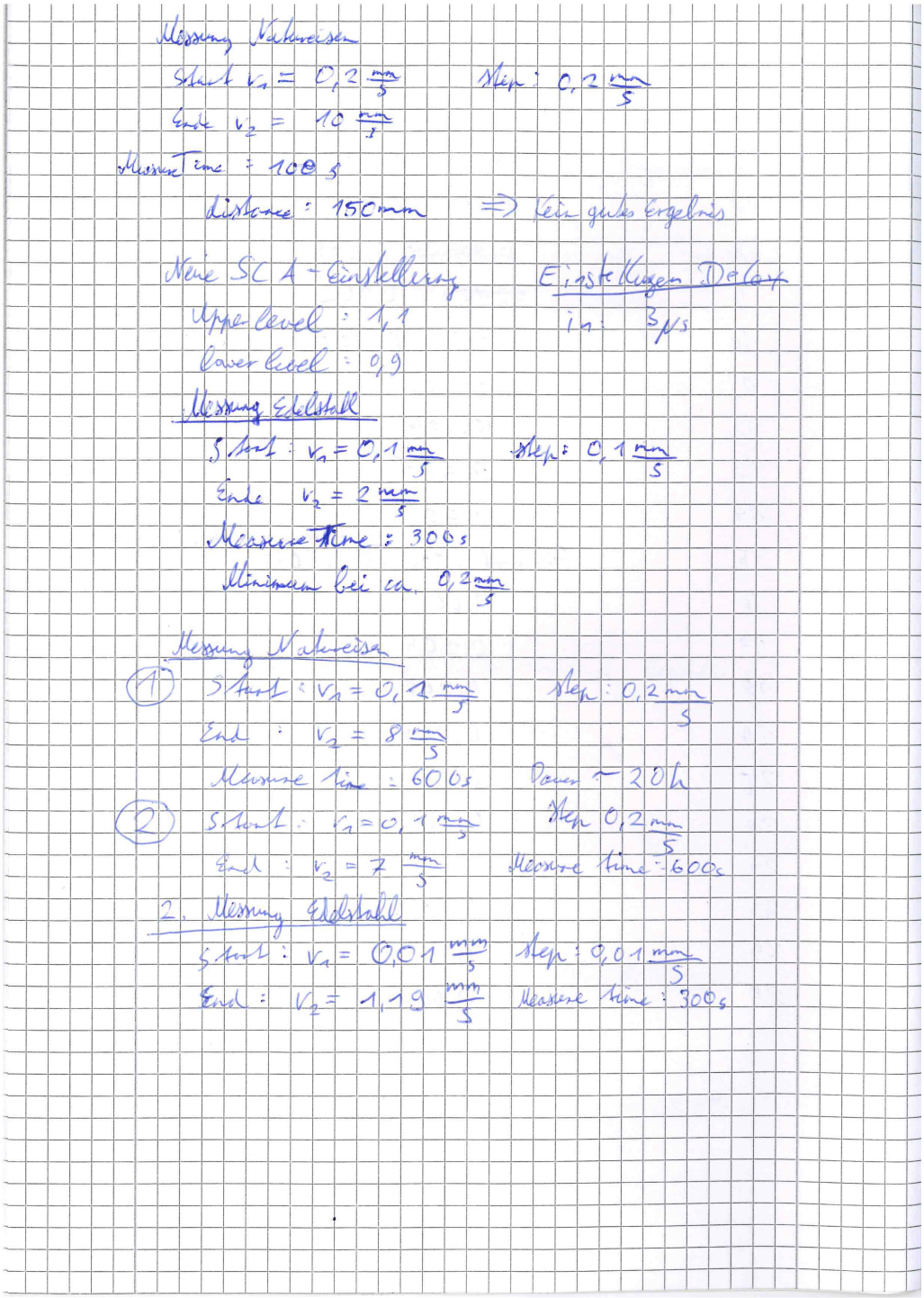
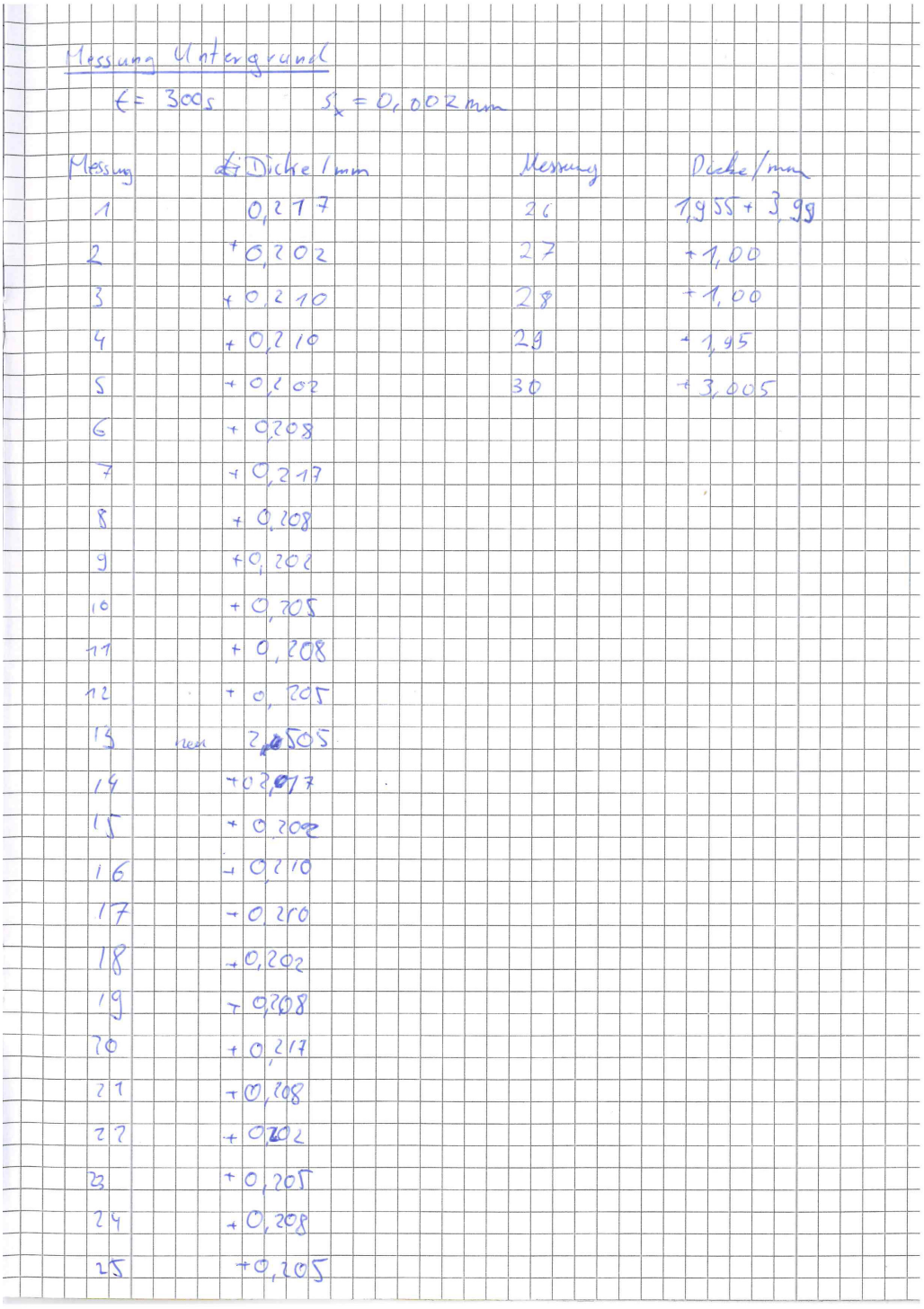


Figure 5.3: Laboratory journal entry page 3



Messung Untergrund

t = 300s s_x = 0,002 mm

Messung	Dicke / mm	Messung	Dicke / mm
1	0,217	26	1,955 + 3,99
2	+ 0,202	27	+ 1,00
3	+ 0,210	28	+ 1,00
4	+ 0,210	29	+ 1,95
5	+ 0,202	30	+ 3,005
6	+ 0,208		
7	+ 0,217		
8	+ 0,208		
9	+ 0,202		
10	+ 0,205		
11	+ 0,208		
12	+ 0,205		
13	neu 2,0505		
14	+ 0,217		
15	+ 0,202		
16	+ 0,210		
17	+ 0,210		
18	+ 0,202		
19	+ 0,208		
20	+ 0,217		
21	+ 0,208		
22	+ 0,202		
23	+ 0,205		
24	+ 0,208		
25	+ 0,205		

Figure 5.4: Laboratory journal entry page 4

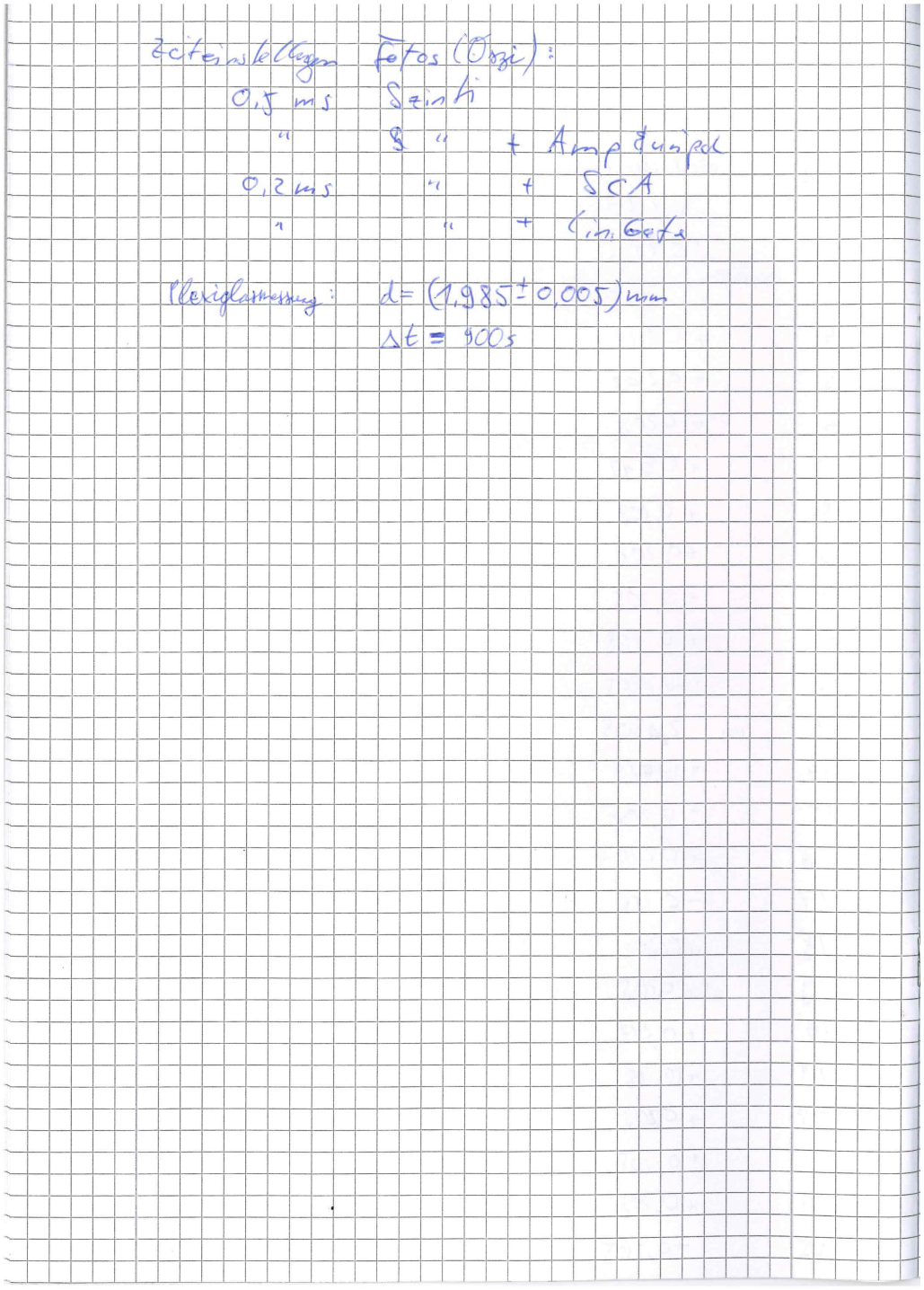


Figure 5.5: Laboratory journal entry page 5

5.2 R source code

5.2.1 Calibration

```
graphics.off()
path = dirname(sys.frame(1)$ofile)


round2 = function(val, numb){
       len = max(nchar(floor(val)))
       return(formatC( round( val, numb ), format='f', digits=numb, width = numb +
           1 + len))
}

plusminus = function(x){
       if(sign(x) == -1){
              return("-")
       } else {
              return("+")
       }
}


library(minpack.lm)
library(Hmisc)
library(pracma)

ranges = list(
       "Barium" = c(100, 450, 32.06, 36.55),
       "Kupfer" = c(100, 400, 8.04, 8.91),
       "Molybdaen" = c(100, 250, 17.44, 19.63),
       "Rubidium" = c(100, 200, 13.37, 14.97),
       "Silber" = c(100, 400, 22.10, 24.99),
       "Terbium" = c(250, 800, 44.23, 50.65)
)

data_path = paste(dirname(path), "Daten", "Energieeichung", sep = "/")

files = dir(data_path, pattern = ".TKA")


  mus  = c()
s_mus  = c()
solls = c()
labs = c()
names = c()
sigmafaktor = .5

i = 0
for(file_ in files[1:6]){
```

```
46        i = i+1
47        counts = unlist(read.csv(paste(data_path, file_, sep = "/"), skip = 2, as.is
              = TRUE))
48        name = sub(".TKA", "", file_)
49
50        range_ = ranges[[name]][1:2]
51        range_ = min(range_):max(range_)
52        soll = ranges[[name]][3]
53        legendaddon = c()
54
55        cat("\n",name, sep = "")
56        flush.console()
57
58
59        png(paste(data_path, "/",name, ".png", sep = ""), width = 6, height = 4,
              units = "in", res = 600)
60        par("mar" = c(4, 4, 1, 1))
61        par("oma" = c(0, 0, 0, 0))
62
63        plot(counts, type = "l", xlab = "Channel", ylab = "counts / 1000", axes =
              FALSE)
64        box()
65        axis(1, at = (0:20)*128, )
66        axis(2, at = pretty(counts), lab = pretty(counts) / 1000, las = 2)
67
68        dat = data.frame(range_, counts[range_], sqrt(counts[range_]))
69        names(dat) = c("x", "y", "s_y")
70        points(dat[1:2], type = "l", col = "red")
71        abline(h = axTicks(2), v = (0:20)*128, col = "#B0B0B0")
72
73        two = 2
74        cat("\nstarte Fit")
75        if(i == 6){
76               # next()
77               fit = nlsLM(
78                      y ~ A * exp(-(x - mu1)^2/(two * sigma1^two)) + B * exp(-(x -
                           mu2)^2/(two * sigma2^two)) + C + D * x,
79                      data = dat,
80                      start = list(
81                             A  = diff(range(dat$y)),
82                             mu1  = min(range_) + mean(range_)/3,
83                             sigma1 = 10,
84                             B  = diff(range(dat$y))* .7,
85                             mu2  = min(range_) + mean(range_)/3*2,
86                             sigma2 = 5,
87                             C  = 0,
88                             D  = 0
89                      weights = 1 / (dat$s_y^2),
90                      control = list(maxiter = 500)
```

```
91              )
92
93              A  = summary(fit)[[10]][1,1]
94              mu1  = summary(fit)[[10]][2,1]
95              sigma1 = summary(fit)[[10]][3,1]
96              B  = summary(fit)[[10]][4,1]
97              mu2  = summary(fit)[[10]][5,1]
98              sigma2 = summary(fit)[[10]][6,1]
99              C  = summary(fit)[[10]][7,1]
100             D  = summary(fit)[[10]][8,1]
101
102
103             s_A  = summary(fit)[[10]][1,2]
104             s_mu1 = summary(fit)[[10]][2,2]
105             s_sigma1= summary(fit)[[10]][3,2]
106             s_B  = summary(fit)[[10]][4,2]
107             s_mu2 = summary(fit)[[10]][5,2]
108             s_sigma2= summary(fit)[[10]][6,2]
109             s_C  = summary(fit)[[10]][7,2]
110             s_D  = summary(fit)[[10]][8,2]
111
112             legendaddon = c(
113                     expression(Fit: A %.% e^(-(x - mu[1])^2/{2 %.% sigma[1]^2}) +
                            B %.% e^(-(x - mu[2])^2/{2 %.% sigma[2]^2}) + C + D %.% x),
114                     as.expression(bquote(A: ~.(round2(A, 2)) * scriptstyle("" %+-%
                            .(round2( s_A, 2))))),
115                     as.expression(bquote(mu[1]: ~.(round2(mu1, 2)) * scriptstyle("
                            " %+-% .(round2(s_mu1, 2))))),
116                     as.expression(bquote(sigma[1]: ~.(round2(sigma1, 2)) *
                            scriptstyle("" %+-% .(round2(s_sigma1, 2))))),
117                     # as.expression(bquote(B: ~.(round2(B, 2)) * scriptstyle(""
                            %+-% .(round2( s_B, 2))))),
118                     # as.expression(bquote(mu[2]: ~.(round2(mu2, 2)) * scriptstyle
                            ("" %+-% .(round2(s_mu2, 2))))),
119                     # as.expression(bquote(sigma[2]: ~.(round2(sigma2, 2)) *
                            scriptstyle("" %+-% .(round2(s_sigma2, 2))))),
120                     as.expression(bquote(C: ~.(round2(C, 2)) * scriptstyle("" %+-%
                            .(round2(s_C, 2))))),
121                     as.expression(bquote(D: ~.(round2(D, 2)) * scriptstyle("" %+-%
                            .(round2(s_D, 2)))))
122
123
124
125             )
126
127
128             labs = c(labs, name, name)
129
130             funct = function(x){
```

```
131                     return(A * exp(-(x - mu1)^2/(two * sigma1^two)) + C + D * x)
132             }
133
134             funct2 = function(x){
135                     return(B * exp(-(x - mu2)^2/(two * sigma2^two)) + C + D * x)
136             }
137             # curve(funct1, add = TRUE, col = "#00B0B0", from = min(range_), to =
                    max(range_))
138             # curve(funct2, add = TRUE, col = "#00B0B0", from = min(range_), to =
                    max(range_))
139
140
141             # funct = function(x){
142                     # return(A * exp(-(x - mu1)^2/(two * sigma1^two)) + B * exp(-(
                        x - mu2)^2/(two * sigma2^two)) + C + D * x)
143             # }
144
145             # names = c(names, name)
146             # mus  = c(mus, mu2)
147
148             # s_mus = c(s_mus, sigma2 * sigmafaktor)
149             # solls = c(solls, ranges[[name]][4])
150
151
152             # arrows(mu2 - sigma2 * sigmafaktor, funct(mu2), mu2 + sigma2 *
                    sigmafaktor, funct(mu2), code = 3, angle = 90, length = 0.0275,
                    col = "#B0B000")
153
154
155     } else {
156             fit = nlsLM(
157                     y ~ A * exp(-(x - mu1)^2/(2 * sigma1^2)) + C + D * x,
158                     data = dat,
159                     start = list(
160                             A  = diff(range(dat$y)),
161                             mu1  = mean(range_),
162
163                             sigma1 = diff(range(range_))/4,
164
165                             C  = 0,
166                             D  = 0
167                     ),
168                     weights = 1 / (dat$s_y^2),
169                     control = list(maxiter = 500)
170             )
171
172             funct = function(x){
173                     return(A * exp(-(x - mu1)^2/(2 * sigma1^2)) + C + D * x)
174             }
```

```
175             labs = c(labs, name)
176
177               A  = summary(fit)[[10]][1,1]
178               C  = summary(fit)[[10]][4,1]
179               D  = summary(fit)[[10]][5,1]
180               mu1 = summary(fit)[[10]][2,1]
181               sigma1= summary(fit)[[10]][3,1]
182
183
184             s_A  = summary(fit)[[10]][1,2]
185             s_C  = summary(fit)[[10]][4,2]
186             s_D  = summary(fit)[[10]][5,2]
187             s_mu1 = summary(fit)[[10]][2,2]
188             s_sigma1= summary(fit)[[10]][3,2]
189
190
191             legendaddon = c(
192                     expression(Fit: A %.% e^(-(x - mu)^2/{2 %.% sigma^2} ) + C + D
                            %.% x),
193                     as.expression(bquote(mu: ~.(round2(mu1, 2)) * scriptstyle(""
                           %+-% .(round2( s_mu1, 2))))),
194                     as.expression(bquote(sigma: ~.(round2(sigma1, 2)) *
                           scriptstyle("" %+-% .(round2(s_sigma1, 2))))),
195                     as.expression(bquote(A: ~.(round2(A, 2)) * scriptstyle("" %+-%
                            .(round2( s_A, 2))))),
196                     as.expression(bquote(C: ~.(round2(C, 2)) * scriptstyle("" %+-%
                            .(round2( s_C, 2))))),
197                     as.expression(bquote(D: ~.(round2(D, 2)) * scriptstyle("" %+-%
                            .(round2( s_D, 2))))
198             )
199     }
200     cat("\nFit beendet")
201     flush.console()
202
203     A  = summary(fit)[[10]][1,1]
204     mu1  = summary(fit)[[10]][2,1]
205     sigma1 = summary(fit)[[10]][3,1]
206
207
208     s_A  = summary(fit)[[10]][1,2]
209     s_mu1 = summary(fit)[[10]][2,2]
210     s_sigma1= summary(fit)[[10]][3,2]
211
212     ranges[[name]][5] = mu1
213
214     funct1 = function(x){
215             return(A * exp(-(x - mu1)^2/sigma1) + C + D * x)
216     }
217
```

```
218         chi2 = summary(fit)$sigma
219 
220         if(name != "Kupfer" ){
221                 mus  = c(mus, mu1)
222                 s_mus = c(s_mus, sigma1 * sigmafaktor)
223                 solls = c(solls, soll)
224         }
225 
226         # curve(funct1, add = TRUE, col = "#00B000", from = min(range_), to = max(
                range_))
227         curve(funct, add = TRUE, col = "#00B000", from = min(range_), to = max(range
                _))
228         # arrows(dat$x, funct1(dat$x) - dat$s_y, dat$x, funct1(dat$x) + dat$s_y,
                code = 3, angle = 90, length = 0.0175, col = "#0000B0")
229         arrows(min(dat$x), 0, max(dat$x), 0, code = 3, angle = 90, length = 0.0275,
                col = "#B00000")
230 
231         arrows(mu1 - sigma1 * sigmafaktor, funct(mu1), mu1 + sigma1 * sigmafaktor,
                funct(mu1), code = 3, angle = 90, length = 0.0275, col = "#B0B000")
232 
233 
234 
235         legend("topright",
236                 c(
237                         "Counts",
238                         "Fitted area",
239                         legendaddon,
240                         as.expression(bquote(chi[red]^2 == .(round2(chi2,2))))
241                 ),
242                 col = c("black", "#B00000", "#00B000"),
243                 lty = c(1,1,1,0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
244                 pt.cex = .5,
245                 cex = .75,
246                 bg = "white"
247         )
248 
249         dev.off()
250 }
251 
252 
253 
254 png(paste(data_path, "Energieeichung.png", sep = "/"), width = 6, height = 4, units
        = "in", res = 600)
255 par("mar" = c(4, 4, 1, 1))
256 par("oma" = c(0, 0, 0, 0))
257 
258 plot(mus, solls, xlab = "Channel", ylab = "Energy / keV", xlim = range(mus - s_mus,
        mus + s_mus), las = 1, pch = 20)
```

```
259 arrows(mus - s_mus, solls, mus + s_mus, solls, code = 3, angle = 90, length =
        0.0175, col = "#0000B0")
260 # text(solls, mus, labs, cex = .5, pos = 4)
261
262 dat2 = data.frame(mus, solls)
263 names(dat2) = c("x", "y")
264
265 fit2 = nlsLM(
266         y ~ A * x + B,
267         data = dat2,
268         start = list(
269                 A  = 10,
270                 B  = 15
271         ),
272         weights = 1/(s_mus^2)
273 )
274 fit2_coef = summary(fit2)$coef
275
276 ener = function(E){
277         return(coef(fit2)[[1]] * E + coef(fit2)[[2]])
278 }
279 chi2b = sum( (solls - ener(mus) )^2 / (s_mus * fit2_coef[1,1])^2 )
280
281 curve(ener, add = TRUE, col = "#00B000")
282
283 abline(h = axTicks(2), v = axTicks(1), col = "#B0B0B0")
284
285 legend("topleft",
286         c(
287                 "Experimental data",
288                 "Errors",
289                 "Fit",
290                 as.expression(bquote(chi[red]^2: ~ .(signif(chi2b, 4)))),
291                 as.expression(bquote(
292                         Energy(ch) == .(round2(fit2_coef[1,1], 2)) * scriptstyle("" %+-% .(round2(fit2_coef[1,2], 2))) %.% ch
293                         - .(round2(abs(fit2_coef[2,1]), 2)) * scriptstyle("" %+-% .(round2(fit2_coef[2,2], 2)))
294                 ))
295         ),
296         lty = c(
297                 0,
298                 0,
299                 1,
300                 0,
301                 rep(0, 6)
302         ),
303         pch = c(
304                 20,
```

```
305                 15,
306                 NA,
307                 NA,
308                 rep(NA, 6)
309         ),
310         col = c(
311                 "black",
312                 "#0000B0",
313                 "#00B000",
314                 "black",
315                 rep("black", 6)
316         ),
317         pt.cex = .5,
318         cex = .75,
319         bg = "white"
320 )
321
322 dev.off()
323
324
325
326
327
328
329 cat("\n")
```

5.2.2 Background

```
1  graphics.off()
2  path = dirname(sys.frame(1)$ofile)
3
4  library(minpack.lm)
5  library(Hmisc)
6  library(pracma)
7
8
9  round2 = function(val, numb){
10         len = max(nchar(floor(val)))
11         return(formatC( round( val, numb ), format='f', digits=numb, width = numb +
               1 + len))
12 }
13
14 untergrund = "Untergrund2"
15 data_path = paste(dirname(path), "Daten", untergrund, sep = "/")
16 exportpath= dirname(path)
17
18 files = dir(data_path, pattern = ".TKA")
19
20
```

```
21 s_dicken = .002
22 if(untergrund == "Untergrund1"){
23         dicke = c(
24                 cumsum(c(.202, .202, .210, .210, .202, .208, .210, .205, .202, .205,
                       .208, .205, 2.505, 2.505)),
25                 cumsum(c(8.105, 1.00, .205, .208, .205, .202, .208, .210, .208,
                       1.955, 3.095))
26         )
27 } else if(untergrund == "Untergrund2"){
28         dicke = c(
29                 cumsum(c( 217, 202, 210, 210, 202, 208, 217, 208, 202, 205, 208, 205)
                       ),
30                 cumsum(c(2505, 217, 202, 210, 210, 202, 208, 217, 208, 202, 205, 208,
                        205)),
31                 cumsum(c(1955 + 3990, 1000, 1000, 1950, 3005))
32         )/1000
33         s_dicke = (sqrt(c(1:12, 1:13, 1:5)) * .002 )
34 }
35   durrs   = c()
36   count_sum  = c()
37   counts_sum = c()
38 s_counts_sum = c()
39 i = 0
40
41 filenumbers = gsub("Messung(\\d+)\\.TKA", "\\1", files)
42 files = files[order(as.numeric(filenumbers))]
43
44 for(file_ in files){
45         i=i+1
46         durr = unlist(read.csv(paste(data_path, file_, sep = "/"), skip = 1, nrows =
                1, as.is = TRUE, head = FALSE))
47         durrs = c(durrs, durr)
48
49         counts = unlist(read.csv(paste(data_path, file_, sep = "/"), skip = 2, nrows
                = 500, as.is = TRUE, head = FALSE))
50         count_sum  = c(count_sum, sum(counts))
51         counts = sum(counts)/durr
52         counts_sum = c(counts_sum, counts)
53         s_counts_sum = sqrt(counts_sum / durrs)
54
55
56 }
57
58 dat   = data.frame(dicke, counts_sum, s_counts_sum)
59 names(dat) = c("x", "y", "s_y")
60
61 png(paste(exportpath, "Untergrund.png", sep = "/"), width = 6, height = 4, units = "
       in", res = 600)
62 par("mar" = c(4, 5, 1, 1))
```

```
63 par("oma" = c(0, 0, 0, 0))
64
65 plot(dicke, counts_sum, xlab = "d / mm", ylab = expression(dot(N) *" / "* s^-1),
       xlim = range(0, dicke), ylim = range(0, counts_sum - s_counts_sum, counts_sum +
       s_counts_sum), pch = 20, cex = .5)
66 abline(v = axTicks(1), h = axTicks(2), col = "#888888")
67 arrows(dicke, counts_sum - s_counts_sum, dicke, counts_sum + s_counts_sum, code = 3,
        angle = 90, length = 0.0175, col = "#0000B0")
68 arrows(dicke - s_dicke, counts_sum, dicke + s_dicke, counts_sum, code = 3, angle =
       90, length = 0.0175, col = "#0000B0")
69
70
71
72 # Startparameter
73 A = -1
74 a = 1
75 B = -1
76 b = 1
77
78
79 # fitcurve = function(x){
80         # return( A * exp(a * x) + B * exp(b * x))
81 # }
82 # curve(fitcurve, add = TRUE, col = "#0000B0", lty = 2, lwd = 5)
83
84 fit = nlsLM(
85         y ~ A * exp(a * x) + B * exp(b * x),
86         data = dat,
87         start = list(
88                 A  = A,
89                 a  = a,
90                 B  = B,
91                 b  = b
92
93         )
94 )
95
96 coef = summary(fit)[[10]]
97   A  = coef[1,1]
98   a  = coef[2,1]
99   B  = coef[3,1]
100  b  = coef[4,1]
101 s_A  = coef[1,2]
102 s_a  = coef[2,2]
103 s_B  = coef[3,2]
104 s_b  = coef[4,2]
105
106
107 fitcurve_counts = function(x){
```

```
108         return( B * exp(b * x))
109 }
110 fitcurve_undergound = function(x){
111         return( A * exp(a * x))
112 }
113
114 s_undergound = function(x){
115         return(
116                 sqrt(
117                         (exp(a * x) * s_A)^2
118                         +
119                         (A * exp(a * x) * x * s_a)^2
120                 )
121         )
122 }
123
124 fitcurve = function(x){
125         return( A * exp(a * x) + B * exp(b * x))
126 }
127
128 chi2  = sum((fitcurve(dat$x) - dat$y)^2 / (dat$y))/df.residual(fit)
129
130 curve(fitcurve, add = TRUE, col = "#00B0B0", from = par("usr")[1], to = par("usr")
      [2])
131 curve(fitcurve_counts, add = TRUE, col = "#00B000", from = par("usr")[1], to = par("
      usr")[2])
132 curve(fitcurve_undergound, add = TRUE, col = "#B00000", from = par("usr")[1], to =
      par("usr")[2])
133
134   rate_unterground0 = fitcurve_undergound(0)
135 s_rate_unterground0 = s_undergound(0)
136
137 legend("topright",
138         c(
139                 "Experimental data",
140                 "Errors",
141                 expression(Fit: A%.%e^(a*x)+B%.%e^(b*x)),
142                 as.expression(bquote(chi[red]^2: ~ .(signif(chi2, 4)))),
143                 expression(A%.%e^{a*d}),
144                 expression(B%.%e^{b*d}),
145                 expression(B%.%e^{b*(d == 0)} ==""),
146                 as.expression(bquote((.(round(rate_unterground0, 2))
147                         *scriptstyle("" %+-% .(round(s_rate_unterground0, 2))) )*s^-1)
                            )
148                 ),
149         col = c(
150                 "Black",
151                 "#0000B0",
152                 "#00B0B0",
```

```
                "Black",
                "#00B000",
                "#B00000",
                "Black",
                "Black"
                ),
        pch = c(
                20,
                15,
                NA,
                NA,
                NA,
                NA,
                NA,
                NA
        )
        , lty = c(
                0,
                0,
                1,
                0,
                1,
                1,
                0,
                0
        ),
        pt.cex = .5,
        cex = .75
)









dev.off()
```

5.2.3 Test of calibration

```
graphics.off()
path = dirname(sys.frame(1)$ofile)

library(minpack.lm)
library(Hmisc)
library(pracma)

data_path = paste(dirname(path), "Daten", "sonstige_Messungen", sep = "/")
exportpath= dirname(path)

files = dir(data_path, pattern = ".TKA")
round2 = function(val, numb){
        len = max(nchar(floor(val)))
        return(formatC( round( val, numb ), format='f', digits=numb, width = numb +
            1 + len))
}


# data obtained by MCA Calibration
a = 9.647
b = 13.642
EofChan = function(Chan){
        # chan = a * E + b
        return((Chan - b)/a)
}
chanofE = function(E){
        return(a * E + b)
}


for(file_ in files){
        counts = unlist(read.csv(paste(data_path, file_, sep = "/"), skip = 2, as.is
             = TRUE))
        x = EofChan(1:length(counts))
        y = counts
        dat = data.frame(x, y)[intersect(which(x < 22), which(x > 8)),]
        helper = range(dat$y)

        png(paste(exportpath, sub(".TKA", ".png", file_), sep = "/"), width = 6,
            height = 4, units = "in", res = 600)
        par("mar" = c(4, 5, 1, 1))
        par("oma" = c(0, 0, 0, 0))

        plot(dat, type = "l", xlab = "E / keV", ylab = "counts", las = 1)
        abline(v = ((min(axTicks(1))):(max(axTicks(1))*20))/2, col = "#888888", lty
            = 3)
        abline(v = axTicks(1), h = axTicks(2), col = "#888888")

        #starting parameter
```

```
46        A  = 3800
47        C  = 330
48        M  = 14.4
49        sigma = 2
50        # print(A)
51        # print(C)
52        # print(M)
53        # print(Gamma)
54        # print(sigma)
55
56        peak = function(x){
57               return(A * Gauss(x, M, sigma) + C)
58        }
59        # curve(peak, add = TRUE, col = "#B00000", lty = 2)
60
61        fit = nlsLM(
62               y ~ A * Gauss(x, M, sigma) + C,
63               data = dat,
64               start = list(
65                      A  = A,
66                      C  = C,
67                      M  = M,
68                      sigma = sigma
69               )
70        )
71        coeff = summary(fit)[[10]]
72        A  = coeff[1,1]
73        C  = coeff[2,1]
74        M  = coeff[3,1]
75        sigma = coeff[4,1]
76
77        s_A  = coeff[1,2]
78        s_C  = coeff[2,2]
79        s_M  = coeff[3,2]
80        s_sigma = coeff[4,2]
81
82
83
84        curve(peak, add = TRUE, col = "#00B000", lwd = 2, from = par("usr")[1], to =
              par("usr")[2])
85        chi2 = sum((peak(dat$x) - dat$y)^2 / dat$y)/df.residual(fit)
86
87        abline(v = M, col = "#0000B0", lwd = 2)
88        abline(v = 14.4, col = "#B00000", lwd = 2)
89        arrows(M - sigma/2, 400, M + sigma/2, 400, code = 3, angle = 90, length =
             0.0375, col = "#0000B0", lwd = 2)
90        text(M, 300, as.expression(bquote(.(round2(M, 2)) %+-% .(round2(sigma/2, 2))
             )), col = "#0000B0", pos = 4)
91        text(14.4, 300, 14.4, col = "#B00000", pos = 2)
```

```
92      box()
93      legend("topright",
94              c(
95                      "Experimental data",
96                      expression(Gaussfit),
97                      as.expression(bquote(chi[red]^2: ~ .(signif(chi2, 4)))),
98                      as.expression(
99                              bquote(
100                                     mu == (.(round2(M, 2)) * scriptstyle("" %+-% .(
                                            round2(s_M, 2)) )) * keV
101                             )
102                     ),
103                     "Maximum",
104                     "Theoretical value"
105             ),
106             col = c(
107                     "Black",
108                     "#00B000",
109                     "Black",
110                     "Black",
111                     "#0000B0",
112                     "#B00000",
113                     "Black"
114                     ),
115             pch = c(
116                     NA,
117                     NA,
118                     NA,
119                     NA,
120                     NA,
121                     NA,
122                     NA
123             )
124             , lty = c(
125                     1,
126                     1,
127                     0,
128                     0,
129                     1,
130                     1,
131                     0,
132                     0
133             ),
134             pt.cex = .5,
135             cex = .75,
136             bg = "white"
137     )
138
139     dev.off()
```

```
}
```

5.2.4 Attenuation from plastic

```
graphics.off()
path = dirname(sys.frame(1)$ofile)
final = TRUE # trigger for creating a png ( if == TRUE) or preview (if == FALSE)

library(minpack.lm)
library(Hmisc)
library(pracma)
library(RcppFaddeeva)

round2 = function(val, numb){
       len = max(nchar(floor(val)))
       return(formatC( round( val, numb ), format='f', digits=numb, width = numb +
           1 + len))
}


  U  = 13.47 #untergrund
s_U  = .11

data_path = paste(dirname(path), "Daten", "Plastik", sep = "/")
exportpath= dirname(path)

files = dir(data_path, pattern = ".TKA")

  rates = c()
s_rates = c()
  names_= c()

for( file_ in files ){
       durr  = unlist(read.csv(paste(data_path, file_, sep = "/"), skip = 1, nrows
           = 1, as.is = TRUE, head = FALSE))
       data_temp = unlist(read.csv(paste(data_path, file_, sep = "/"), skip = 2, as
           .is = TRUE, head = FALSE))
       rate  = sum(data_temp) / durr
       rates  = c(rates, rate)
       s_rates  = c(s_rates, sqrt(rate / durr))
       names_  = c(names_ , sub(".TKA", "", file_))
}

print( names_)
print(round( rates, 1))
print(round(s_rates, 1))

Es = c(8, 10, 15, 20)
mus = c(6.49, 3.357, 1.101, 0.5714)
```

```
43
44 png(paste(exportpath, "Blaschdig.png", sep = "/"), width = 6, height = 4, units = "
       in", res = 600)
45 par("mar" = c(4, 5.75, 1, 1))
46 par("oma" = c(0, 0, 0, 0))
47
48 plot(
49        Es,
50        mus,
51        xlab  = expression(E / keV),
52        ylab  = expression(mu / rho * " / " * frac(cm^2, g)),
53        pch   = 20
54 )
55 abline(v = axTicks(1), h = axTicks(2), col = "#888888")
56
57 A = 110
58 B = - .4
59 C = .5
60 fitcurve = function(x){
61        return(A * exp(B * x) + C)
62 }
63 # curve(fitcurve, add = TRUE, col = "#B00000", lty = 2, lwd = 2)
64
65 dat = data.frame(Es, mus)
66 names(dat) = c("x", "y")
67 fit = nlsLM(
68        y ~ A * exp(B * x) + C,
69        dat = dat,
70        start = list(
71               A = A,
72               B = B,
73               C = C
74        )
75
76 )
77 coeff = summary(fit)[[10]]
78   A = coeff[1,1]
79   B = coeff[2,1]
80   C = coeff[3,1]
81
82 s_A = coeff[1,2]
83 s_B = coeff[2,2]
84 s_C = coeff[3,2]
85
86 s_fitcurve = function(x){
87        return(
88               sqrt(
89                        (s_A * (exp(B * x)) )^2
90                      + (s_B * (A * x * exp(B * x)))^2
```

```
                        + (s_C)^2
                )
        )
}

  mu = fitcurve(14.4) * 1.19
s_mu = s_fitcurve(14.4) * 1.19

  dicke = 1.985e-1
s_dicke = 0.005e-1

  abschw= rates[2] * exp(-mu * dicke)
s_abschw= sqrt(
          (s_rates[2] * exp(-mu * dicke) )^2
        + (rates[2] * exp(-mu * dicke) * dicke * s_mu)^2
        + (rates[2] * exp(-mu * dicke) * mu * s_dicke)^2
        )



curve(fitcurve, add = TRUE, col = "#00B000", lwd = 2, from = par("usr")[1], to = par
    ("usr")[2])
cat("\nnu/rho:\t", fitcurve(14.4), "\nnu:\t", mu , " +- ", s_mu, "\nabschw:\t",
    abschw, " +- ", s_abschw, sep ="")
legend(
        "topright",
        c(
                "Datapoints",
                expression(Fit: A %.% e^{B %.% x} + C),
                as.expression(bquote(A == .(round(A, 0)) %+-% .(round(s_A, 0)))),
                as.expression(bquote(B == .(round(B, 2)) %+-% .(round(s_B, 2)))),
                as.expression(bquote(C == .(round(C, 2)) %+-% .(round(s_C, 2))))
                 # ,as.expression(bquote(chi^2 == frac(x,sum(0[i]^2, i, 4))))
        ),
        lty  = c( 0, 1, 0, 0, 0, 0),
        pch  = c(20, NA, NA, NA, NA, NA), #Batman
        col  = c("black", "#00B000", "black", "black", "black", "black"),
        pt.cex = .5,
        cex  = .75,
        bg  = "white"
)

dev.off()
```

5.2.5 Steel

```
graphics.off()
path = dirname(sys.frame(1)$ofile)
final = TRUE # trigger for creating a png ( if == TRUE) or preview (if == FALSE)
```

```
4
5  library(minpack.lm)
6  library(Hmisc)
7  library(pracma)
8  library(RcppFaddeeva)
9
10
11 round2 = function(val, numb){
12         len = max(nchar(floor(val)))
13         return(formatC( round( val, numb ), format='f', digits=numb, width = numb +
               1 + len))
14 }
15
16
17   U  = 13.47 #untergrund
18 s_U  = .11
19
20
21 data_path = paste(dirname(path), "Daten", "Stahl", sep = "/")
22 exportpath= dirname(path)
23
24 files = dir(data_path, pattern = ".csv")
25 data_ = data.frame(speed = double(), dauer = double(), counts = integer() )
26
27
28 for(file_ in files){
29
30         data_temp = read.csv(paste(data_path, file_, sep = "/" ) , dec = ",", sep =
               "\t", as.is = TRUE, header = FALSE)
31         data_  = rbind(data_, data_temp)
32
33 }
34
35 names(data_) = c("speed", "dauer", "counts")
36
37 # erstmal normal rechnen
38 data_$rate  = data_$counts / data_$dauer * 1000
39 data_$s_rate = sqrt(data_$rate / data_$dauer)
40
41 #Untergrund abziehen
42 data_$rate  = data_$rate - U
43 data_$s_rate = sqrt(data_$s_rate^2 + s_U^2)
44 data_$s_rate_  = data_$s_rate * 1
45
46 if(final == TRUE){
47         png(paste(exportpath, "Stahlfit.png", sep = "/"), width = 6, height = 4,
               units = "in", res = 600)
48         par("mar" = c(4, 5, 1, 1))
49         par("oma" = c(0, 0, 0, 0))
```

```
50 }
51
52
53 plot(data_$speed, data_$rate, cex = .5, pch = 20, xlab = expression(v * " / " * over
       (mm, s) ) , ylab = expression(dot(N) *" / "* s^-1))
54 arrows(data_$speed, data_$rate - data_$s_rate_, data_$speed, data_$rate + data_$s_
       rate_, code = 3, angle = 90, length = 0.0175, col = "#0000B0")
55
56 # abline(v = (-30:30)/10, h = axTicks(2), col = "#888888", lwd = .5)
57 abline(v = ((min(axTicks(1))*15):(max(axTicks(1))*15))/10, col = "#888888", lty = 3)
58 abline(v = axTicks(1), h = axTicks(2), col = "#888888")
59
60 helper = range(data_$rate)
61   x = data_$speed
62   y = data_$rate
63 s_y = data_$s_rate
64 dat = data.frame(x, y, s_y)
65
66
67 # Startparameter für fit
68 A  = -5
69 B  = 0
70 C  = 12
71 M  = 0
72 Gamma = .2
73 sigma = .2
74
75 # Kurve um Startparameter zu testen
76
77 # plot(data_$speed, data_$rate, cex = .5, pch = 20, xlab = "velocity / mm/s" , ylab
       = "counting rate / 1/s")
78 # curve(A * Voigt(x, M, sigma, Gamma) + C, add = TRUE, col = "#B00000", lty = 2)
79
80
81 fit = nlsLM(
82        y ~ A * Voigt(x, M, sigma, Gamma) + C,
83        data = dat,
84        start = list(
85               A  = A,
86               C  = C,
87               M  = M,
88               Gamma = Gamma,
89               sigma = sigma
90        ),
91        weights  = 1/(data_$s_rate^2)
92 )
93
94
95 fit_L = nlsLM(
```

```
96          y ~ A * Lorentz(x, M, Gamma) + C,
97          data = dat,
98          start = list(
99                  A  = A,
100                 C  = C,
101                 M  = M,
102                 Gamma = Gamma
103         ),
104         weights  = 1/(data_$s_rate^2)
105 )
106
107 fit_G = nlsLM(
108         y ~ A * Gauss(x, M, sigma) + C,
109         data = dat,
110         start = list(
111                 A  = A,
112                 C  = C,
113                 M  = M,
114                 sigma = sigma
115         ),
116         weights  = 1/(data_$s_rate^2)
117 )
118
119
120
121   A  = summary(fit)[[10]][1,1]
122   C  = summary(fit)[[10]][2,1]
123   M  = summary(fit)[[10]][3,1]
124   Gamma = summary(fit)[[10]][4,1]
125   sigma = summary(fit)[[10]][5,1]
126
127 s_A  = summary(fit)[[10]][1,2]
128 s_C  = summary(fit)[[10]][2,2]
129 s_M  = summary(fit)[[10]][3,2]
130 s_Gamma = summary(fit)[[10]][4,2]
131 s_sigma = summary(fit)[[10]][5,2]
132
133 peak  = function(x){
134         return(A * Voigt(x, M, sigma, Gamma) + C)
135 }
136
137 peak_L  = function(x){
138         return(summary(fit_L)[[10]][1,1] * Lorentz(x, summary(fit_L)[[10]][3,1],
              summary(fit_L)[[10]][4,1]) + C)
139 }
140
141 peak_G  = function(x){
142         return(summary(fit_G)[[10]][1,1] * Gauss(x, summary(fit_G)[[10]][3,1],
              summary(fit_G)[[10]][4,1]) + C)
```

```
143 }
144
145 curve(peak , add = TRUE, col = "#00B000")
146 curve(peak_L, add = TRUE, col = "#B00000", lty = 2, lwd = 2)
147 curve(peak_G, add = TRUE, col = "#B0B000", lty = 2, lwd = 2)
148
149 # Chi2 bestimmen
150 chi2  = round(sum(summary(fit)[[2]]^2)/summary(fit)[[4]][[2]],2)
151 chi2_G  = round(sum(summary(fit_G)[[2]]^2)/summary(fit_G)[[4]][[2]],2)
152 chi2_L  = round(sum(summary(fit_L)[[2]]^2)/summary(fit_L)[[4]][[2]],2)
153
154 cat("\nVoigt:\t",chi2, "\nGauss:\t", chi2_G, "\nLorentz:\t", chi2_L, "\n", sep = "")
155
156
157 # Auswertung
158
159 # Gamma
160   Gamma_eV  = Gamma / 1000 / 2 / 3e8 * 14.4e3 # in eV
161 s_Gamma_eV  = s_Gamma / 1000 / 2 / 3e8 * 14.4e3 # in eV
162 l_Gamma_ev  = round2(c(Gamma_eV, s_Gamma_eV)*1e9, 3)
163 cat("Gamma:", paste("\n\t(", round2(Gamma_eV*1e9, 3), "\t+- ", round2(s_Gamma_eV*1e9
      , 3), ")neV", sep = "", collapse = ""), "\n", sep = "")
164
165 #
166 #
167 #
168 #
169 # Isomere Verschiebung
170   Isoversch  = M # in eV
171 s_Isoversch  = s_M # in eV
172
173   Isoversch_eV = Isoversch / 1000 / 3e8 * 14.4e3 # in eV
174 s_Isoversch_eV = s_Isoversch / 1000 / 3e8 * 14.4e3 # in eV
175 l_Isoversch_eV = round2(c(Isoversch_eV, s_Isoversch_eV)*1e9, 3)
176
177 cat("Isomere Verschiebung:", paste("\n\t(", round2(Isoversch, 3), "\t+- ", round2(s_
      Isoversch, 3), ")mm/s", sep = "", collapse = ""), "\n", sep = "")
178 cat("Isomere Verschiebung:", paste("\n\t(", round2(Isoversch_eV*1e9, 3), "\t+- ",
      round2(s_Isoversch_eV*1e9, 3), ")neV", sep = "", collapse = ""), "\n", sep = "")
179
180 #
181 #
182 #
183 # Effektive Absorberdicke
184
185 # Daten für Sigma
186 I  = 1/2  # Kernpin [-]
187 I_  = 3/2  # kernspin angeregt [-]
188   alpha = 8.9  # Konversionskoeefizient [-]
```

```
s_alpha = .7
omega0 = 14.4e3 / 6.582e-16
                                    # Kreisfrequenz des 14.4keV Photon[s^-1]
beta = .022  # 57Fe Anteil in Probe [-]
dA  = 25e-6  # Dicke des Absorbers[10^-6 m]

#Eisendaten
rho_Fe = 7.87  # Dichte [g/cm^3]
M_Fe = 55.85  # Molare Mass e[g/mol]
Avocado = 6.022e23 # Avogadroconstante [g/mol]

fa  = .8  # Eisenanteil in Probe [-]

# Berechnung "Eisenatomdichte" [m^-3]
  na  = rho_Fe / M_Fe * Avocado *100^3 * .7 # (70 +-5)% Anteil Eisen
s_na  = rho_Fe / M_Fe * Avocado *100^3 * .05 # Fehler
# Wirkungsquerschnitt [m^2]
  sigma0 = 2 * pi * (3e8/omega0)^2 * ((2 * I_ + 1) / (2 * I +1)) * (1/(1 + alpha) )
s_sigma0 = sqrt( (2 * pi * (3e8/omega0)^2 * ((2 * I_ + 1) / (2 * I +1)) * -(1/(1 +
    alpha)^2))^2 * s_alpha^2)
# Effektive Absorberdicke [-]
  Ta  = fa * beta * sigma0 * dA * na
s_Ta = sqrt((fa * beta * sigma0 * dA * s_na)^2 + (fa * beta * s_sigma0 * dA * na)^2)

l_Ta = round2(c(Ta,s_Ta), 3)
cat("effektive Absorberdicke:", paste("\n\t ", l_Ta[1], "\t+- ", l_Ta[2], " ", sep =
     "", collapse = ""), "\n", sep = "")

#
#
#
# Debye-Weller-Faktor Quelle
Voigt0 = Voigt(M, M, sigma, Gamma)



  Z_inf = C
s_Z_inf = s_C
  Z_0 = A * Voigt(M, M, sigma, Gamma) + C
s_Z_0 = sqrt(
                              (s_A * Voigt(M, M, sigma, Gamma))^2
                      +
                              (s_C)^2
                      +
                                                  # nutze eine Pseudo Ableitung um
                                                      Fehler durch M, sigma und Gamma
                                                       zu betimmen
                              (s_M * A * (
                                      abs(Voigt0 - Voigt(M - s_M, M, sigma, Gamma))
```

```
234                                     + abs(Voigt0 - Voigt(M + s_M, M, sigma, Gamma))
235                                     )/2
236                             )^2
237                     +
238                             (s_sigma * A * ( # voigt-function zu stark von sigma
                                    abhängig für vernünftiges ergebnis
239                                       abs(Voigt0 - Voigt(M, M, sigma - 1e-6, Gamma))
240                                     + abs(Voigt0 - Voigt(M, M, sigma + 1e-6, Gamma))
241                                     )/2
242                             )^2
243                     +
244                             (s_Gamma * A * (
245                                       abs(Voigt0 - Voigt(M, M, sigma, Gamma - s_
                                              Gamma))
246                                     + abs(Voigt0 - Voigt(M, M, sigma, Gamma + s_
                                        Gamma))
247                                     )/2
248                             )^2
249             )
250
251   expTa = exp(-Ta / 2)
252 s_expTa = abs(.2*exp(-Ta / 2) * s_Ta)
253
254   J0 = besselI( Ta / 2, 0)
255                             # nutze auch hier Pseudoableitungen
256 s_J0 = sqrt(
257                     (s_Ta *
258                             (
259                               abs(besselI( Ta / 2, 0) - besselI( (Ta - s_Ta) / 2,
                                      0))
260                             + abs(besselI( Ta / 2, 0) - besselI( (Ta+ s_Ta) / 2, 0)
                                    )
261                             )/2
262                     )^2
263             )
264
265
266   fQ = (Z_inf - Z_0) / (Z_inf * (1 - expTa * J0))
267 s_fQ = sqrt(
268                     (
269                             (s_Z_inf * (Z_0 * (expTa * J0)))^2
270                     +
271                             (s_Z_0 * (Z_inf * (expTa * J0)))^2
272                     +
273                             (s_expTa * ((Z_inf - Z_0) * Z_inf * J0))^2
274                     +
275                             (s_J0 * ((Z_inf - Z_0) * Z_inf * expTa))^2
276                     )
277                     /(Z_inf * (1- expTa * J0))^4
```

```
278                             # erstes ^2 durch quotientenregel
279                             # zweites ^2 durch fehlerrechnung
280
281                 )
282
283 l_fQ = round2(c(fQ, s_fQ), 3)
284 cat("DW-Factor Quelle", paste("\n\t ", l_fQ[1], "\t+- ", l_fQ[2], " ", sep = "",
        collapse = ""), "\n", sep = "")
285
286 #
287 #
288 #
289 # lifespan of excited state
290   hbar = 6.582119e-16
291   tau_ = hbar / Gamma_eV
292 s_tau_ = hbar / Gamma_eV^2 * s_Gamma_eV
293
294 l_tau_ = round2(c(tau_, s_tau_)*1e9, 1)
295 cat("Lifetime", paste("\n\t ", l_tau_[1], "\t+- ", l_tau_[2], " ", sep = "",
        collapse = ""), "ns\n", sep = "")
296
297   T12 = log(2) * tau_
298 s_T12 = log(2) * s_tau_
299
300 l_T12 = round2(c(T12, s_T12)*1e9, 1)
301
302 cat("Halflife", paste("\n\t ", l_T12[1], "\t+- ", l_T12[2], " ", sep = "", collapse
        = ""), "ns\n", sep = "")
303
304
305 #
306 #
307 #
308 # Lebensdauer aus Ta
309   rel_broad  = 1.8066
310 s_rel_broad  = 0.0746
311
312   W = 2 * rel_broad
313 s_W = 2 * s_rel_broad
314
315   Gamma_alt_eV = Gamma_eV / rel_broad
316 s_Gamma_alt_eV = sqrt(
317                                                 (s_Gamma_eV / rel_broad)^2
318                                             +
319                                                 ( Gamma_eV / rel_broad^2)^2 * s_
                                                        rel_broad^2
320                                             )
321   tau_alt = hbar / Gamma_alt_eV
322 s_tau_alt = hbar / Gamma_alt_eV^2 * s_Gamma_alt_eV
```

```
323
324   T12_alt = log(2) * tau_alt
325 s_T12_alt = log(2) * s_tau_alt
326
327 l_tau_alt = round2(c(tau_alt, s_tau_alt)*1e9, 1)
328 l_T12_alt = round2(c(T12_alt, s_T12_alt)*1e9, 1)
329
330 cat("Lifetime_alt", paste("\n\t ", l_tau_alt[1], "\t+- ", l_tau_alt[2], " ", sep = "
      ", collapse = ""), "ns\n", sep = "")
331 cat("Halflife_alt", paste("\n\t ", l_T12_alt[1], "\t+- ", l_T12_alt[2], " ", sep = "
      ", collapse = ""), "ns\n", sep = "")
332
333 legend(
334        "bottomleft",
335        c(
336               "Experimental data",
337               "Errors",
338               "Voigtfit",
339               "Lorentzfit",
340               "Gaussfit"
341        ),
342        lty = c(
343               0,
344               0,
345               1,
346               2,
347               2,
348               0
349        ),
350        pch = c(
351               20,
352               15,
353               NA,
354               NA,
355               NA,
356               NA
357        ),
358        col = c(
359               "black",
360               "#0000B0",
361               "#00B000",
362               "#B00000",
363               "#B0B000",
364               "black"
365        ),
366        pt.cex = .5,
367        cex = .75,
368        bg = "white"
369 )
```

```
370
371 if(final == TRUE){
372         dev.off()
373 }
```

5.2.6 Iron

```
1  graphics.off()
2  path = dirname(sys.frame(1)$ofile)
3  final = TRUE # trigger for creating a png ( if == TRUE) or preview (if == FALSE)
4
5  library(minpack.lm)
6  library(Hmisc)
7  library(pracma)
8  library(RcppFaddeeva)
9
10 round2 = function(val, numb){
11         len = max(nchar(floor(val)))
12         return(formatC( round( val, numb ), format='f', digits=numb, width = numb +
               1 + len))
13 }
14
15 fs = function(){ # findspeed from manualy selected point
16         print(data_$speed[identify(data_$speed, data_$rate, n = 1)])
17 }
18
19   U  = 13.47 #untergrund
20 s_U  = .11
21
22 data_path = paste(dirname(path), "Daten", "Eisen", sep = "/")
23 exportpath= dirname(path)
24
25 files = dir(data_path, pattern = ".csv")
26 data_ = data.frame(speed = double(), dauer = double(), counts = integer() )
27
28
29 for(file_ in files){
30
31         data_temp = read.csv(paste(data_path, file_, sep = "/" ) , dec = ",", sep =
               "\t", as.is = TRUE, header = FALSE)
32         data_  = rbind(data_, data_temp)
33
34 }
35
36 names(data_) = c("speed", "dauer", "counts")
37
38 # erstmal normal rechnen
39 data_$rate = data_$counts / data_$dauer * 1000
40 data_$s_rate= sqrt(data_$rate / data_$dauer)
```

```
41 data_$s_rate_ = data_$s_rate * 10
42
43 #Untergrund abziehen
44 data_$rate  = data_$rate - U
45 data_$s_rate = sqrt(data_$s_rate^2 + s_U^2)
46 data_$s_rate_  = data_$s_rate * 1
47
48 data_ = data_[order(data_$speed),]
49
50
51 if(final == TRUE){
52        png(paste(exportpath, "Ironfit.png", sep = "/"), width = 6, height = 4,
              units = "in", res = 600)
53        par("mar" = c(4, 5, 1, 1))
54        par("oma" = c(0, 0, 0, 0))
55 }
56
57
58 plot(data_$speed, data_$rate, cex = .5, pch = 20, xlab = expression(v * " / " * over
       (mm, s) ) , ylab = expression(dot(N) *" / "* s^-1), type = "p", lwd = 2, ylim =
       range(data_$rate - data_$s_rate, data_$rate + data_$s_rate), xlim = range(data_$
       speed-7, data_$speed))
59 arrows(data_$speed, data_$rate - data_$s_rate_, data_$speed, data_$rate + data_$s_
       rate_, code = 3, angle = 90, length = 0.0175, col = "#0000B0")
60 abline(v = ((min(axTicks(1))*10):(max(axTicks(1))*10))/2, col = "#888888", lty = 3)
61 abline(v = axTicks(1), h = axTicks(2), col = "#888888")
62
63
64 # where the relevant peaks are
65 ranges = list(
66        c(-6.5, -4),
67        c(-4.2, -2.0),
68        c(-1.5, .5),
69        c( .2, 1.5),
70        c( 2.0, 4.5),
71        c( 4.0, 6.7)
72 )
73
74 As  = c()
75 Gammas = c()
76 Ms  = c()
77 cols = rainbow(6, alpha = .65)
78 C  = mean(c(head(data_$rate), tail(data_$rate)))
79
80 for(rng_ in ranges){
81        print(rng_)
82        rng  = intersect(
83                              which(data_$speed <= max(rng_)),
84                              which(data_$speed >= min(rng_))
```

```
85                          )
86        As   = c(As,
87                                (min(data_$rate[rng]) - max(data_$rate[rng]))
88                          )
89        Ms   = c(Ms,
90                                mean(rng_)
91                          )
92        Gammas = c(Gammas,
93                                diff(rng_)/7
94                          )
95
96  }
97  As   = c(-1.0, -1.0, -.3, -.3, -1.0, -1.0)
98  Gammas = c(.15, .15, .15, .15, .15, .15)
99   Ms   = c(-5, -3, -1, 1, 3, 5)
100 #Lorentz(x, x0, gamma)
101
102 peaks = function(x){
103       out  = C
104       for(i in 1:length(ranges)){
105             out = out + As[i] * Lorentz(x, Ms[i], Gammas[i])
106       }
107
108       return( out )
109 }
110
111 # curve(peaks, add = TRUE, col = "#B0000088", lty = 2, lwd = 2)
112
113 dat = data.frame(data_$speed, data_$rate, data_$s_rate)
114 names(dat) = c("x", "y", "s_y")
115
116
117 fit = nlsLM(
118       y ~ A1 * Lorentz(x, M1, Gamma1)
119         + A2 * Lorentz(x, M2, Gamma2)
120         + A3 * Lorentz(x, M3, Gamma3)
121         + A4 * Lorentz(x, M4, Gamma4)
122         + A5 * Lorentz(x, M5, Gamma5)
123         + A6 * Lorentz(x, M6, Gamma6)
124         + C,
125       data = dat,
126       start = list(
127             A1  = As[1],
128             A2  = As[2],
129             A3  = As[3],
130             A4  = As[4],
131             A5  = As[5],
132             A6  = As[6],
133             M1  = Ms[1],
```

```
                M2  = Ms[2],
                M3  = Ms[2],
                M4  = Ms[4],
                M5  = Ms[5],
                M6  = Ms[6],
                Gamma1 = Gammas[1],
                Gamma2 = Gammas[2],
                Gamma3 = Gammas[3],
                Gamma4 = Gammas[4],
                Gamma5 = Gammas[5],
                Gamma6 = Gammas[6],
                C  = C
        ),
        weights  = 1/(data_$s_rate^2),
        control  = list(
                                        ftol = 10^-30,
                                        ptol = 10^-30,
                                        gtol = 10^-30,
                                        maxiter = 1024
                                )
)


chi2  = sum((peaks(dat$x) - dat$s_y)^2 / dat$y)/df.residual(fit)


coeff  = summary(fit)[[10]]
  As  = coeff[1:6, 1]
  Ms  = coeff[7:12, 1]
  Gammas = coeff[13:18, 1]
  C   = coeff[19,1]
s_C   = coeff[19,2]

s_As  = coeff[1:6, 2]
s_Ms  = coeff[7:12, 2]
s_Gammas = coeff[13:18, 2]


A1  = As[1]
A2  = As[2]
A3  = As[3]
A4  = As[4]
A5  = As[5]
A6  = As[6]
M1  = Ms[1]
M2  = Ms[2]
M3  = Ms[2]
M4  = Ms[4]
M5  = Ms[5]
```

```
M6  = Ms[6]
Gamma1 = Gammas[1]
Gamma2 = Gammas[2]
Gamma3 = Gammas[3]
Gamma4 = Gammas[4]
Gamma5 = Gammas[5]
Gamma6 = Gammas[6]

s_A1  = s_As[1]
s_A2  = s_As[2]
s_A3  = s_As[3]
s_A4  = s_As[4]
s_A5  = s_As[5]
s_A6  = s_As[6]
s_M1  = s_Ms[1]
s_M2  = s_Ms[2]
s_M3  = s_Ms[2]
s_M4  = s_Ms[4]
s_M5  = s_Ms[5]
s_M6  = s_Ms[6]
s_Gamma1 = s_Gammas[1]
s_Gamma2 = s_Gammas[2]
s_Gamma3 = s_Gammas[3]
s_Gamma4 = s_Gammas[4]
s_Gamma5 = s_Gammas[5]
s_Gamma6 = s_Gammas[6]


curve(peaks, add = TRUE, col = "#00B00088", lty = 1, lwd = 2, from = par("usr")[1],
    to = par("usr")[2])
  Ms_eV  = Ms / 1000/ 3e8 * 14.4e3 # in eV
s_Ms_eV  = s_Ms / 1000/ 3e8 * 14.4e3 # in eV

  Ms_neV = round2( Ms_eV * 1e9, 2)
s_Ms_neV = round2(s_Ms_eV * 1e9, 2)


  Gammas_eV = Gammas/ 2 / 1000 / 3e8 * 14.4e3 # in eV
s_Gammas_eV = s_Gammas/ 2 / 1000 / 3e8 * 14.4e3 # in eV

#
#
#
# isomerieverschiebung
  Isoversch = (Ms[c(1,2,3)] + Ms[c(6,5,4)]) / 2  # in mm/s
s_Isoversch = sqrt((s_Ms[c(1,3,5)]^2+s_Ms[c(2,4,6)]^2) / 4 ) # in mm/s
  # / 1000 / 3e8 # in eV
# / 2 / 1000 / 3e8 # in eV
  Isoversch_eV = Isoversch / 1000 / 3e8 * 14.4e3
```

```
231 s_Isoversch_eV = s_Isoversch / 1000 / 3e8 * 14.4e3
232
233   mean_Isoversch = sum(Isoversch/s_Isoversch^2)/ sum(1/s_Isoversch^2)
234 s_mean_Isoversch = 1/sqrt(sum(1/s_Isoversch^2))
235
236
237   mean_Isoversch_eV = sum(Isoversch_eV/s_Isoversch_eV^2)/ sum(1/s_Isoversch_eV^2)
238 s_mean_Isoversch_eV = 1/sqrt(sum(1/s_Isoversch_eV^2))
239
240 cat("Peaks:", paste("\n\t(", round2(Ms , 2), "\t+- ", round2(s_Ms , 2), ")mm/s", sep
      = "", collapse = ""), "\n", sep = "")
241 cat("Peaks:", paste("\n\t(", Ms_neV, "\t+- ", s_Ms_neV, ")neV" , sep = "", collapse
    = ""), "\n", sep = "")
242 cat("Gammas:", paste("\n\t(", round2(Gammas, 2), "\t+- ", round2(s_Gammas, 2), ")mm/
    s", sep = "", collapse = ""), "\n", sep = "")
243 cat("Gammas:", paste("\n\t(", round2(Gammas_eV*1e9, 2), "\t+- ", round2(s_Gammas_eV*
    1e9, 2), ")neV", sep = "", collapse = ""), "\n", sep = "")
244 cat("Isomere Verschiebung:", paste("\n\t(", round2(Isoversch, 3), "\t+- ", round2(s_
    Isoversch, 3), ")mm/s", sep = "", collapse = ""), "\n", sep = "")
245 cat("mittelwert:", paste("\n\t(", round2(mean_Isoversch, 3), "\t+- ", round2(s_mean_
    Isoversch, 3), ")mm/s", sep = "", collapse = ""), "\n", sep = "")
246 cat("Isomere Verschiebung:", paste("\n\t(", round2(Isoversch_eV*1e9, 3), "\t+- ",
    round2(s_Isoversch_eV*1e9, 3), ")neV", sep = "", collapse = ""), "\n", sep = "")
247 cat("mittelwert:", paste("\n\t(", round2(mean_Isoversch_eV*1e9, 3), "\t+- ", round2(
    s_mean_Isoversch_eV*1e9, 3), ")neV", sep = "", collapse = ""), "\n", sep = "")
248 cat("\nchi2:\t",chi2, "\n", sep = "")
249
250 #
251 #
252 #
253 # kernmagentone
254
255 # kernmagneton groundstate
256   mu_N  = 3.152e-8 # eV/T
257   mu_g  = 90.44e-3 # * mu_N
258 s_mu_g  = 0.07e-3 # * mu_N
259
260   Dva  = Ms[[6]] - Ms[[1]]
261   Dvb  = Ms[[5]] - Ms[[2]]
262   Dvc  = Ms[[4]] - Ms[[3]]
263
264 s_Dva  = sqrt(s_Ms[[6]]^2 + s_Ms[[1]]^2)
265 s_Dvb  = sqrt(s_Ms[[5]]^2 + s_Ms[[2]]^2)
266 s_Dvc  = sqrt(s_Ms[[4]]^2 + s_Ms[[3]]^2)
267
268   mu_a_ab = mu_g * (Dva - Dvb) / (Dva/3 - Dvb)
269   mu_a_ac = mu_g * (Dvc - Dva) / (Dva/3 + Dvc)
270   mu_a_bc = mu_g * 3 * (Dvc - Dvb) / (Dvc + Dvb)
271
```

```
s_mu_a_ab = (Dva - 3 * Dvb)^-2 *sqrt(
                              ((Dva - Dvb) * (Dva - 3 * Dvb)^2 * s_mu_g)^2
                      + ( -6 * mu_g * Dvb * s_Dva)^2
                      + ( 6 * mu_g * Dva * s_Dvb)^2
                      )
s_mu_a_ac = (Dva + 3 * Dvc)^-2 *sqrt(
                              ((Dvc - Dva) * (Dva + 3 * Dvc)^2* s_mu_g)^2
                      + (-12 * mu_g * Dvc * s_Dva)^2
                      + ( 12 * mu_g * Dva * s_Dvc)^2
                      )
s_mu_a_bc = (Dvc + Dvb)^-2 * sqrt(
                              ( 3 * (Dvc - Dvb) * (Dvc + Dvb)^2* s_mu_g)^2
                      + ( -6 * mu_g * Dvc * s_Dvb)^2
                      + ( 6 * mu_g * Dvb * s_Dvc)^2
                      )

  mu_as  = c( mu_a_ab, mu_a_ac, mu_a_bc)
s_mu_as  = c(s_mu_a_ab, s_mu_a_ac, s_mu_a_bc)

  mean_mu_as = sum(mu_as/s_mu_as^2)/ sum(1/s_mu_as^2)
s_mean_mu_as = 1/sqrt(sum(1/s_mu_as^2))

cat("mu_gs:", paste("\n\t(", round2( mu_as, 4), "\t+- ", round2(s_mu_as , 4), ")mu_N
    ", sep = "", collapse = ""), "\n", sep = "")
cat("mean:", paste("\n\t(", round2( mean_mu_as, 4), "\t+- ", round2(s_mean_mu_as ,
    4), ")mu_N", sep = "", collapse = ""), "\n", sep = "")
cat("lit.:", paste("\n\t(", "-0.1549", "\t+- ", "0.0002", ")mu_N", sep = "",
    collapse = ""), "\n", sep = "")


#
#
#
# Magnetfeld im Kern

  B_a  = Dva * 14.4e3/ 2 / 3e11 / (-mu_a_ab + mu_g) / mu_N
  B_b  = Dvb * 14.4e3/ 2 / 3e11 / (-mu_a_ac/3 + mu_g) / mu_N
  B_c  = Dvc * 14.4e3/ 2 / 3e11 / ( mu_a_bc/3 + mu_g) / mu_N


s_B_a  = abs(B_a)*
                      sqrt(
                              (s_Dva / Dva)^2 + (s_mu_a_ab / ( mu_a_ab + mu_g))^2 + (
                                  s_mu_g / ( mu_a_ab + mu_g))^2
                      )
s_B_b  = abs(B_b)*
                      sqrt(
                              (s_Dvb / Dvb)^2 + (s_mu_a_ac / (-mu_a_ac/3 + mu_g))^2 +
                                   (s_mu_g / (-mu_a_ac/3 + mu_g))^2
```

```
316                       )
317 s_B_c  = abs(B_c)*
318                       sqrt(
319                              (s_Dvc / Dvc)^2 + (s_mu_a_bc / ( mu_a_bc/3 + mu_g))^2 +
                                     (s_mu_g / ( mu_a_bc/3 + mu_g))^2
320                       )
321   Bs  = c( B_a, B_b, B_c)
322 s_Bs  = c(s_B_a, s_B_b, s_B_c)
323
324
325   mean_Bs = sum(Bs/s_Bs^2)/ sum(1/s_Bs^2)
326 s_mean_Bs = 1/sqrt(sum(1/s_Bs^2))
327
328
329 cat("Bs:", paste("\n\t(", round2( Bs, 4), " \\pm ", round2(s_Bs , 4), ")T", sep = ""
      , collapse = ""), "\n", sep = "")
330 cat("mean:", paste("\n\t(", round2( mean_Bs, 4), " \\pm ", round2(s_mean_Bs , 4), ")
      T", sep = "", collapse = ""), "\n", sep = "")
331 cat("lit.:", paste("\n\t(", "~33", ")T", sep = "", collapse = ""), "\n", sep = "")
332
333  legend(
334        "bottomleft",
335        c(
336               "Experimental data",
337               "Errors",
338               "Fit",
339               as.expression(bquote(chi[red]^2: ~ .(signif(chi2, 4)))),
340               as.expression(bquote(mu[1] == (.(Ms_neV[1]) %+-% .(s_Ms_neV[1]))*neV)
                     ),
341               as.expression(bquote(mu[2] == (.(Ms_neV[2]) %+-% .(s_Ms_neV[2]))*neV)
                     ),
342               as.expression(bquote(mu[3] == (.(Ms_neV[3]) %+-% .(s_Ms_neV[3]))*neV)
                     ),
343               as.expression(bquote(mu[4] == (.(Ms_neV[4]) %+-% .(s_Ms_neV[4]))*neV)
                     ),
344               as.expression(bquote(mu[5] == (.(Ms_neV[5]) %+-% .(s_Ms_neV[5]))*neV)
                     ),
345               as.expression(bquote(mu[6] == (.(Ms_neV[6]) %+-% .(s_Ms_neV[6]))*neV)
                     )
346        ),
347        lty = c(
348               0,
349               0,
350               1,
351               0,
352               rep(0, 6)
353        ),
354        pch = c(
355               20,
```

```
356                 15,
357                 NA,
358                 NA,
359                 rep(NA, 6)
360         ),
361         col = c(
362                 "black",
363                 "#0000B0",
364                 "#00B000",
365                 "black",
366                 rep("black", 6)
367         ),
368         pt.cex = .5,
369         cex = .75,
370         bg = "white"
371 )
372
373
374
375 if(final == TRUE){
376         dev.off()
377 }
```

6 List of Figures

7 List of Tables

8 Bibliography

[1] Köhli M., *Szintillationszähler*. Versuchsanleitung, Universität Freiburg, April 2011

[2] *Szintillationszähler*. Gerätebeschreibung, Universität Freiburg, September 2005

[3] Kotyk, T., *Versuche zur Radioaktivität im Physikalischen Fortgeschrittenen Praktikum an der Albert-Ludwigs-Universität Freiburg*. Zulassungsarbeit, Universität Freiburg, November 2005

[4] Margulies, S., Debrunner, P., Frauenfelder H. *Transmission and line broadening in the Mossbauer Effect. II*, Nuclear Instruments and Methods 21 (1963) 217-231, North-Holland Publishing Co.

[5] Stone N. J., *Table of Nuclear Magnetic Dipole and Electric Quadrupole Moments*, Oxford Physics

[6] Kistner O. C., Sunyar A. W., *Phys. Rev. 139, B295*, Published 26 July 1965

[7] Wegener H., *Der Mössbauer-Effekt und seine Anwendung in Physik und Chemie*, Hochschultaschenbücher, 1964

[8] Brent Fultz, *Mössbauer Spectrometry in Characterization of Materials*, John Wiley, New York, 2011

[9] Mayer-Kuckuk, T., *Kernphysik*, John Wiley, New York, 2011 Vieweg+Teubner Verlag, 2002