

# Geschwindigkeitsmessung am Mößbauer-Versuch (Velocity measurement at the Moessbauer experiment)

Bachelorarbeit  
von

**Alexis Michel**

am Institut für Experimentelle Teilchenphysik

ETP-Bachelor-KA/2022-05

Referent: Prof. Dr. G. Quast  
Korreferent: Dr. J. Wolf

Bearbeitungszeit: 15.10.2021 – 19.04.2022



# Erklärung zur Selbstständigkeit

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der gültigen Fassung vom 24.05.2018 beachtet habe.

Karlsruhe, den 19.04.2022, \_\_\_\_\_  
Alexis Michel

Als Prüfungsexemplar genehmigt von

Karlsruhe, den 19.04.2022, \_\_\_\_\_  
Prof. Dr. G. Quast



# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
<b>2. Theoretische Grundlagen</b>	<b>3</b>
2.1. Mößbauereffekt . . . . .	3
2.1.1. Grundlegendes zum Mößbauereffekt . . . . .	3
2.1.2. Änderung der Energie des Photons . . . . .	3
2.1.3. Zustände von $^{57}\text{Fe}$ . . . . .	4
2.1.4. Messung und beobachtbare Effekte . . . . .	5
2.1.4.1. Chemische Verschiebung . . . . .	5
2.1.4.2. Magnetische Hyperfeinstrukturaufspaltung . . . . .	6
2.1.4.3. Quadrupolaufspaltung . . . . .	6
2.2. Interferometer . . . . .	8
2.2.1. Aufbau des Interferometers . . . . .	8
2.2.2. Signal des Interferometers . . . . .	8
2.2.3. Chirp-Signal . . . . .	10
2.3. Autokorrelation . . . . .	11
2.3.1. Autokorrelation an Beispielen . . . . .	11
<b>3. Bisherige Versuchsdurchführung</b>	<b>15</b>
3.1. Ursprünglicher Versuch im Praktikum . . . . .	15
3.1.1. Mechanischer Teil . . . . .	15
3.1.2. Radioaktiver Teil . . . . .	15
3.1.3. Optischer Teil . . . . .	16
3.1.4. Elektronischer Teil . . . . .	16
3.1.4.1. DFG-1200 . . . . .	16
3.1.4.2. MDU-1200 . . . . .	16
3.1.4.3. MVC-450 . . . . .	17
3.1.4.4. Zählerkarte . . . . .	17
3.1.4.5. Signalkette Zählrohr . . . . .	17
3.2. Bearbeiteter Versuch mit FPGA-Trapezfilter . . . . .	18
<b>4. Alternative Geschwindigkeitsmessung</b>	<b>19</b>
4.1. Ziel der alternativen Geschwindigkeitsmessung . . . . .	19
4.2. Beschreibung der Testmethode . . . . .	19
4.3. Beschreibung des Signals . . . . .	19
4.4. Bekannte Probleme . . . . .	20
4.5. Getestete Verfahren . . . . .	22
4.5.1. Peakfinding . . . . .	22
4.5.1.1. Allgemeine Beschreibung und Idee des Verfahrens . . . . .	22
4.5.1.2. Spezifische Beschreibung . . . . .	22
4.5.1.3. Modifikation: Mittelwertfilter . . . . .	23
4.5.1.4. Fazit . . . . .	23

4.5.2.	FFT . . . . .	25
4.5.2.1.	Allgemeine Beschreibung und Idee des Verfahrens . . . . .	25
4.5.2.2.	Spezifische Beschreibung . . . . .	25
4.5.2.3.	Fazit . . . . .	26
4.5.3.	Autokorrelation . . . . .	27
4.5.3.1.	Allgemeine Beschreibung und Idee des Verfahrens . . . . .	27
4.5.3.2.	Spezifische Beschreibung . . . . .	27
4.5.3.3.	Gegenkontrolle: Visualisierung der Autokorrelation . . . . .	28
4.5.3.4.	Modifikation: Bedingung für die Autokorrelationsfunktion . . . . .	29
4.5.3.5.	Modifikation: Gleitende Frames . . . . .	29
4.5.3.6.	Gegenkontrolle: Testsignal . . . . .	30
4.5.3.7.	Modifikation: Längere Frames . . . . .	31
4.5.3.8.	Anwendung im Versuch und Vergleich zum alten Verfahren . . . . .	33
4.5.3.9.	Fazit . . . . .	35
<b>5.</b>	<b>Fazit</b>	<b>37</b>
	<b>Anhang</b>	<b>39</b>
A.	Musterprotokoll . . . . .	39
	<b>Literaturverzeichnis</b>	<b>67</b>

# Abbildungsverzeichnis

2.1. Zerfallsschema von $^{57}\text{Co}$ , Werte aus [6, S. 443] . . . . .	4
2.2. Transmissionsspektrum von Vacromium . . . . .	6
2.3. Transmissionsspektrum von Eisen . . . . .	7
2.4. Transmissionsspektrum von $\text{FeSO}_4$ . . . . .	7
2.5. Schematische Zeichnung eines Michelson-Interferometers . . . . .	8
2.6. Linearer Chirp . . . . .	10
2.7. Autokorrelation einer Konstante . . . . .	12
2.8. Autokorrelation einer Geraden . . . . .	13
2.9. Autokorrelation von Sinus und Cosinus . . . . .	14
4.1. Verlauf des aufgezeichneten Signals für rund eine Schwingung der Membran	20
4.2. Geschwindigkeitsmessung über Peakfinding ohne Mittelwertfilter . . . . .	22
4.3. Geschwindigkeitsmessung über Peakfinding mit Mittelwertfilter . . . . .	23
4.4. Signalframe und zugehörige Fouriertransformierte . . . . .	25
4.5. Erstes Signalframe und zugehörige Autokorrelation mit erstem Minimum . .	28
4.6. Erste Autokorrelationsmessung . . . . .	29
4.7. Autokorrelationsmessung mit erweiterten Bedingungen . . . . .	30
4.8. Autokorrelationsmessung mit gleitenden Frames am echten Signal . . . . .	31
4.9. Testsignal, doppelt gespiegelter linearer Chirp . . . . .	32
4.10. Autokorrelationsmessung mit gleitenden Frames am Testsignal . . . . .	32
4.11. Autokorrelationsmessung mit langen Frames am echten Signal . . . . .	33
4.12. Autokorrelationsmessung mit langen Frames am Testsignal . . . . .	34
4.13. Vergleich der neuen Geschwindigkeitsmessung mit dem alten Verfahren . . .	34





# 1. Einleitung

Mößbauerspektroskopie ist ein wichtiges physikalisches Analyseverfahren, welches sogar von mehreren Rovern der NASA auf dem Mars durchgeführt wird [1].

Auch am KIT wird im Rahmen des physikalischen Praktikums Mößbauerspektroskopie durchgeführt, beim Versuch *der Mößbauereffekt*. Allerdings befindet sich der Versuch in einem sehr veralteten Zustand und entspricht nicht den Ansprüchen an moderne Messtechnik.

In letzter Zeit gibt es Ambitionen, den Versuch auf moderne Messtechnik zu portieren, sodass ein “Field Programmable Gate Array”, kurz *FPGA*, die Auswertung übernimmt. Teile des Versuches wurden dafür bereits umgestellt.

Diese Arbeit befasst sich primär mit der Geschwindigkeitsmessung am Versuch, die mithilfe eines Michelson-Interferometers durchgeführt wird. Das Problem der Geschwindigkeitsmessung reduziert sich letztlich auf das Problem, bei einem sinusartigen Signal veränderlicher Frequenz, einem sogenannten *Chirp*-Signal, eine Frequenzmessung durchzuführen. Dafür werden unterschiedliche Verfahren überlegt, die auf einem FPGA implementierbar sind. Getestet werden diese Verfahren in der Programmiersprache Python.

Dabei werden, durch die ganze Arbeit hinweg, die Python-Libraries *numpy* [2], *matplotlib* [3] und *pandas* [4] verwendet.



## 2. Theoretische Grundlagen

### 2.1. Mößbauereffekt

Da sich diese Arbeit mit dem Experiment Mößbauereffekt beschäftigt, wird an dieser Stelle auf den Mößbauereffekt, sowie auf die weitere Theorie hinter dem Experiment eingegangen. Sofern keine andere Quelle angegeben ist, stammen alle Informationen aus dem sogenannten “Blauen Buch” des Praktikums, aus dem Kapitel “Versuch 9: Der Mößbauereffekt”, [5, S.171-190].

#### 2.1.1. Grundlegendes zum Mößbauereffekt

Unter dem Mößbauereffekt versteht man die rückstoßfreie Emission und Absorption von Photonen an Atomkernen. Das wird dadurch erreicht, dass Atome, die an Emission oder Absorption beteiligt sind, in einem Kristallgitter eingebettet sind.

Bei Emission (Atomkern verlässt einen angeregten Zustand, ein Photon wird frei) oder Absorption (Photon trifft auf Atomkern, wird absorbiert) gilt jeweils Erhaltung für Energie und Impuls.

Im Falle der Emission erhält das Photon nahezu vollständig die freiwerdende Energie, da der Atomkern in einem festen Kristall eingebettet wird, der Impuls wird auf den ganzen Kristall verteilt und die Bewegung (somit auch die Energie, die der Kristall aufnimmt) ist vernachlässigbar.

Im Falle der Absorption laufen diese Prozesse umgekehrt ab, die Beobachtungen bleiben jedoch dieselben. Hier trifft ein Photon auf einen Atomkern, der in einem Kristallgitter fest eingebettet ist. Der Atomkern erhält dabei nahezu die volle Energie des Photons, da die Bewegung des Kristalls (und somit auch wieder die Energie, die der Kristall aufnimmt), vernachlässigbar ist.

Kombiniert man einen Emmitter und einen Absorber mit gleicher oder sehr ähnlicher Anregungsenergie, so erhält man das Analyseverfahren der Mößbauer-Spektroskopie, was dem Experiment Mößbauereffekt im Praktikum entspricht.

#### 2.1.2. Änderung der Energie des Photons

Oft ist es praktisch, die Energie, die die Photonen eines Emitters haben, zu variieren. Dies wird durch eine Bewegung des Emitters erreicht, die emittierten Photonen haben dann eine dopplerverschobene Energie. Die Photonenenergie im Ruhesystem des Emitters ist  $E = pc$ , das um die Geschwindigkeit  $v$  dopplerverschobene Photon hat die Energie  $E^*$  mit

$$E^* = \frac{1}{\sqrt{1-\beta^2}}(E + vp) = E \left(1 + \frac{v}{c}\right), \quad (2.1)$$

wobei die Näherung  $\beta = v/c \ll 1$  verwendet wird. Sofern eine feste Anregungsenergie  $E$  für einen Absorber bekannt ist, wird noch die Information der Geschwindigkeit des Emitters benötigt, um die Energie des Photons im Laborsystem zu bestimmen.

### 2.1.3. Zustände von $^{57}\text{Fe}$

Für den Versuch relevant ist der Zerfall von  $^{57}\text{Co}$  zu  $^{57}\text{Fe}$ . Dieser Zerfall passiert im Emittor, der am Versuch verwendet wird.

Das  $^{57}\text{Co}$  mit einer Halbwertszeit von rund 270 Tagen zerfällt durch Elektroneneinfang zu angeregtem  $^{57}\text{Fe}$  im Zustand  $I = 5/2$ , mit der Kernspinquantenzahl  $I$ . Aus diesem Zustand kann das Eisen in den angeregten Zustand  $I = 3/2$  zerfallen, oder direkt in den Grundzustand mit  $I = 1/2$ , wobei der Zerfall in den Zustand  $I = 3/2$  etwa um einen Faktor 10 wahrscheinlicher ist, als der direkte Zerfall in den Grundzustand. In beiden Fällen werden Photonen frei, im Übergang von  $I = 5/2$  zu  $I = 3/2$  haben diese Photonen eine Energie von rund 122 keV, der Zerfall von  $I = 5/2$  zu  $I = 1/2$  produziert Photonen mit etwa 136,4 keV Energie. Beide dieser Übergänge treten auf, sind aber für den Versuch nicht relevant.

Der für den Versuch relevante Übergang ist von  $I = 3/2$  zu  $I = 1/2$ . Hier wird ein Photon mit der Energie 14,4 keV frei, was genau der Differenz der anderen beiden Übergänge entspricht. Umgekehrt kann ein Eisen-Atomkern auch angeregt werden (Übergang von  $I = 1/2$  zu  $I = 3/2$ ), wenn er von einem Photon mit Energie 14,4 keV getroffen wird. Der Zerfall von  $^{57}\text{Co}$  zu  $^{57}\text{Fe}$  ist in Abbildung 2.1 noch als Termschema zu sehen, der relevante Übergang ist dort mit rot eingezeichnet.

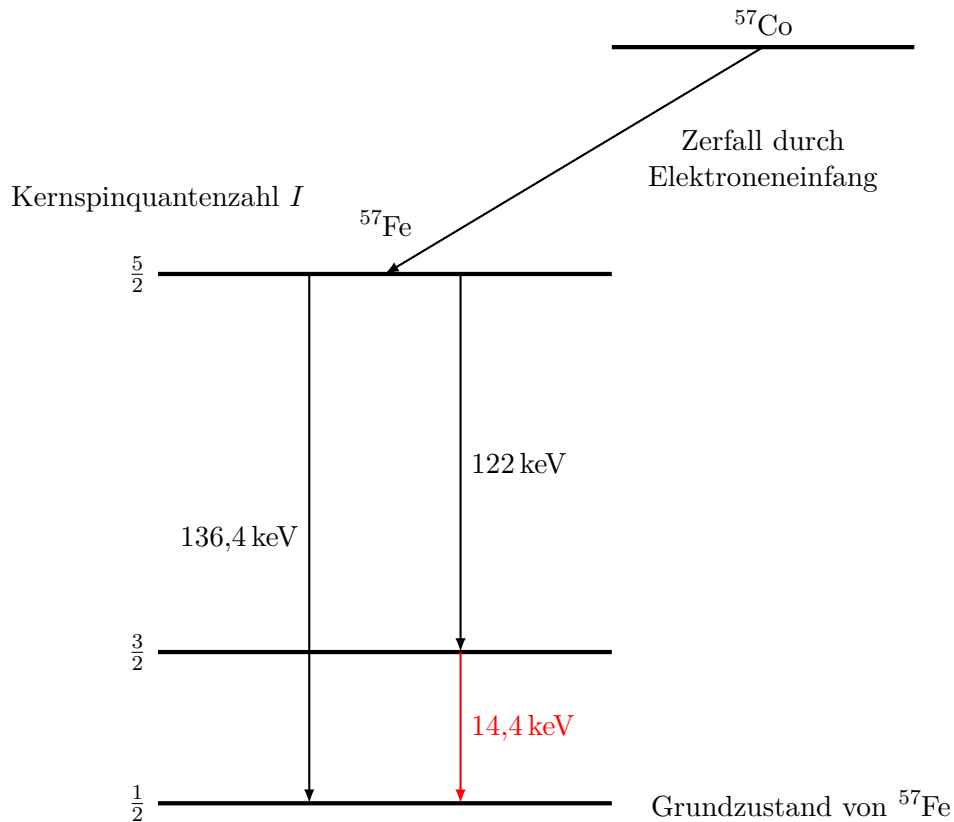


Abbildung 2.1.: Zerfallsschema von  $^{57}\text{Co}$ , Werte aus [6, S. 443]

Der Atomkern hat nicht nur die Kernspinquantenzahl  $I$ , sondern auch noch die magnetische Quantenzahl  $m$ , die in Schritten von 1 alle Zahlen von  $-I$  bis  $+I$  annehmen kann. Durch äußere Felder kann es sein, dass eine Entartung der Energien auftritt, die sich am Versuch beobachten lässt, mehr dazu im nächsten Kapitel 2.1.4.

Zur magnetischen Quantenzahl sei noch gesagt, dass eine Auswahlregel gilt:

$$\Delta m = m_{I=3/2} - m_{I=1/2} \in \{-1, 0, +1\}. \quad (2.2)$$

Beim Übergang von  $I = 3/2$  zu  $I = 1/2$  (und umgekehrt) sind somit von den acht möglichen Übergängen unter Berücksichtigung der magnetischen Quantenzahlen zwei Übergänge verboten ( $m_{I=3/2} = +3/2 \rightarrow m_{I=1/2} = -1/2$  und  $m_{I=3/2} = -3/2 \rightarrow m_{I=1/2} = +1/2$  sind verboten); die anderen sechs Übergänge sind erlaubt.

#### 2.1.4. Messung und beobachtbare Effekte

Die Messung, die am Versuch durchgeführt wird, ist eine Transmissionsmessung. Bei der Messung wird ein Emmitter (am Versuch  $^{57}\text{Co}$ ) mit näherungsweise linearer Geschwindigkeit hin- und herbewegt. Der Emmitter produziert Photonen mit einer scharfen Energie, am Versuch entspricht diese Energie dem Übergang mit 14,4 keV, der in Kapitel 2.1.3 beschrieben wird. Alle Photonen, die aus dem Emmitter stammen, haben eine dopplerverschobene Energie, die von der Geschwindigkeit des Emitters abhängt. Die dopplerverschobene Energie wird in Kapitel 2.1.2 hergeleitet.

Unter dem Emmitter sitzt ein Absorber, der untersucht werden soll. Unter dem Absorber sitzt ein Zählrohr, welches Ereignisse und deren Energie aufzeichnet. Es sind nur die Ereignisse relevant, die im Bereich der 14,4 keV liegen, für diese wird die Rate in Abhängigkeit zur Geschwindigkeit des Emitters untersucht.

Im Fall von Resonanzabsorption haben die Photonen, die aus dem Emmitter kommen, gerade die Energie, die der Absorber zur Anregung benötigt. Dann werden vermehrt Photonen vom Absorber aufgenommen, die dann nicht mehr das Zählrohr erreichen; am Zählrohr sieht man einen Einbruch, der durch eine Voigt-Funktion (oder approximativ mit einer Gauß- oder Lorentz-Funktion) beschrieben wird.

Ohne weitere Effekte zu berücksichtigen, würde man davon ausgehen, dass die Resonanz in der Ruhelage des Emitters ist ( $v = 0$ ), da Emmitter und Absorber dieselbe Anregungsenergie haben. Ohne die Betrachtung weiterer Effekte ist auch zu erwarten, nur eine Resonanz zu sehen, also, dass keine Entartung vorliegt. Diese Erwartungen stimmen allerdings nicht und müssen durch weitere, im Folgenden beschriebene Effekte ergänzt werden.

##### 2.1.4.1. Chemische Verschiebung

Die chemische Verschiebung, auch Isomerieverschiebung genannt, sorgt für eine konstante Verschiebung des Geschwindigkeitsspektrums. Dadurch kann es sein, dass die Resonanz nicht bei  $v = 0$  liegt, sondern bei

$$v = C \cdot (|\Psi_A(0)|^2 - |\Psi_E(0)|^2) \cdot (\bar{R}_a^2 - \bar{R}_g^2), \quad (2.3)$$

wobei  $C$  eine Konstante ist,  $\Psi_A$  die Elektronenverteilung der Hülle im Absorber,  $\Psi_E$  die Elektronenverteilung der Hülle im Emmitter,  $\bar{R}_a$  der Ladungsradius des Kerns im angeregten Zustand und  $\bar{R}_g$  der Ladungsradius des Kerns im Grundzustand ist. Eine chemische Verschiebung  $v \neq 0$  tritt dann nach Gleichung 2.3 nur auf, wenn sich die Ladungsradien des Kerns im Grundzustand und im angeregten Zustand unterscheiden, und Emmitter und Absorber unterschiedliche Elektronenverteilungen in der Hülle haben. Die Elektronenverteilung in der Hülle wird durch chemische Bindungen beeinflusst, daher der Name chemische Verschiebung. Sind Absorber und Emmitter identisch aufgebaut, ist somit keine chemische Verschiebung zu sehen, da die Elektronenverteilung in der Hülle identisch ist.

Ein Spektrum, in dem die chemische Verschiebung auftritt, ist in Abbildung 2.2 zu sehen. Die Abbildung stammt aus dem selbsterstellten Musterprotokoll, welches im Anhang eingefügt ist. Zu sehen ist das Transmissionsspektrum (Anzahl Events über Geschwindigkeit) von Vacromium, mit den einzelnen Messwerten und einer gefitteten Gauß-Kurve. Es ist klar zu sehen, dass die einzige Resonanz verschoben ist, das Minimum der Transmission liegt nicht bei  $v = 0$ , sondern eher bei  $v = -0,25 \text{ mm/s}$ . Das liegt an der chemischen Verschiebung.

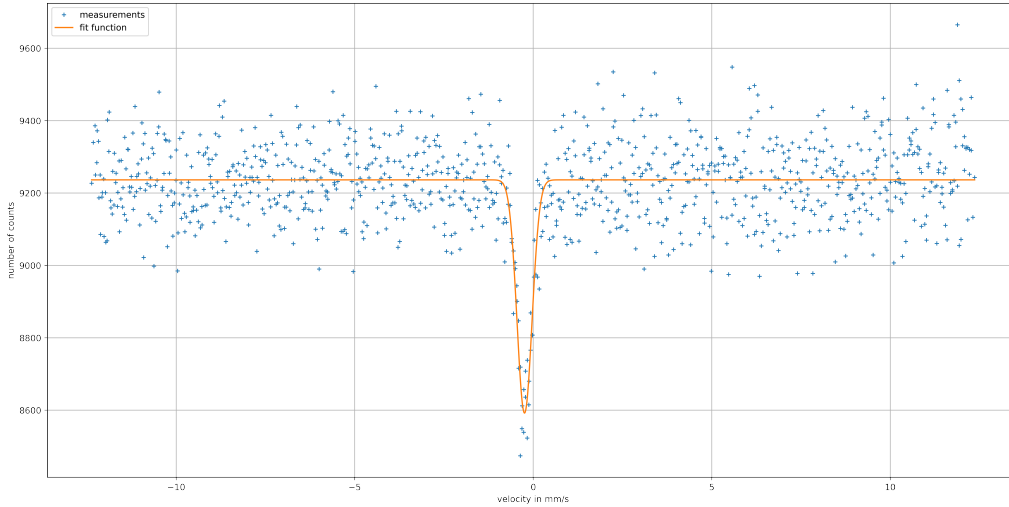


Abbildung 2.2.: Transmissionsspektrum von Vacromium

#### 2.1.4.2. Magnetische Hyperfeinstrukturaufspaltung

Die magnetische Hyperfeinstrukturaufspaltung sorgt für eine Aufspaltung der Resonanzlinie in mehrere Resonanzlinien. Verursacht wird dies durch ein Magnetfeld im Inneren des Atomkerns zusammen mit dem Zeeman-Effekt. Es findet eine Aufhebung der vorherigen Entartung statt; die zuvor entarteten Zustände für alle sechs Übergänge der magnetischen Quantenzahl  $m$  haben dann unterschiedliche Energien. Die Theorie hinter den Übergängen und der magnetischen Quantenzahl  $m$  ist in Kapitel 2.1.3 zu finden. Aufgrund des Zeeman-Effektes gilt für die Energiedifferenz  $\Delta E_m$  vom entarteten Zustand:

$$\Delta E_m = -\frac{m}{I} \cdot \mu_I \cdot B, \quad (2.4)$$

wobei  $B$  das Magnetfeld und  $\mu_I$  das magnetische Moment zur Quantenzahl  $I$  ist. Im Versuch hat der Emittor keine inneren B-Felder, somit ist es einfacher, nur den Absorber zu untersuchen. Für die Übergänge ergibt sich dann eine Energieänderung von

$$E_0 \left(1 + \frac{v}{c}\right) = E'_0 - \frac{m_a \cdot \mu_a \cdot B}{I_a} + \frac{m_g \cdot \mu_g \cdot B}{I_g}, \quad (2.5)$$

mit dem angeregten Zustand  $I_a = 3/2$ , dem Grundzustand  $I_g = 1/2$  und dem magnetischen Moment des Grundzustandes  $\mu_g$ , sowie dem magnetischen Moment  $\mu_a$  des angeregten Zustandes. Zusätzlich zur magnetischen Hyperfeinstrukturaufspaltung kann auch noch eine chemische Verschiebung stattfinden; das ist in Gleichung 2.5 durch die Energie  $E'_0$  berücksichtigt. Diese Energie ist die Energie  $E_0$  mit einer möglichen Isometrie-Verschiebung. Eine eigene Auswertung aus dem Musterprotokoll ist in Abbildung 2.3 zu sehen. Hier ist die 6-fache Aufspaltung durch magnetische Hyperfeinstrukturaufspaltung zu erkennen, ebenso eine leichte chemische Verschiebung.

#### 2.1.4.3. Quadrupolaufspaltung

Ein weiterer Effekt, welcher für eine Aufspaltung sorgt, ist die Quadrupolaufspaltung. Diese wird durch einen Gradienten  $\partial^2/(\partial z^2)$  des elektrischen Feldes  $V$  am Atomkern erzeugt, wenn der Kern eine asymmetrische Ladungsverteilung hat, also das Quadrupolmoment  $Q$  nicht verschwindet. Um den Effekt einfacher beobachten zu können, wird der Emittor ohne innere elektrische Feldgradienten gewählt. Für die Energieaufspaltung im Absorber gilt

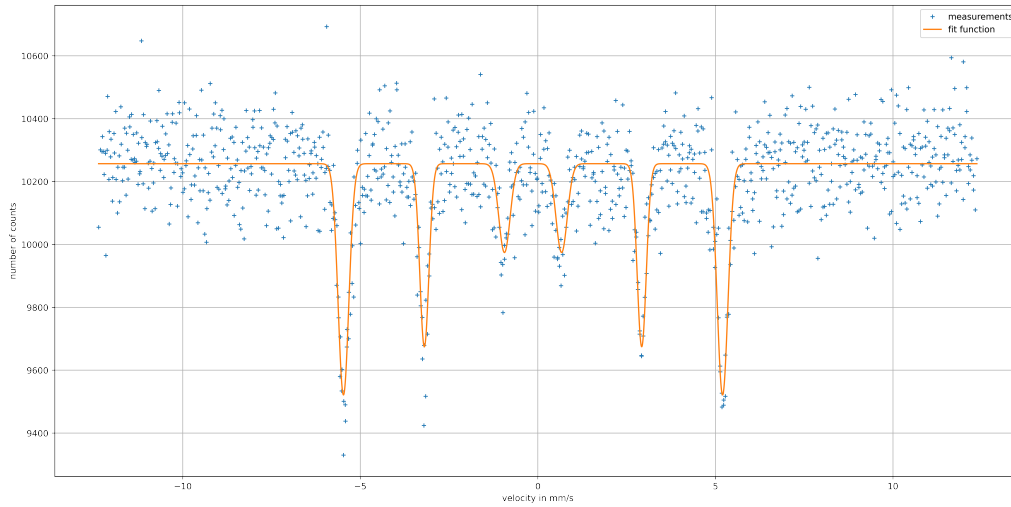


Abbildung 2.3.: Transmissionsspektrum von Eisen

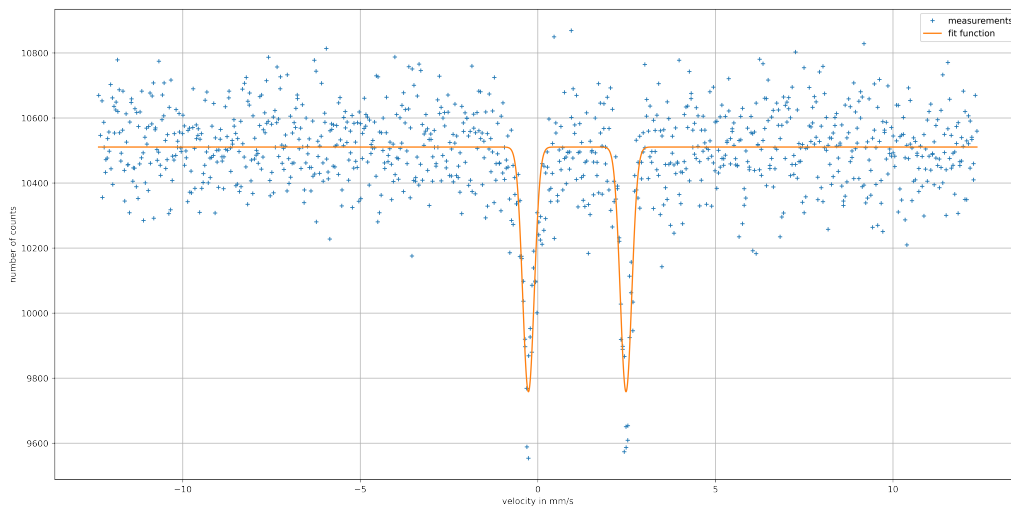
dann

$$\Delta E_Q(m) = \frac{e \cdot Q}{4} \cdot \left( \frac{\partial^2 V}{\partial z^2} \right) \cdot \frac{3m^2 - I(I+1)}{3I^2 - I(I+1)}, \quad (2.6)$$

mit der Elementarladung  $e$ ;  $m$  und  $I$  sind die aus Kapitel 2.1.3 bekannten Quantenzahlen. Zudem tritt die Quadrupolaufspaltung nur bei Kernen mit  $I > 1$  auf, also hier nur für den angeregten Zustand  $I = 3/2$ , der Grundzustand  $I = 1/2$  ist nicht betroffen.

Da Gleichung 2.6 nur vom Betrag von  $m$  abhängt, spielt das Vorzeichen von  $m$  keine Rolle. Somit ergeben sich für den Zustand  $I = 3/2$  zwei unterschiedliche Energien: eine Energie für  $m = \pm 1/2$  und eine Energie für  $m = \pm 3/2$ . Diese unterschiedlichen Energien zeigen sich wieder in einer Aufspaltung in zwei Resonanzen. Eine überlagernde Isometrie-Verschiebung ist wieder möglich.

In Abbildung 2.4 ist eine eigene Auswertung von  $\text{FeSO}_4$ , einem Eisensalz, zu sehen. Klar zu erkennen ist die doppelte Aufspaltung, verursacht durch Quadrupolaufspaltung. Ebenso ist deutlich eine Isometrie-Verschiebung erkennbar, da das Spektrum nicht symmetrisch zur Geschwindigkeit  $v = 0$  liegt, sondern verschoben ist.

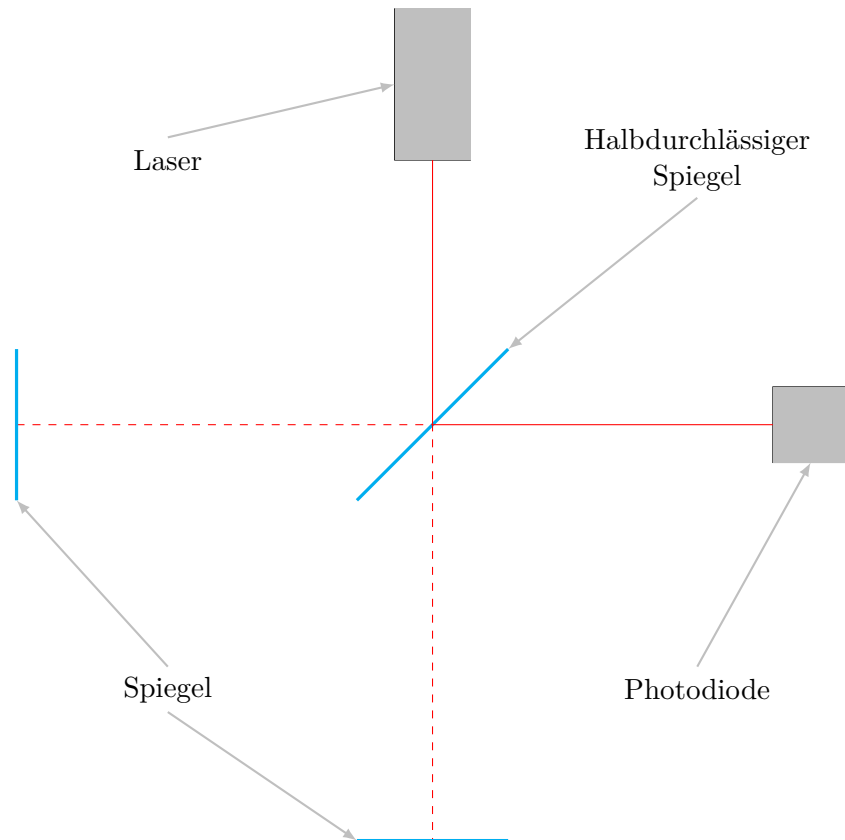
Abbildung 2.4.: Transmissionsspektrum von  $\text{FeSO}_4$

## 2.2. Interferometer

Im Versuch kommt ein gewöhnliches Michelson-Interferometer zum Einsatz. Da sich diese Arbeit primär mit der Auswertung des Signals eines solchen Interferometers zur Geschwindigkeitsmessung beschäftigt, wird an dieser Stelle auf die Funktionsweise des Interferometers eingegangen.

### 2.2.1. Aufbau des Interferometers

In Abbildung 2.5 ist ein Michelson-Interferometer abgebildet, wie es im Versuch verwendet wird.



**Abbildung 2.5.:** Schematische Zeichnung eines Michelson-Interferometers

Ein Laser kommt beim Interferometer als Quelle für kohärentes Licht zum Einsatz. Kernstück des Interferometers ist ein halbdurchlässiger Spiegel, der als Strahlenteiler eingesetzt wird. Vom Laser-Licht, welches auf den halbdurchlässigen Spiegel trifft, wird etwa die Hälfte reflektiert und die andere Hälfte transmittiert.

Das am Strahlenteiler zunächst transmittierte Licht wird an einem Spiegel reflektiert, es trifft wieder auf den Strahlenteiler und wird dort zum Teil reflektiert. Das am Strahlenteiler zunächst reflektierte Licht wird an einem anderen Spiegel reflektiert, trifft danach auch wieder auf den Strahlenteiler und wird dann teilweise transmittiert. Am Strahlenteiler wird so der zunächst aufgeteilte Strahl wieder zusammengeführt. An der Photodiode wird die Intensität der beiden, zuvor aufgeteilten, nun zusammengeführten Strahlen ausgewertet.

### 2.2.2. Signal des Interferometers

Die Herleitung des Signals des Interferometers orientiert sich am Blauen Buch [5, S.186], geht jedoch noch darüber hinaus.

Da kohärentes Laser-Licht mit Wellenlänge  $\lambda$  verwendet wird, tritt hier Interferenz auf.



Die beiden Strahlen können sich also auslöschen (destruktive Interferenz) oder verstärken (konstruktive Interferenz), je nach Phasenunterschied  $\delta$ . Im Versuch wird einer der Spiegel still gehalten, der andere bewegt sich mit dem Emitter.

Die beiden aufgeteilten Strahlen lassen sich als ebene Wellen betrachten,

$$y_1 = y_0 \cdot \sin(kx - \omega t - \delta_1), \quad (2.7)$$

$$y_2 = y_0 \cdot \sin(kx - \omega t - \delta_2). \quad (2.8)$$

Hier sind  $y_1$  und  $y_2$  die Amplituden der einzelnen Wellen,  $y_0$  die maximale Amplitude beider Wellen. Die Kreiswellenzahl  $k$  ist  $k = 2\pi/\lambda$ ,  $x$  ist der Ort, an dem gemessen wird,  $\omega$  die Kreiswellenzahl und  $t$  die Zeit. Die Variablen  $\delta_1$  und  $\delta_2$  beschreiben noch eine Phasenverschiebung der einzelnen Wellen.

Der gemessene Strahl besteht aus der Summe beider Strahlen, also gilt

$$y = y_1 + y_2 = y_0 \cdot (\sin(kx - \omega t - \delta_1) + \sin(kx - \omega t - \delta_2)) \quad (2.9)$$

$$= 2 \cdot y_0 \cdot \cos\left(\frac{\delta_2 - \delta_1}{2}\right) \cdot \sin\left(kx - \omega t - \frac{\delta_1 + \delta_2}{2}\right), \quad (2.10)$$

wobei das Additionstheorem  $\sin(a+b) + \sin(a+c) = 2 \cos(b/2 - c/2) \sin(a + b/2 + c/2)$  verwendet wurde. Im Folgenden wird für den Phasenunterschied zwischen den beiden Strahlen  $\delta = \delta_2 - \delta_1$  verwendet.

Bei der Messung mit der Photodiode ist die Intensität  $I$  des Lichtes relevant. Für die Intensität  $I$  muss die Amplitude  $y$  noch quadriert werden und über große Zeiten gemittelt werden, also gilt

$$I = \langle y^2 \rangle = 4y_0^2 \cdot \cos^2\left(\frac{\delta}{2}\right) \cdot \left\langle \sin^2\left(kx - \omega t - \frac{\delta_1 + \delta_2}{2}\right) \right\rangle \quad (2.11)$$

$$= 2y_0^2 \cdot \cos^2\left(\frac{\delta}{2}\right), \quad (2.12)$$

weil  $\langle \sin^2 \rangle = 1/2$ , was direkt aus  $\sin^2 + \cos^2 = 1$  und  $\cos(x) = \sin(x + \pi/2)$  folgt.

Mit der Identität  $\cos^2(x) = 1/2 \cdot (\cos(2x) + 1)$  ergibt sich

$$I = y_0^2 \cdot (\cos \delta + 1) \quad (2.13)$$

Die Phasendifferenz  $\delta$  zwischen den beiden Lichtstrahlen hängt von der Längendifferenz der Strahlengänge  $\Delta L$  ab. Es gilt

$$\delta = k \cdot \Delta L = \frac{2\pi}{\lambda} \cdot \Delta L. \quad (2.14)$$

Für einen sehr kleinen Abschnitt, in dem die Geschwindigkeit eines Spiegels  $v$  als konstant angenommen werden kann (und der andere Spiegel Geschwindigkeit 0 hat), gilt

$$\Delta L = 2 \cdot v \cdot t + l_0, \quad (2.15)$$

wobei  $l_0$  die Längendifferenz (modulo  $\lambda$ ) angibt, die die Spiegel bei  $v = 0$  haben. Einsetzen der Zwischenergebnisse in Gleichung 2.13 liefert

$$I = y_0^2 \cdot \left( \cos\left(\frac{4\pi v}{\lambda} t + \tilde{\phi}\right) + 1 \right), \quad (2.16)$$

wobei die Phasenverschiebung  $\tilde{\phi} = 2\pi l_0/\lambda$  die Längendifferenz für  $v = 0$  berücksichtigt.

Da Sinus und Cosinus nur in der Phase verschoben sind, lässt sich Gleichung 2.16 auch durch einen Sinus ausdrücken, indem  $\tilde{\phi}$  durch  $\phi = \tilde{\phi} + \pi/2$  ersetzt wird. Zudem lässt sich die Frequenz  $f$  des Signals zu  $f(v) = 2v/\lambda$  identifizieren. Dann gilt

$$I = y_0^2 \cdot (\sin(2\pi f(v)t + \phi) + 1), \text{ mit } f(v) = \frac{2v}{\lambda} \implies v(f) = \frac{f\lambda}{2}. \quad (2.17)$$

Zuletzt wird noch die Abhängigkeit der Spannung  $U$  an der Photodiode betrachtet. Theoretisch ist die Spannung  $U$  an der Photodiode proportional zur Intensität  $I$  des einfallenden Laser-Lichts des Interferometers, allerdings kann es hier zu einer konstanten Verschiebung kommen, wenn die Photodiode Umgebungslicht detektiert. Diese Verschiebung soll zusammen mit der Verschiebung des Signals (+1-Term aus Gleichung 2.17) zum Spannungsmittelwert  $\langle U \rangle$  zusammengefasst werden. Die Proportionalitätskonstante zwischen  $U$  und  $I$  wird mit der Amplitude  $y_0^2$  zur Spannungsamplitude  $U_0$  zusammengefasst. Als theoretisch hergeleitete Signalspannung ergibt sich

$$U(t) = U_0 \cdot \sin(2\pi f(v(t))t + \phi) + \langle U \rangle, \quad (2.18)$$

wobei die Geschwindigkeit  $v(t)$  als Funktion der Zeit betrachtet wird.

### 2.2.3. Chirp-Signal

Ein Signal der Form von Gleichung 2.18 wird als sogenanntes *Chirp-Signal* bezeichnet. Solche Signale werden dadurch charakterisiert, dass sie von der Form

$$S(t) = A \cdot \sin(2\pi f(t)t + \phi) \quad (2.19)$$

sind, mit dem Signal  $S$ , der Amplitude  $A$ , der Zeit  $t$ , der Frequenz  $f(t)$  und dem Phasenwinkel  $\phi$ .

Ist  $f(t)$  linear in der Zeit, also in der Form  $f(t) = \alpha t + \beta$ , spricht man von einem *linearen Chirp*. Im Versuch wird die Geschwindigkeit linear angesteuert. Da die Frequenz nach Gleichung 2.17 proportional zur Geschwindigkeit ist, ist die Frequenz auch linear und es liegt ein linearer Chirp vor.

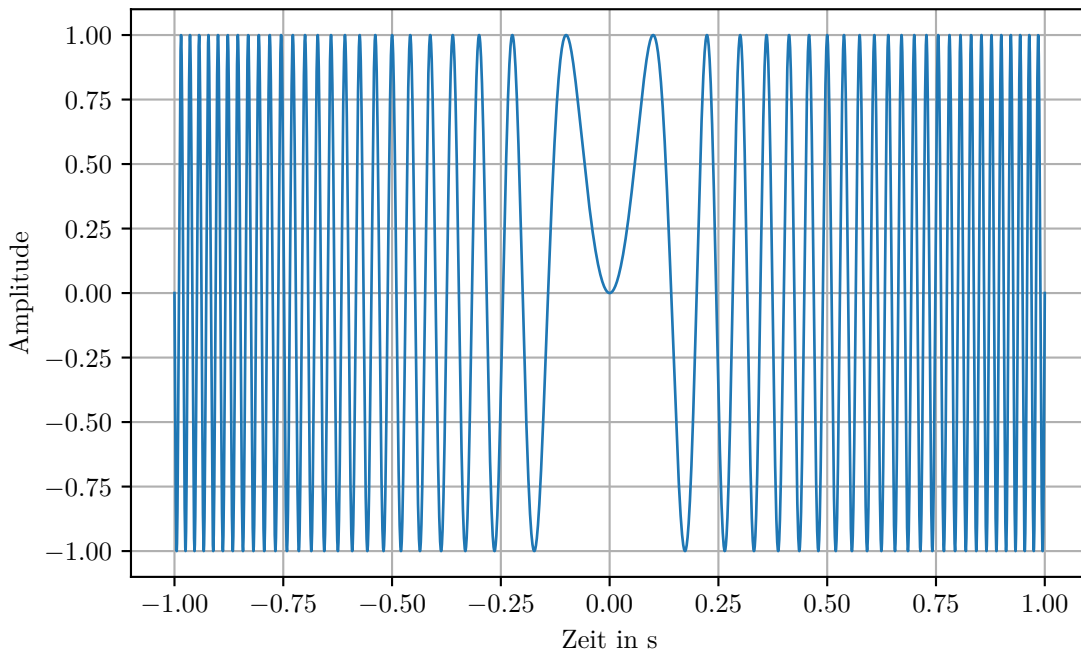


Abbildung 2.6.: Linearer Chirp

In Abbildung 2.6 ist zur Veranschaulichung ein linearer Chirp abgebildet. Für diesen Plot sind die Parameter  $A = 1$  und  $\phi = 0$  gesetzt. Die Frequenz wechselt innerhalb von zwei Sekunden linear von  $f = -50$  Hz auf  $f = 50$  Hz. Bei  $t = 0$  ist hier  $f = 0$ , was auch  $v = 0$  impliziert.

Das Interferometer-Signal ist aus vielen Abschnitten wie in Abbildung 2.6 zusammengesetzt, da die Geschwindigkeit in näherungsweise linearen Abschnitten abgefahren wird. Dabei gehen die unterschiedlichen Abschnitte sozusagen fließend ineinander über, das Signal und seine Ableitung sind stetig.

### 2.3. Autokorrelation

Der Hauptteil der Auswertung des Signals des Interferometers stützt sich auf Autokorrelation. Deshalb wird an dieser Stelle auf die Theorie hinter der Autokorrelation eingegangen. Quelle für dieses Kapitel ist [7]

Zunächst wird der allgemeine Fall betrachtet. Die nicht-normierte Autokorrelationsfunktion  $A$  der komplexen Funktion  $f(t)$  bestimmt sich zu

$$A(f(t)) = f \star f = \int_{-\infty}^{\infty} f(\tau) \bar{f}(t - \tau) d\tau = \int_{-\infty}^{\infty} f(\tau) \bar{f}(t - \tau) d\tau, \quad (2.20)$$

wobei  $\star$  die Kreuzkorrelation und  $\bar{f}$  das komplex Konjugierte von  $f$  bezeichnet.

Das betrachtete Signal ist jedoch diskret, es liegen  $N$  Werte mit konstantem Abstand vor, die als Werte  $a_i$  des Vektors  $\vec{a}$  mit Länge  $N$  aufgefasst werden, mit  $i \in [0, N - 1]$ , für Indices  $i$  außerhalb des Definitionsbereiches ist  $a_i = 0$ , also wird mit Nullen aufgefüllt. Im diskreten Fall wird das Integral zur Summe und es gilt dann

$$A_i(\vec{a}) = \sum_{j=0}^{N-1} a_j \bar{a}_{j+i}, \quad (2.21)$$

wobei die nicht normierte Autokorrelationsfunktion hier eigentlich zum Autokorrelationsvektor  $\vec{A}$  wird (analog zum Werte-Vektor  $\vec{a}$ ). Trotzdem wird weiterhin die Bezeichnung Autokorrelationsfunktion verwendet, da für kleine Abstände zwischen den diskreten Messwerten eine Funktion gut approximiert wird.

Da das betrachtete Signal nicht komplexwertig ist, wird auch die Autokorrelation lediglich für reelle Werte betrachtet und es gilt

$$A_i(\vec{a}) = \sum_{j=0}^{N-1} a_j a_{j+i} = (a_0, \dots, a_{N-1-i})^T \cdot (a_i, \dots, a_{N-1})^T. \quad (2.22)$$

Die diskrete Autokorrelation kann also als Skalarprodukt berechnet werden, wobei vom Vektor  $\vec{a}$  auf einer Seite des Skalarproduktes vom unteren Ende  $i$  Einträge entfernt werden, auf der anderen Seite werden vom oberen Ende des Vektors  $i$  Einträge entfernt.

Die Autokorrelation hat die Eigenschaft, dass der erste Wert ( $i = 0$ ) direkt der größte Wert ist. Deshalb wird häufig die normierte Autokorrelationsfunktion  $\rho$  verwendet, bei der alle Werte durch den ersten Wert dividiert werden,

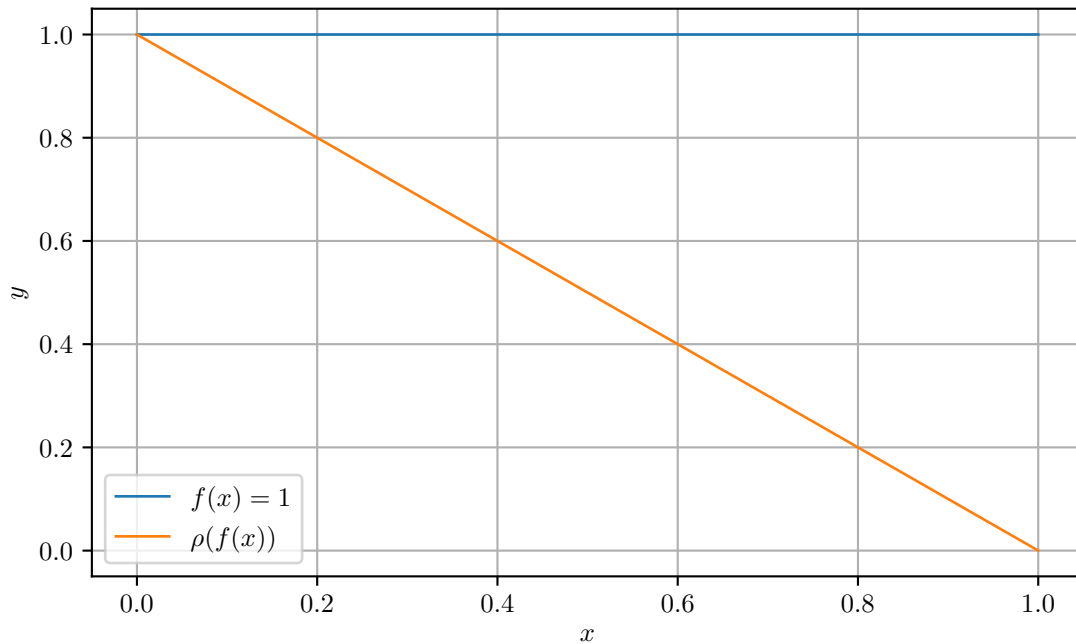
$$\rho_i(\vec{a}) = \frac{A_i(\vec{a})}{A_0(\vec{a})} = \frac{A_i(\vec{a})}{\vec{a}^2}. \quad (2.23)$$

#### 2.3.1. Autokorrelation an Beispielen

Anschaulich beschreibt die Autokorrelation, was die gemeinsame Fläche einer unverschobenen Kurve mit einer verschobenen Kurve ist, in Abhängigkeit zu der Verschiebung.

Die Autokorrelations-Funktion ist per Definition mit der Normierung 1 bei  $x = 0$  und am rechten Ende  $x_{\max}$  bei 0. Der Verlauf zwischen diesen Randbedingungen ist interessant.

Als erstes Beispiel wird die Autokorrelation einer Konstanten betrachtet. Das ist in Abbildung 2.7 zu sehen. Die gewählte konstante Funktion ist  $f(x) = 1$ , die im Intervall von 0 bis 1 berechnet wird. Die dazugehörige Autokorrelations-Funktion  $\rho(f(x))$  ist in



**Abbildung 2.7.:** Autokorrelation einer Konstante

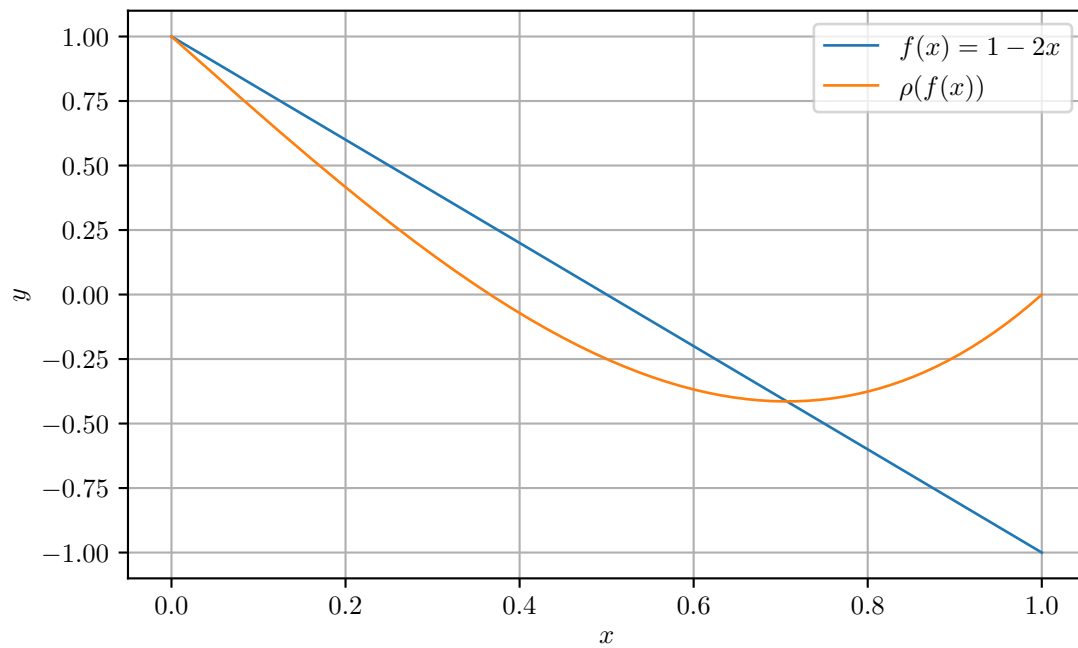
orange eingezeichnet. Weil  $f(x)$  eine Konstante ist, ist die Verschiebung linear zwischen den Randpunkten. Es ergibt sich eine Gerade.

Als nächstes Beispiel wird die Autokorrelation einer Geraden betrachtet. Als Funktion wurde hier  $f(x) = 1 - 2x$  gewählt, wieder im Intervall  $x \in [0, 1]$ . Die Funktion und die zugehörige Autokorrelationsfunktion sind in Abbildung 2.8 zu sehen.

Es ist klar zu erkennen, dass diese Autokorrelationsfunktion ein Minimum hat, bei  $x \approx 0,7$ . Wichtig dafür ist der Vorzeichenwechsel der Funktion  $f$ , da so in der Verschiebung Werte unterschiedlichen Vorzeichens aufeinandertreffen, was im negativen Vorzeichen der Autokorrelation resultiert.

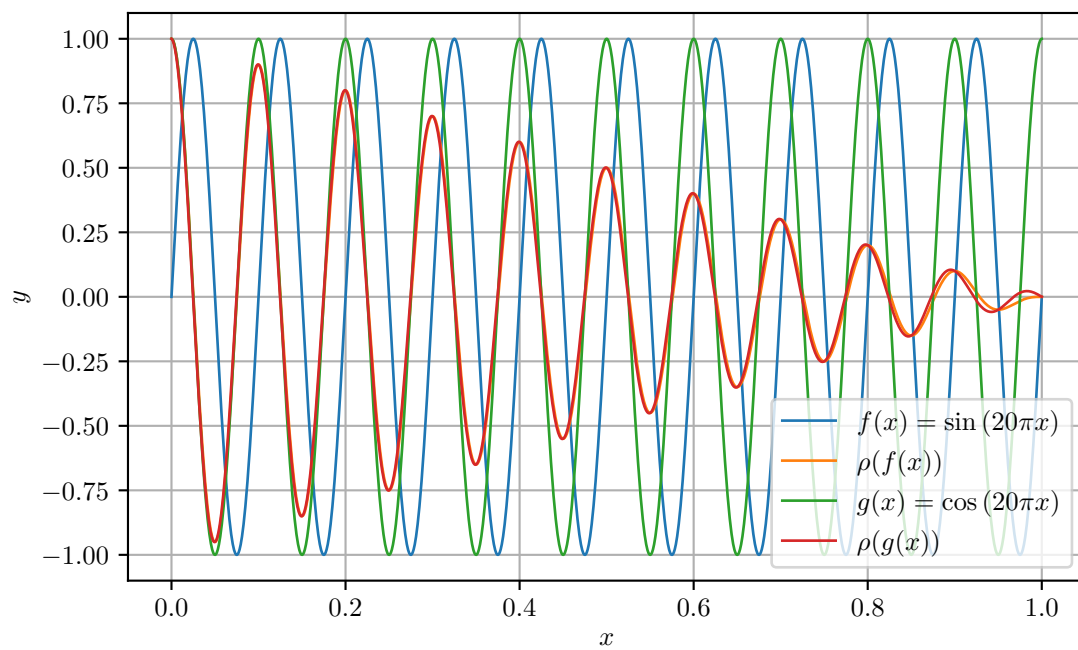
Als letztes und wichtigstes Beispiel wird die Autokorrelation eines periodischen Signals betrachtet. Als Funktion werden hier Sinus und Cosinus betrachtet, in der Form  $f(x) = \sin(20\pi x)$  und  $g(x) = \cos(20\pi x)$ . Die Funktionen wurden so gewählt, dass im Intervall  $x \in [0, 1]$  jeweils genau zehn Perioden enthalten sind. Zudem muss die Schwingung um  $y = 0$  stattfinden, damit die Autokorrelation sinnvolle Werte liefert. Das ist in Abbildung 2.9 zu sehen.

Die Autokorrelationsfunktion eines periodischen Signals ist wieder ein periodisches Signal, welches allerdings linear in seiner Amplitude abfällt. Die Periode einer periodischen Funktion und ihrer Autokorrelation ist identisch. Die Autokorrelationsfunktion ist somit geeignet, Periode (und damit die Frequenz) eines periodischen Signals zu bestimmen. Dabei ist die Form des Originalsignals ziemlich egal, auch starkes Rauschen hat keinen großen Einfluss auf die Autokorrelationsfunktion. Die Phase hat einen kleinen, aber existenten Einfluss, wie in Abbildung 2.9 zu erkennen ist. Ein Sinus und ein Cosinus mit identischer Frequenz sind dasselbe Signal, aber um eine Phase  $\pi/2$  verschoben. Anfangs verläuft die Autokorrelationsfunktion beider Funktionen quasi identisch, allerdings ist ab ca.  $x = 0,9$  eine Aufspaltung zu erkennen, die nur auf die Phasenverschiebung zurückzuführen ist. In 2.9 sind zudem nur ganze Perioden enthalten. Die Bestimmung der Periode funktioniert



**Abbildung 2.8.:** Autokorrelation einer Geraden

auch mit angeschnittenen Perioden, allerdings können hier ähnliche Abweichungen wie bei einer verschobenen Phase auftreten, die sich primär im hinteren Bereich der Autokorrelationsfunktion zeigen.



**Abbildung 2.9.:** Autokorrelation von Sinus und Cosinus

## 3. Bisherige Versuchsdurchführung

### 3.1. Ursprünglicher Versuch im Praktikum

Der im Praktikum eingesetzte Versuch ist ein Standard-Lehrversuch des Herstellers *Wissenschaftliche Elektronik GmbH* (kurz WissEl). Die Bauteile mit Funktion werden hier erklärt.

#### 3.1.1. Mechanischer Teil

Das Kernstück des Versuches ist der *Mössbauer Velocity Transducer* (kurz MVT). Dieses Bauteil besteht aus einem Hohlzylinder, auf dessen Mittelachse ein zweiter Zylinder sitzt. Der innere Zylinder ist über eine Membran mit dem äußeren Hohlzylinder verbunden. Im mittleren Zylinder sind Permanentmagneten angebracht, der Hohlzylinder verfügt über zwei Spulen, eine Antriebsspule und eine Regelspule. Durch Ströme in der Antriebsspule wird der innere Zylinder zum Schwingen gebracht.

Das Funktionsprinzip des MVT ähnelt sehr dem eines Lautsprechers (oder Kopfhörertreibers). In allen Fällen gibt es ein bewegliches Bauteil mit Permanentmagneten, welches über eine Membran mit der Außenwelt verbunden ist und mit der Hilfe von Spulen zum Schwingen gebracht wird.

#### 3.1.2. Radioaktiver Teil

Auf der Unterseite des schwingenden Zylinders befindet sich ein radioaktives Präparat. Dieses Präparat besteht aus  $^{57}\text{Co}$ , welches durch Elektroneneinfang in angeregtes  $^{57}\text{Fe}$  zerfällt. Für den Versuch relevant ist ein Übergang von  $^{57}\text{Fe}$  vom angeregten Zustand in den Grundzustand. Bei diesem Übergang wird ein Photon einer sehr scharfen Energie von 14,4 keV frei. Da dieses Photon aus dem Atomkern stammt, handelt es sich trotz der vergleichsweise niedrigen Energie um  $\gamma$ -Strahlung.

Dadurch, dass das radioaktive Präparat mit dem inneren Zylinder des MVTs mitschwingt, wird die vom Präparat emittierte Strahlung, insbesondere die 14,4 keV-Linie, dopplerverschoben. Auf die scharfe Photonenenergie von 14,4 keV wird also, je nach Bewegungsrichtung, ein wenig Energie hinzugefügt oder abgezogen.

Vom radioaktiven Präparat ausgestrahlte Photonen werden nach unten weiter beobachtet. Dort sitzt ein Geiger-Müller-Zählrohr, welches die Photonen registriert. Direkt über dem Zählrohr lassen sich zu untersuchende Proben (im Versuch sind das Eisen, Vacromium,  $\text{FePO}_4$  und  $\text{FeSO}_4$ ) befestigen, die dann im Strahlengang zwischen radioaktivem Präparat und Zählrohr sind.

Für jede dieser Proben wird dann im Versuch ein Absorptionsspektrum aufgenommen, indem man kontinuierlich die Ereignisse am Zählrohr aufzeichnet und zusätzlich die momentane Geschwindigkeit des radioaktiven Präparates aufzeichnet.

### 3.1.3. Optischer Teil

Zur präzisen Geschwindigkeitsmessung ist auf den MVT ein Interferometer angebracht, dieses ist Teil des *Mößbauer Velocity Calibrator 450* (kurz MVC-450 oder MVC) von WissEl. Dabei handelt es sich vom Aufbau her um ein Michelson-Interferometer. Ein Laser wird als Quelle verwendet. Im originalen Versuchsaufbau des Herstellers ist hier ein HeNe-Laser mit einer Wellenlänge von  $\lambda = 632,8 \text{ nm}$  vorgesehen. Dieser wurde jedoch nach einem Defekt des HeNe-Lasers durch einen günstigeren Halbleiterlaser mit  $\lambda = 650 \text{ nm}$  ersetzt. Der Strahlenteiler (halbdurchlässiger Spiegel) ist auch hier zu finden, er lässt sich über Schrauben verstellen.

Die Spiegel, die normalerweise bei einem Michelson-Interferometer zu finden sind, sind hier durch Katzenaugenprismen ersetzt. Diese haben die Eigenschaft, dass sie Strahlen genau dahin reflektieren, wo diese hergekommen sind. Das erleichtert die Einstellung des Interferometers gegenüber Spiegeln, ändert aber nichts an der Funktionsweise.

Ein Katzenaugenprisma ist fest verbaut. Es existiert jedoch eine Vorrichtung, um dieses Prisma mithilfe eines Piezos zu bewegen, um Phasenprobleme mit der alten Geschwindigkeitsauswertung, verursacht durch zu konsistenten Bewegungen, zu beseitigen. Diese Vorrichtung wird jedoch nicht verwendet.

Das andere Katzenaugenprisma sitzt auf dem inneren Zylinder des MVTs, der auch mit dem radioaktiven Präparat verbunden ist. Da sich dieses Prisma bewegt, ändert sich das Interferenzmuster entsprechend.

Die Aufnahme des Interferometersignals erfolgt über eine Photodiode (Modell BPX65), deren Signal über eine Transistorschaltung verstärkt wird. Die Schaltung dafür ist am Rand einer runden Platine untergebracht, die sich so drehen lässt, dass die Photodiode möglichst gut im kombinierten Strahl der beiden Prismen liegt.

Die genaue Dokumentation mit Schaltungen ist im Handbuch des MVC-450 zu finden.

### 3.1.4. Elektronischer Teil

Im ursprünglichen Versuchsaufbau läuft die Datenverarbeitung über viele analoge Baukästen. Diese analogen Baukästen liegen oft in der Form von sogenannten *Nuclear Instrumentation Module* (kurz NIM) Kassetten vor. Diese NIM-Kassetten sind hochkant in einem Rack untergebracht, wo sie mit Strom versorgt werden. Die Kassetten sind mit BNC-Kabeln oder proprietären Steckern verbunden. Im Folgenden wird kurz auf die verwendeten NIM-Kassetten und ihre Aufgaben eingegangen, sowie auf andere wichtige Module.

#### 3.1.4.1. DFG-1200

Als Spannungsgeber wird der *Digital Function Generator 1200* (kurz DFG oder DFG-1200) von WissEl verwendet. Dieser hat eine variable Anzahl an Schritten, die allerdings im Versuch fest auf 1024 (10 Bit) eingestellt wird.

Der DFG hat drei relevante Ausgänge, das analoge Spannungssignal und zwei digitale Signale (CHA und START), mit denen sich der Zählerwert bestimmen lässt.

START wird einmal gepulst am Anfang jedes Zyklus, CHA wird jedes mal gepulst, wenn der Zähler inkrementiert wird. Beide digitalen Signale haben ein logisches High von 5 V. Der DFG hat mehrere Modi für die Ausgangsspannung, wird jedoch ausschließlich im Modus der Dreiecksspannung betrieben. Die Frequenz ist über ein Potentiometer einstellbar und sollte so eingestellt werden, dass es der Resonanzfrequenz des MVT entspricht.

#### 3.1.4.2. MDU-1200

Die Ansteuerung des MVT erfolgt durch die *Mössbauer Drive Unit 1200* (kurz MDU-1200 oder MDU) von WissEl. Dieser wird an die Spulen des MVT angeschlossen. Die MDU nutzt die Messspule zum Regeln der Steuerspule.



Die MDU hat als Eingangssignal den analogen Ausgang des DFGs. Die MDU versucht, eine Geschwindigkeit einzustellen, die proportional zu diesem Signal ist. Am MDU befindet sich ein Potentiometer, mit dem man die maximale Geschwindigkeit einstellen kann. Im Versuch wird die maximale Geschwindigkeit aber nicht verstellt.

Die MDU verfügt auch über eine Geschwindigkeitsanzeige, mit dem Ausgang des MVC macht die MDU eine Berechnung der Geschwindigkeit für alle Kanäle (alle ab Kanal 3). Allerdings stimmt dieser Wert nicht mehr, da die Wellenlänge des Lasers für die Berechnung benötigt wird. Im MDU ist die alte Wellenlänge des HeNe-Lasers hinterlegt, die sich nach dem Umbau auf den Halbleiterlaser geändert hat, folglich stimmt der angezeigte Wert nicht mehr. Im Versuch wird die angezeigte Geschwindigkeit allerdings nicht verwendet.

#### 3.1.4.3. MVC-450

Neben dem Interferometer besteht der MVC-450 von WissEl auch aus einer NIM-Kassette, an die das Interferometer angeschlossen wird. Dort erfolgt eine vergleichsweise primitive Verarbeitung des Interferometersignals von der Photodiode. Für jeden Hell-Dunkel-Übergang (eine Periode im Interferometersignal) gibt der MVC-450 einen kurzen Puls mit 5 V Spannung aus. Damit die Pulse zuverlässig erzeugt werden, muss die Optik durch Stellschrauben so lange justiert werden, bis die Amplitude des Signals groß genug ist.

Diese Pulse lassen sich, wie die Pulse des Zählrohres, mit der Zählerkarte auswerten. Um eine Geschwindigkeitsreferenz zu haben, lässt sich einstellen, dass für die Zählerwerte 1 und 2 des DAC nicht die Pulse vom Interferometersignal ausgegeben werden, sondern von einem internen Oszillator mit fester Frequenz  $f_{\text{ref}}$  (im Versuch:  $f_{\text{ref}} = 100 \text{ kHz}$ ).

#### 3.1.4.4. Zählerkarte

Die an einen Windows-XP-Rechner angeschlossene Zählerkarte kann kontinuierlich für jeden Kanal des DFGs Pulse zählen. Um den aktuellen Zählerwert der DFGs zu erhalten, ist die Zählerkarte an die Signale CHA und START des DFGs angeschlossen.

Die zu zählenden Pulse sind, je nach Einstellung am MVC, die verarbeiteten Pulse des Zählrohrs, oder die Pulse, die der MVC aus dem Interferometersignal erzeugt.

Zur Auswertung werden am Versuch fünf Messungen gemacht; eine zur Kalibrierung der Geschwindigkeit, und vier Messungen mit den vier zu untersuchenden Proben, die als Absorber über dem Zählrohr eingebaut werden. Die Messungen lassen sich als txt-Datei abspeichern und z.B. mit Python weiterverarbeiten.

Die Zählerkarte hat auch einen ADC-Modus, der im Versuch aber nicht oder nur am Rand benutzt wird.

#### 3.1.4.5. Signalkette Zählrohr

Die Signalkette des Signals am Zählrohr ist vergleichsweise lang. Eine vollständige Beschreibung ist in [8, S.12 f.] zu finden.

Fundamental wichtig in der alten Signalverarbeitungskette ist, dass das Signal des Zählrohrs über einen Diskriminator verarbeitet wird. Die Peaks im rohen Signal sind abhängig von der Energie, die das detektierte Photon hatte. Der Diskriminator muss so eingestellt werden, dass nur die Events mit einer Energie um 14,4 keV durchgelassen werden. Das ist notwendig, weil neben diesen Photonen noch Photonen anderer Energie auftreten, die z.B. beim Zerfall des  $^{57}\text{Co}$  frei werden. Allerdings findet Resonanzabsorption nur bei den 14,4 keV-Photonen des Eisenübergangs statt. Deshalb werden über den Diskriminator diese Events selektiert.

Der Diskriminator lässt sich durch Ausprobieren einstellen; wenn die Resonanzabsorption in der Messrate schnell zu erkennen ist, ist der Diskriminator gut eingestellt. Zum Einstellen des Diskriminators kann man sich auch das sog. Pulshöhenspektrum anschauen, dieses ist ein Histogramm, das darstellt, welche Energie mit welcher Häufigkeit gemessen wurde. Im Versuch lässt sich das Pulshöhenspektrum mit dem ADC-Modus der Zählerkarte messen.

### 3.2. Bearbeiteter Versuch mit FPGA-Trapezfilter

Im Rahmen einer anderen Bachelorarbeit wurde der Teil des Versuches überarbeitet, der mit der Verarbeitung des Zählsignals zusammenhängt, dies ist in [8] dokumentiert. Die alte Signalkette des Zählrohrs wurde dabei komplett verworfen und mit moderner Datenanalyse ersetzt.

Der neue Kern des Versuchs ist ein sogenannter *Red Pitaya*, Modell *STEMlab 125-14*, ein FPGA-Entwicklungsboard, welches mit je zwei hochauflösenden 14-Bit-ADCs und 14-Bit-DACs ausgestattet ist. Die ADCs und DACs haben einen Taktzyklus von 8 ns. Der Red Pitaya hat zusätzlich zum FPGA noch einen Arm-Prozessor, der auf den FPGA zugreifen kann. Auf dem ARM-Prozessor läuft eine schmale Linux-Distribution.

Das Signal des Zählrohrs wird direkt an einen der ADC-Eingänge des Red Pitaya eingesteckt. Auf dem Red Pitaya läuft auf dem FPGA ein Trapezfilter, der die exponentiellen Pulse vom Zählrohr in Trapeze umrechnet. Die Höhe der Trapeze ist proportional zur Energie, die das detektierte Photon hatte. Mit einer Reihe an Skripten werden die gemessenen Daten auf dem Linux-System des Red Pitayas vom FPGA ausgelesen und übers Netzwerk versendet, wo sie empfangen werden.

Zur Nutzung dieses Systems wird ein Python-Programm namens *mbclient* verwendet. Dieses Programm kann auf einem beliebigen Rechner laufen gelassen werden. Dem Programm müssen einige Parameter übergeben werden, u.a. wichtige Parameter für den Trapezfilter. Das Programm kümmert sich dann automatisch um den Verbindungsaufbau zum Red Pitaya, das Empfangen und Speichern der Daten.

In der Analyse liegen dann alle Events vor, die detektiert wurden, mit dem Zeitpunkt, dem aktuellen Zählerwert des DFGs, und der Höhe des Trapezes, welche proportional zur Energie ist. Damit lässt sich ein Filtern der Events wie am Diskriminator später in der Auswertung realisieren. Die sonstige Auswertung, insbesondere die Kalibrierung der Geschwindigkeit, wurde allerdings nicht verändert.

Zum Zeitpunkt dieser Bachelorarbeit gibt es allerdings Probleme mit dem Versuch, es ist fast unmöglich, brauchbare Messungen aufzunehmen. Die letzte große Änderung am Versuch war, dass das radioaktive Präparat ausgetauscht wurde, zuvor hat der Versuch sowohl im Original funktioniert, als auch in der bearbeiteten Version.

Beim bearbeiteten Versuch ist es insbesondere mit der am Versuch vorhandenen Expertise nicht möglich, Parameter für den Trapezfilter zu finden, die ähnlich gute Resultate liefern, wie Messungen vor dem Wechsel des radioaktiven Präparats.

Der Originalversuch konnte, wenn auch mit weniger guten Resultaten als vor dem Wechsel des radioaktiven Präparates, wieder zum Laufen gebracht werden. Aus diesem Grund wurden im Rahmen dieser Bachelorarbeit alle Messungen mit dem alten Messsystem durchgeführt.

## 4. Alternative Geschwindigkeitsmessung

### 4.1. Ziel der alternativen Geschwindigkeitsmessung

Im Rahmen dieser Bachelorarbeit soll eine alternative Geschwindigkeitsmessung am Mößbauer-Versuch entwickelt werden. Im Idealfall ist diese Geschwindigkeitsmessung FPGA-freundlich, sodass sich die Geschwindigkeitsmessung zusammen mit der bereits geschriebenen Software für die Auswertung des Zählrohrsignals auf dem Red Pitaya ausführen lässt.

Da es ohne vorher bekannten Algorithmus nicht möglich ist, etwas für den FPGA zu programmieren, befasst sich diese Arbeit damit, unterschiedliche Algorithmen zu testen, um einen Algorithmus zu finden, der für die Messung der Geschwindigkeit geeignet ist.

Im besten Fall soll eine Live-Messung der Geschwindigkeit möglich sein. Das bedeutet, dass, während Werte des analogen Interferometersignals einlaufen, kontinuierlich eine Umrechnung stattfindet, die aus den vorhandenen Werten die aktuelle Geschwindigkeit berechnet. Die so berechneten Werte für die Geschwindigkeit könnte man anschließend an die Events anhängen, die die Trapezfilter-Software versendet. Aktuell ergänzt diese Software die Events mit der Zählernummer des DFGs, um die Geschwindigkeitsinformation zu erhalten; es würde dann Sinn machen, die Zählernummer durch die Geschwindigkeit aus der Live-Messung zu ersetzen. So könnte man komplett auf die DFG-Kanäle verzichten und sogar den DFG auf den FPGA packen. Der Red Pitaya lässt sich mit quelloffener Software bereits als Funktionsgenerator nutzen. Da die DAC-Ausgänge noch frei sind, sollte es kein Problem sein, den DFG auch auf den Red Pitaya zu portieren. So lässt sich der Versuch Schritt für Schritt modernisieren und wandert von NIM-Kassetten auf den FPGA.

### 4.2. Beschreibung der Testmethode

Zum Testen der Algorithmen wird ein Testsignal benötigt, welches mit einem handelsüblichen digitalen Oszilloskop (Picoscope Modell 3204D, im Folgenden stets als Picoscope bezeichnet) aufgezeichnet wird. Dabei wird eine Abtastzeit von 8 ns verwendet. Dies entspricht der Abtastzeit des ADC am Red Pitaya.

Das verwendete Oszilloskop hat eine Auflösung von 8 Bit, was deutlich unter der 14-Bit-Auflösung des Red Pitayas liegt, allerdings ist diese Auflösung trotzdem ausreichend, um das Signal adäquat aufzuzeichnen. Das Picoscope wird im DC-Modus betrieben, was dem direkten Anschließen an einen ADC des Red Pitayas entspricht.

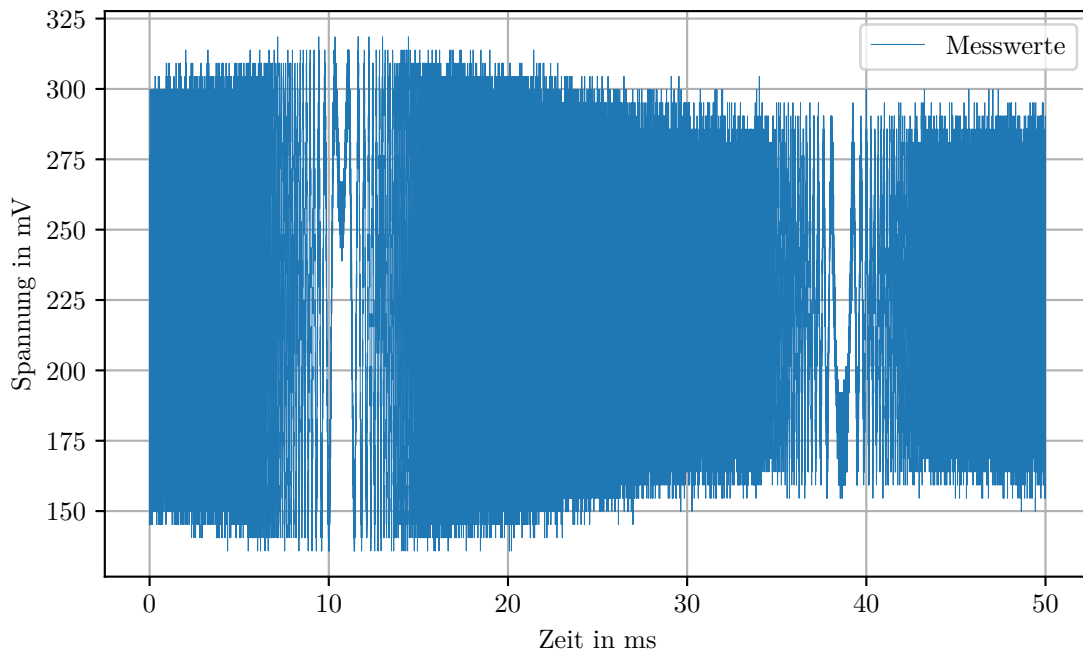
Zum Testen wurde rund ein voller Durchlauf des MVTs aufgezeichnet, was einem Durchlauf des DFGs entspricht. Im folgenden Absatz 4.3 wird das so gemessene Signal beschrieben.

### 4.3. Beschreibung des Signals

Bei dem gemessenen Signal handelt es sich um ein Chirp-Signal, wie es theoretisch in Kapitel 2.2.2 hergeleitet wird. Der Signalverlauf der Spannung  $U(t)$  hat also die Form aus Gleichung 2.18,

$$U(t) = U_0 \cdot \sin(2\pi f(v(t))t + \phi) + \langle U \rangle \stackrel{\text{lin. Abschnitt}}{=} U_0 \cdot \sin(2\pi\alpha \cdot t^2 + \phi) + \langle U \rangle, \quad (4.1)$$

mit der Amplitude  $U_0$ , der veränderlichen Frequenz  $f(v(t))$  und dem Phasensprung  $\phi$ . Das aufgezeichnete Signal ist in Abbildung 4.1 zu sehen. Das Signal wurde mehrfach aufgezeichnet, allerdings wurden die Verfahren alle mit diesem Signal getestet, da die anderen Aufzeichnungen sich nicht wesentlich unterscheiden. Die Aufzeichnung dauert 50 ms und besteht aus 6,25 Millionen Messwerten, was mit der gewählten Sampling-Zeit von 8 ns übereinstimmt.



**Abbildung 4.1.:** Verlauf des aufgezeichneten Signals für rund eine Schwingung der Membran

Der verschobene Mittelwert des Signals  $\langle U \rangle$  aus Gleichung 4.1 liegt bei rund  $\langle U \rangle \approx 225$  mV. Die zu bestimmende Frequenz  $f(v(t))$  macht einen in guter Näherung linearen Verlauf von  $f_{\min} = 0$  Hz an den Wendepunkten bis  $f_{\max} \approx 40$  kHz an der schnellsten Stelle. An den Wendepunkten ist die Frequenz  $f = 0$  und wird anschließend gespiegelt.

Der Phasensprung  $\phi$  ist bei jedem Wenden etwas unterschiedlich, was bedeutet, dass die Bewegung nicht perfekt identisch ist, sondern, dass es hier kleine Abweichungen gibt, und der Zylinder etwas früher oder später wendet. Mit dem Phasensprung wird das Aussehen am Wendepunkt beschrieben, hier können sich unterschiedliche Bilder ergeben.

Die Amplitude  $U_0$  ist im Mittel rund  $U_0 \approx 75$  mV. Eine weitere interessante Beobachtung am echten Signal ist, dass die Amplitude  $U_0$  nicht konstant ist, sondern vom Abstand des Zylinders abhängig ist.

Zuletzt lässt sich noch Rauschen auf dem Signal beobachten. Dabei handelt es sich teilweise um echtes Signalaruschen, teilweise lässt sich auch digitales Rauschen beobachten, was auf die vergleichsweise geringe 8-Bit-Auflösung des Picoscopes zurückzuführen ist.

#### 4.4. Bekannte Probleme

Bei der direkten Bestimmung der Geschwindigkeit aus dem Interferometersignal gibt es zwei größere Probleme, die bereits im Voraus bekannt sind.

Zum Einen lässt sich aus dem Interferometersignal nicht das Vorzeichen der Bewegung ablesen. Eine Bewegung nach oben oder nach unten sieht am Interferometer gleich aus, da

der Strahlengang derselbe ist. Zwar lässt sich, wie in Absatz 4.3 beschrieben, eine Amplitudenmodulation je nach Lage des Zylinders beobachten, allerdings ist diese Modulation sehr schwach, und von der genauen Einstellung am Interferometer abhängig, weshalb es nicht möglich ist, daraus zuverlässig das Vorzeichen der Geschwindigkeit zu bestimmen. Um an das Vorzeichen der Geschwindigkeit zu kommen, muss man also zwangsläufig das Vorzeichen des DFGs betrachten.

Das zweite große Problem ist die Geschwindigkeitsmessung an den Wendepunkten (um  $v = 0$ ), da hier die Frequenz  $f$  ebenfalls sehr klein (bis  $f = 0$ ) ist, was sich mit vielen Verfahren nicht mehr gut messen lässt. Im Folgenden wird für jedes getestete Verfahren auf Probleme in dieser Umgebung eingegangen.

## 4.5. Getestete Verfahren

### 4.5.1. Peakfinding

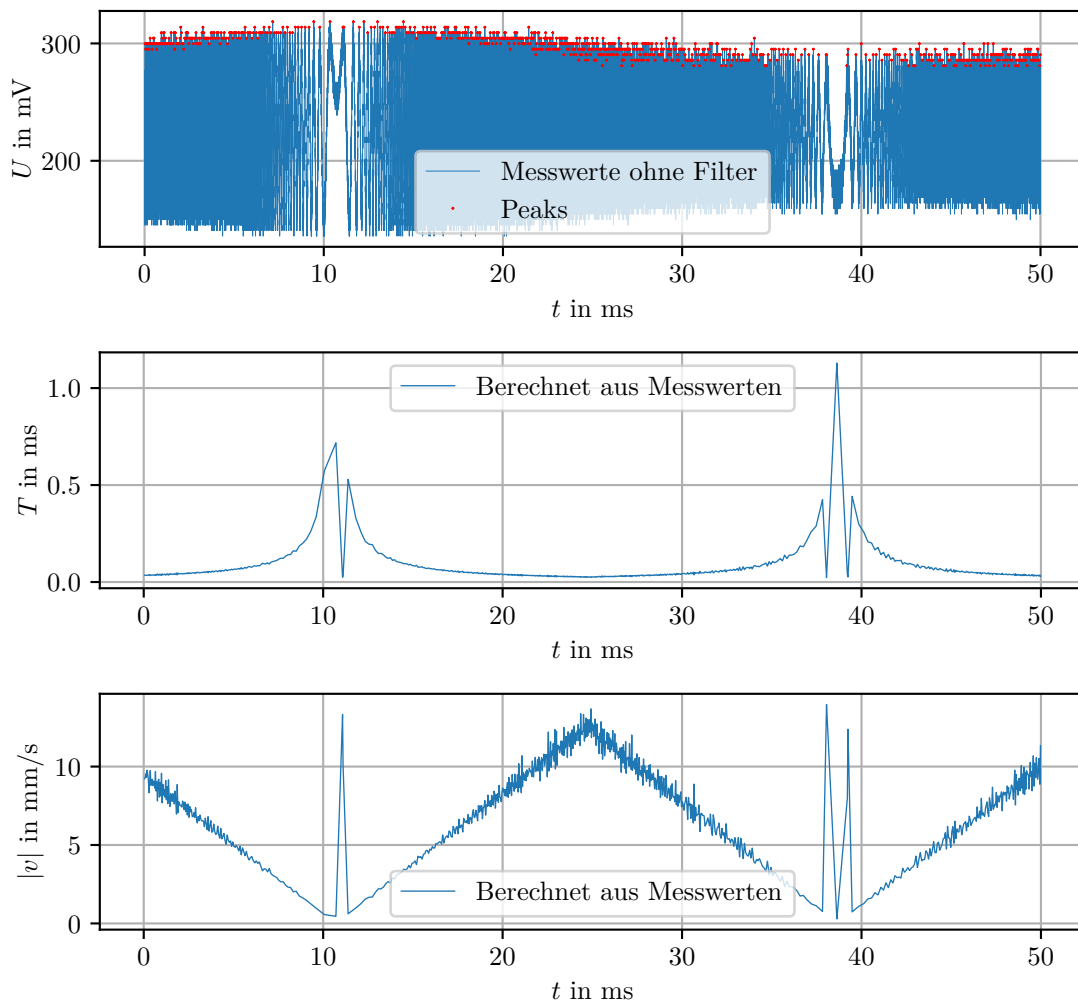
#### 4.5.1.1. Allgemeine Beschreibung und Idee des Verfahrens

Ein sehr simples Verfahren zur Frequenzbestimmung ist es, die Peaks im Signal zu detektieren. Diese Peaks haben einen zeitlichen Abstand, der der aktuellen Periodendauer  $T$  entspricht, die über  $T = f^{-1}$  mit der aktuellen Frequenz zusammenhängt. Aus der Frequenz  $f$  lässt sich die Geschwindigkeit  $v$  berechnen durch  $v = \lambda/2 \cdot f$ .

Die berechnete Geschwindigkeit gilt am besten mittig zwischen den Peaks, allerdings wird angenommen, dass die so bestimmte Geschwindigkeit im kompletten Intervall zwischen den Peaks gilt, da es sonst mit diesem Verfahren nicht möglich ist, dort eine Geschwindigkeit zuzuordnen.

#### 4.5.1.2. Spezifische Beschreibung

Zum Testen mit Python wurde wegen des starken Rauschens im Signal der fortgeschrittene Peakfinder aus der Python-Library *scipy.signal* [9] verwendet. Trotz des fortgeschrittenen Peakfinders ist die Detektion nicht komplett zuverlässig.



**Abbildung 4.2.:** Geschwindigkeitsmessung über Peakfinding ohne Mittelwertfilter

In Abbildung 4.2 ist die Berechnung der Geschwindigkeit nach Peakfinding zu sehen. Im obersten Plot ist der Signalverlauf des Chirp-Signals mit den detektierten Peaks zu sehen.

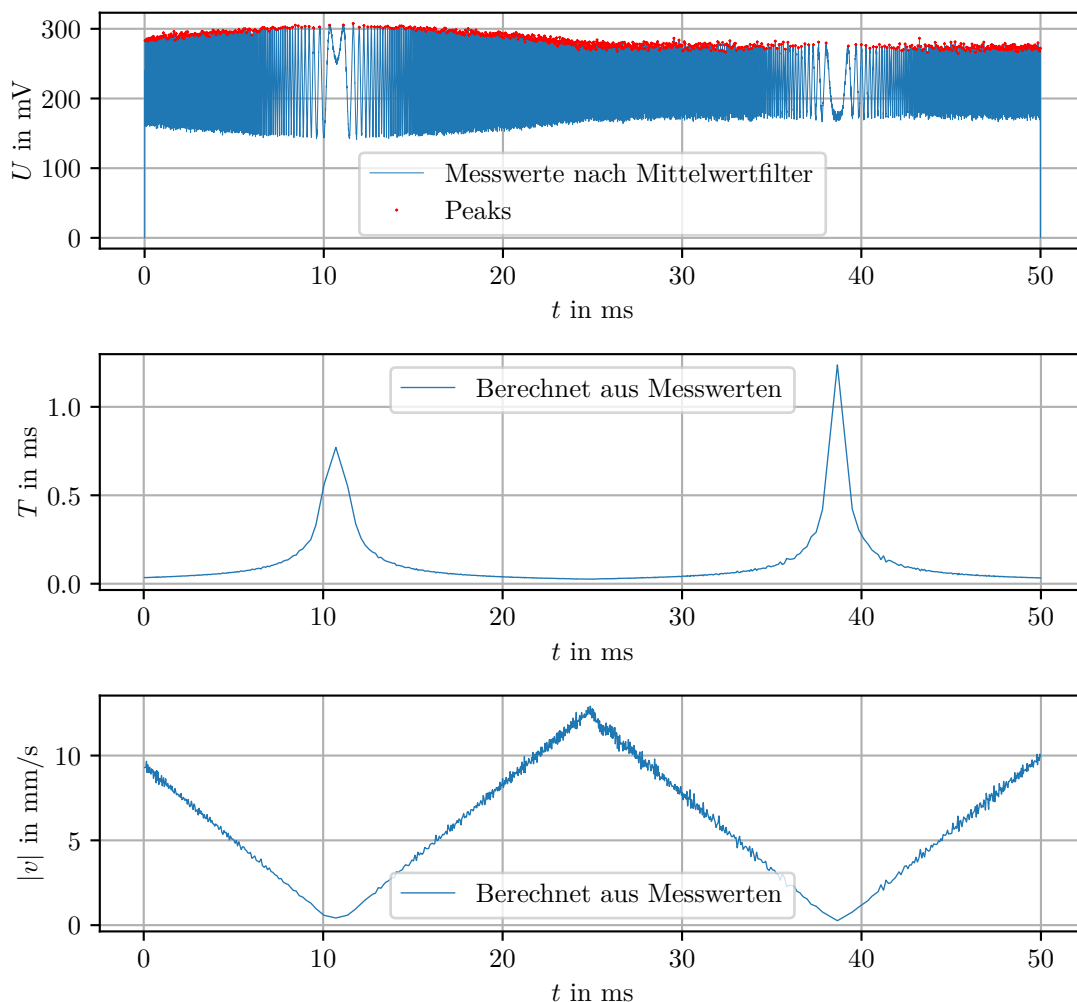
In mittleren Plot ist die Periodendauer zu sehen, die aus dem Abstand der Peaks berechnet wird. Im untersten Plot ist der Verlauf der Geschwindigkeit (genauer: der Betrag der Geschwindigkeit) zu sehen.

Der grobe Verlauf der Geschwindigkeit passt zur Erwartung, allerdings ist der Geschwindigkeitsverlauf sehr stark verrauscht. Zudem sind drei Ausreißer in der Nähe der Wendepunkte zu sehen; an diesen Stellen hat der schon sehr gut eingestellte Peakfinder wegen sehr starkem Rauschen doppelt ausgelöst.

#### 4.5.1.3. Modifikation: Mittelwertfilter

Abhilfe für das sehr starke Rauschen im Signal der Geschwindigkeit schafft ein gleitender Mittelwertfilter, der im Rohsignal alle  $n$  Werte im Rohsignal aufaddiert und durch  $n$  dividiert. So wird das Rauschen im Ausgangssignal deutlich reduziert. Der neue Verlauf mit dem Mittelwertfilter von  $n = 1000$  ist in Abbildung 4.3 zu sehen.

Das Rauschen im finalen Geschwindigkeitsverlauf ist ebenfalls reduziert, allerdings immer noch vergleichsweise stark. Zumindest die zuvor vorhandenen Fehldetektionen fallen weg, da das Rauschen im Ausgangssignal schwächer geworden ist.



**Abbildung 4.3.:** Geschwindigkeitsmessung über Peakfinding mit Mittelwertfilter

#### 4.5.1.4. Fazit

Beim Peakfinding handelt es sich um ein vergleichsweise sehr simples Verfahren, welches ohne aufwendigere Berechnungen eine Geschwindigkeit liefert. Für eine Live-Messung der

Geschwindigkeit ist es aufgrund des starken Rauschens im Ergebnis allerdings nicht so gut geeignet. In der Umgebung um die Wendepunkte der Oszillation ist das Verhalten nicht so klar, hier kann es sein, dass die Phase einen sehr starken Einfluss hat und so eine falsche Geschwindigkeit berechnet wird. Ist die Phase gerade so, dass der Wendepunkt in einem kleinen Minimum liegt und direkt neben dem Wendepunkt sind Maxima auf beiden Seiten (wie im untersuchten Signal aus Abbildung 4.1 bei ca. 12 ms, nur noch extremer), dann wird ein geringer Abstand detektiert, der aber in diesem Fall nicht die Periodendauer ist; die berechnete Geschwindigkeit fällt dann viel zu groß aus.

Umgekehrt kann es auch sein, dass für eine lange Zeit am Wendepunkt keine Peaks gefunden werden. Dann leidet die Auflösung der Geschwindigkeit, da Events von einem sehr großen Zeitraum alle dieselbe Geschwindigkeit zugeordnet bekommen.

Zusammenfassend lässt sich sagen, dass Peakfinding grundsätzlich geeignet ist, allerdings insbesondere an den Wendepunkten große Schwächen hat und auch trotz Mittelwertfilter noch sehr verrauscht ist.



## 4.5.2. FFT

### 4.5.2.1. Allgemeine Beschreibung und Idee des Verfahrens

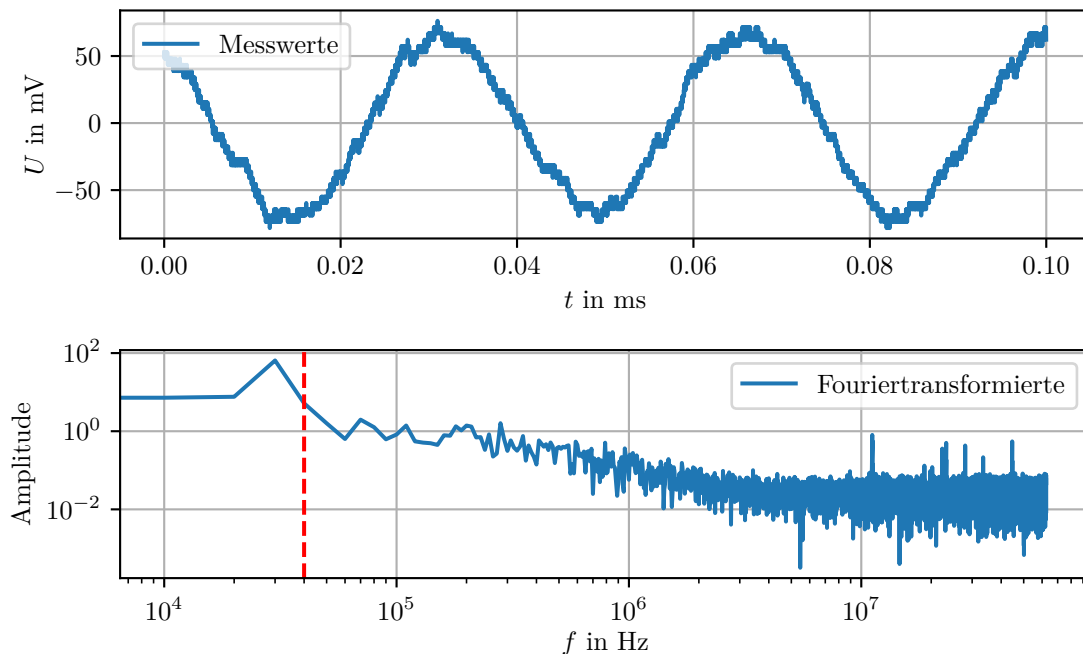
*Fast Fourier Transform*, kurz *FFT*, ist ein Standardverfahren zur Zerlegung eines Signals in seine Frequenzen. Die FFT leistet, dass ein diskretes Signal, wie es hier vorliegt, in sein Frequenzspektrum zerlegt wird. Dadurch wird allen relevanten Frequenzen eine Amplitude zugeordnet. Die Frequenz, bei der die Amplitude maximal ist, ist die dominante Frequenz. Die Idee dieses Verfahrens ist es, das Signal des Interferometers in Fenster (auch *Frame* genannt) fester Länge zu zerlegen. Für jedes Fenster wird die Frequenz als konstant angenommen, und die Fouriertransformierte wird berechnet. Die dominante Frequenz wird dann der Mitte des Frames zugeordnet.

FFT wird unter anderem deshalb betrachtet, weil fertige, sogenannte *Intellectual Property*-Blöcke, kurz *IP*-Blöcke, existieren, die direkt auf dem FPGA eingesetzt werden können und schnell und effizient die FFT berechnen.

### 4.5.2.2. Spezifische Beschreibung

Das untersuchte Signal ist in Abbildung 4.1 zu sehen. Davon wurden die ersten 0,1 ms als Frame genommen. Im Frame wird die Frequenz als konstant angenommen. Das Frame ist der obere Plot in Abbildung 4.4. Dabei wurde die Spannung noch um den Mittelwert nach unten verschoben, sodass die resultierende Oszillation um 0 V stattfindet.

Grob lässt sich die zu erwartende dominante Frequenz darüber abschätzen, dass in dem Frame mit Länge 0,1 ms etwa 3 Perioden enthalten sind. Die Periodendauer  $T$  ist also rund  $T \approx 0,033$  ms, der Kehrwert davon ist die Frequenz  $f = 30$  kHz. Die FFT sollte also eine dominante Frequenz von rund  $f \approx 30$  kHz liefern.



**Abbildung 4.4.:** Signalframe und zugehörige Fouriertransformierte

Mit der Funktion *Fourier\_fft()* der Python-Library *PhyPraKit* [10] wird die Fouriertransformierte des Signals berechnet. Die so berechnete Fouriertransformierte ist in Abbildung 4.4 im unteren Plot in doppelt-logarithmischer Darstellung zu sehen. Die doppelt-logarithmische Darstellung wurde zur besseren Übersichtlichkeit gewählt.

Der für das Interferometersignal relevante Frequenzbereich ist grob von 0 bis 40 kHz. Die

höchste auftretende Frequenz von 40 kHz ist im unteren Plot von Abbildung 4.4 mit einer vertikalen roten, gestrichelten Linie markiert. Daneben ist das Maximum der FFT, die dominante Frequenz zu sehen. Diese liegt, wie nach Erwartung, bei genau 30 kHz. Allerdings ist die Auflösung der FFT im relevanten Frequenzbereich sehr gering, was sich auch im Plot am eckigen Verlauf sehen lässt. Tatsächlich gibt es im relevanten Frequenzbereich von 0 bis 40 kHz nur fünf Stützstellen, für die die FFT eine Amplitude berechnet, diese liegen bei 0 Hz, aufwärts in Schritten von 10 kHz bis 40 kHz. Von diesen fünf Stellen, an denen die FFT eine relevante Amplitude liefert, ist die Stelle mit  $f = 0$  zudem nicht brauchbar, da diese die Information über die Phase für  $t = 0$  enthält. Allerdings ist die Phase hier nicht relevant, nur die Frequenz. Somit bleiben vier Stützstellen, anhand derer die Frequenz des Signals und daraus die Geschwindigkeit des Emitters bestimmt werden soll. Diese geringe Auflösung in der Frequenz macht eine genaue Geschwindigkeitsmessung über FFT unmöglich.

#### 4.5.2.3. Fazit

Grundsätzlich ist das Verfahren, über die FFT die dominante Frequenz zu bestimmen für die Frequenzmessung geeignet. Insbesondere bei höheren Frequenzen funktioniert eine Analyse über FFT sehr gut, wie sich an Abbildung 4.4 im unteren Plot sehen lässt. Um diese Auflösung zu erreichen, werden allerdings deutlich mehr als drei Perioden in einem Frame benötigt, was hier nicht realistisch ist, da sich aufgrund des Chirp-Signals die Frequenz laufend ändert und dann im Frame nicht mehr konstant ist, sodass große Unterschiede auftreten können.

Wegen der geringen Frequenzauflösung im relevanten Bereich ist FFT leider nicht geeignet, weshalb der Ansatz nicht weiter verfolgt wird.

Auch an den Wendepunkten mit  $f = 0$  versagt das FFT-Verfahren, weil die Amplitude der Fouriertransformierten für  $f = 0$  keine relevante Aussagekraft hat.

### 4.5.3. Autokorrelation

#### 4.5.3.1. Allgemeine Beschreibung und Idee des Verfahrens

Für die Geschwindigkeitsmessung wird noch ein eigenes Verfahren entwickelt und getestet, welches auf Autokorrelation basiert.

Dafür wird das Signal wieder in Fenster aufgeteilt, wie beim FFT-Verfahren. Für jedes Fenster wird die Autokorrelationsfunktion  $\rho(t)$  bestimmt, wobei in jedem Fenster die Zeit bei  $t_{\text{Frame}} = 0$  anfängt. Von dieser Autokorrelationsfunktion wird das *erste Minimum* gesucht. Im Fenster wird die Frequenz  $f$  und die Periodendauer  $T = 1/f$  als konstant angenommen, die bestimmte Frequenz kann dann der Mitte des Fensters zugeordnet werden. Das erste Minimum sollte immer an der Stelle  $t_{\text{Frame}} = T/2$  zu finden sein. Aus der Lage des ersten Minimums sollte sich dann die Periodendauer bestimmen lassen, daraus dann die Frequenz und mit der Frequenz die gesuchte Geschwindigkeit.

Die Theorie hinter der Autokorrelation ist in Kapitel 2.3 zu finden. Da hier auch näherungsweise periodische Signale analysiert werden, ist das Verfahren sehr ähnlich zu Abbildung 2.9, dort ist die Autokorrelation für Sinus und Cosinus zu sehen.

Es kann vorkommen, dass kein Minimum gefunden wird. Insbesondere an den Wendepunkten, wo die Frequenz sehr klein wird, kann das der Fall sein. Sobald die Periodendauer des Signals länger als die Länge des Frames wird, gibt es Probleme. Ist die Periodendauer deutlich länger, also bei sehr kleinen Frequenzen, ähnelt das Signal im Frame entweder einer Konstanten (Bauch einer langsamen Schwingung) oder einer Gerade mit Nulldurchgang (Knoten einer langsamen Schwingung), mit gemischten Fällen dazwischen. Der Verlauf der Autokorrelation für diese Fälle ist in den Abbildungen 2.7 und 2.8 zu sehen.

#### 4.5.3.2. Spezifische Beschreibung

Zunächst wird dieselbe Fenstergröße wie bei dem Verfahren über FFT verwendet, die Fenster haben also eine Länge von 0,1 ms.

Zur Berechnung der Autokorrelationsfunktion wird die Funktion `autocorrelate()` der Python-Library `PhyPraKit` [10] verwendet.

In einem ersten Schritt wird für jedes Fenster die Autokorrelationsfunktion bestimmt. In einem zweiten Schritt wird das Minimum gesucht und es finden Berechnungen mit dem gefundenen Minimum statt.

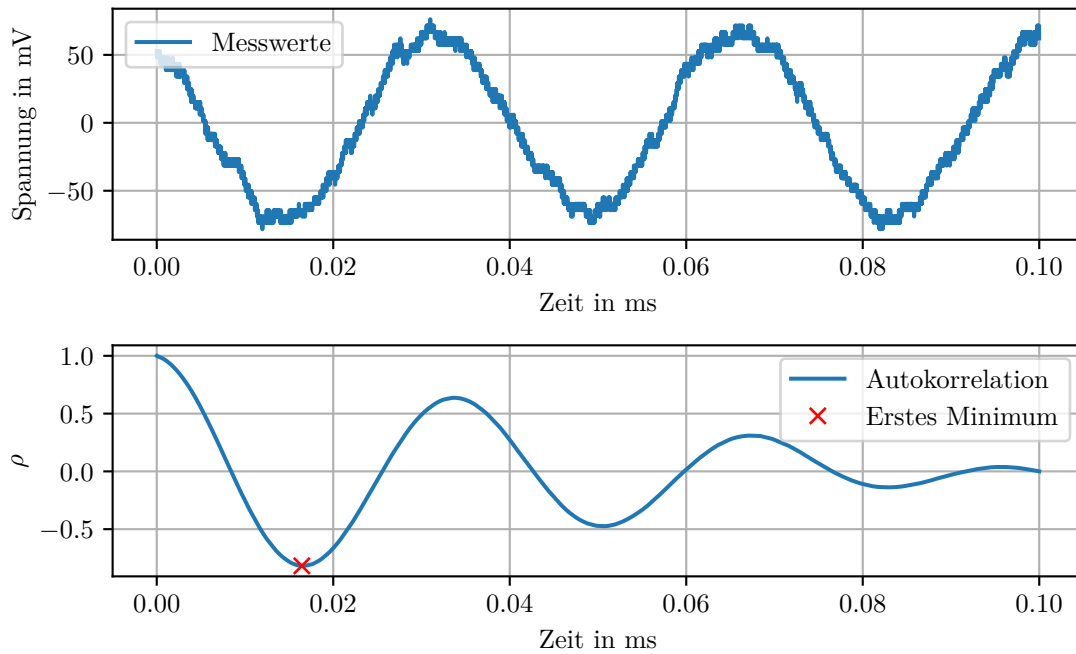
Das erste Minimum der Autokorrelationsfunktion wird gesucht, indem von  $t = 0$  nach rechts durchlaufend getestet wird, welcher Wert kleiner oder gleich groß wie seine nächsten  $n$  Nachbarn in beide Richtungen ist. Es wird durchgehend  $n = 100$  verwendet. Da das erste Minimum auch immer der global kleinste Wert ist (alle anderen Minima haben eine kleinere Amplitude) könnte auch einfach das globale Minimum der Autokorrelation genommen werden, allerdings ist ein Trigger, wie er hier verwendet wird, näher an einer möglichen FPGA-Implementierung.

Wird kein Minimum gefunden, wird die Periodendauer auf  $T = 1$  ms gesetzt, was das Zehnfache der Fensterlänge ist. In diesem Fall lässt sich schlicht keine Aussage machen, weil das Verfahren versagt.

Das Frame mit Signal ist in Abbildung 4.5 im oberen Plot zu sehen. Die dazugehörige Autokorrelations-Funktion mit Minimum sind im unteren Plot zu sehen.

Das erste Minimum der Autokorrelation liegt am Index 2055. Mit einer Schrittweite von 8 ns und einer Verdopplung (erstes Minimum liegt bei  $T/2$ ) ergibt sich die Periodendauer  $T$  im Frame zu  $T = 0,03288$  ms. Das entspricht einer Frequenz  $f$  von 30,414 kHz. Im vorherigen Kapitel 4.5.2 zur FFT wurde die Frequenz grob auf 30 kHz bestimmt. Das Resultat deckt sich also mit dieser Hochrechnung.

Im Gegensatz zur FFT ist allerdings die Frequenzauflösung deutlich höher. Die halbe



**Abbildung 4.5.:** Erstes Signalframe und zugehörige Autokorrelation mit erstem Minimum

Periodendauer (= erstes Minimum der Autokorrelation) wird bestenfalls mit der verwendeten Sample Zeit aufgelöst, also hier 8 ns. In der Praxis ist die Auflösung schlechter durch diverse Probleme, die in den kommenden Abschnitten behandelt werden, aber immer noch deutlich besser als bei der FFT.

Das aufgezeichnete Signal wird in mehrere Frames zerlegt. Dabei wird das Signal so aufgeteilt, dass die Frames sich nicht überschneiden. Sobald ein Frame aufhört, fängt das nächste Frame an.

Die Berechnung der Periodendauer wird für jedes Frame gemacht, die so berechneten Periodendauern sind in Abbildung 4.6 im oberen Plot zu sehen. Die aus den Periodendauern berechneten Geschwindigkeiten sind in Abbildung 4.6 im unteren Plot zu sehen.

Es ist in Abbildung 4.6 zu erkennen, dass es auch hier noch einige Fehldetektionen gibt, die sich an großen Ausreißern zeigen. Durch diese ist das eigentliche Signal klein gestreckt. Der Verlauf stimmt allerdings mit der Erwartung überein, im Geschwindigkeitsverlauf lassen sich Dreiecke erkennen.

#### 4.5.3.3. Gegenkontrolle: Visualisierung der Autokorrelation

Es ist schwer, das Autokorrelationsverfahren nur am Endresultat zu verbessern, da schwer zu erkennen ist, was genau nicht so gut funktioniert. Um nicht mehr oder weniger ‘blind’ zu fahren, wäre es schön, den Signalverlauf zu sehen.

Um das zu erreichen, wird der Signalverlauf und der Verlauf der Autokorrelation visualisiert. Dafür wird bei der Berechnung der Autokorrelation und des Minimums für jedes Frame ein Bild abgespeichert. Die Bilder sind ähnlich zu Abbildung 4.5, im oberen Plot ist der Signalverlauf zu sehen. Im unteren die Autokorrelation mit dem gefundenen ersten Minimum.

Zur besseren Visualisierung bekommt der untere Plot einen grünen Hintergrund, falls ein Minimum gefunden wird, und einen roten Hintergrund, wenn kein Minimum gefunden wird.

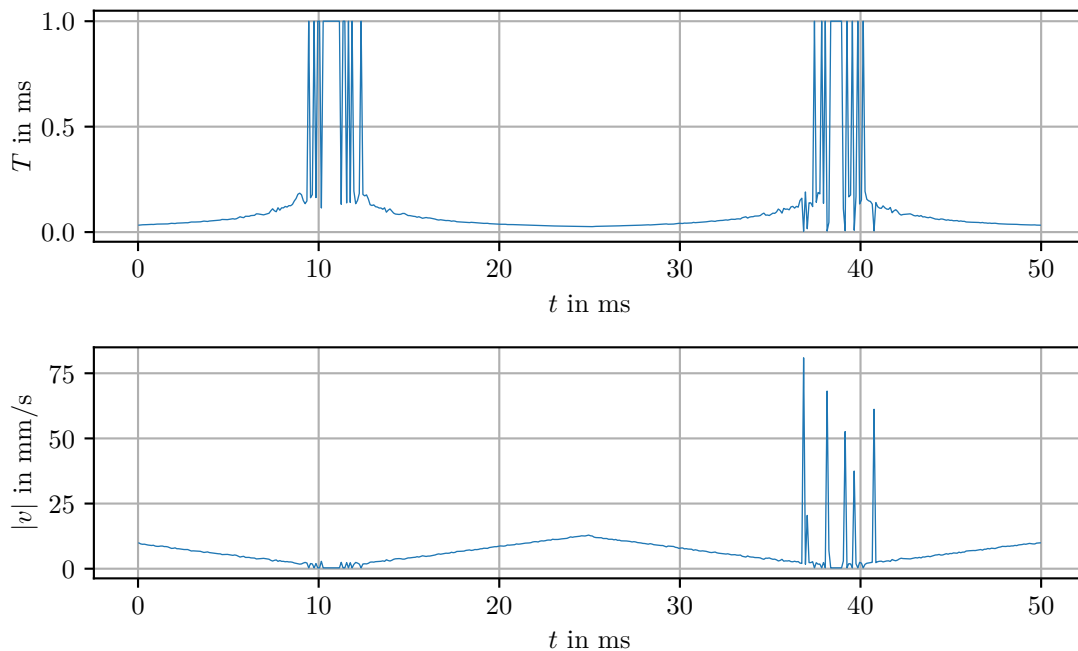


Abbildung 4.6.: Erste Autokorrelationsmessung

Mit dem Programm *FFmpeg* [11] werden hunderte dieser Einzelbilder zu einem Video zusammengerechnet. Mit diesen Videos lässt sich das Verfahren weiter verbessern und Probleme lassen sich erkennen.

Da sich Videos schlecht abdrucken lassen, wurden diese auf der Videoplatform *YouTube* hochgeladen. Das zu Abbildung 4.6 gehörige Video lässt sich unter <https://www.youtube.com/watch?v=gzW10kJYHYI> aufrufen.

Für alle weiteren Geschwindigkeitsmessungen via Autokorrelation wird eine URL angegeben, mit der das entsprechende Video angeschaut werden kann.

#### 4.5.3.4. Modifikation: Bedingung für die Autokorrelationsfunktion

Mit der erfolgten Visualisierung kann man das Verfahren weiter anpassen. Eine gute Bedingung, die ab sofort verwendet wird, ist, dass das erste Minimum der Autokorrelationsfunktion  $\rho$  kleiner als  $-0,5$  sein muss. Dadurch wird ein falsches Auslösen weitestgehend verhindert.

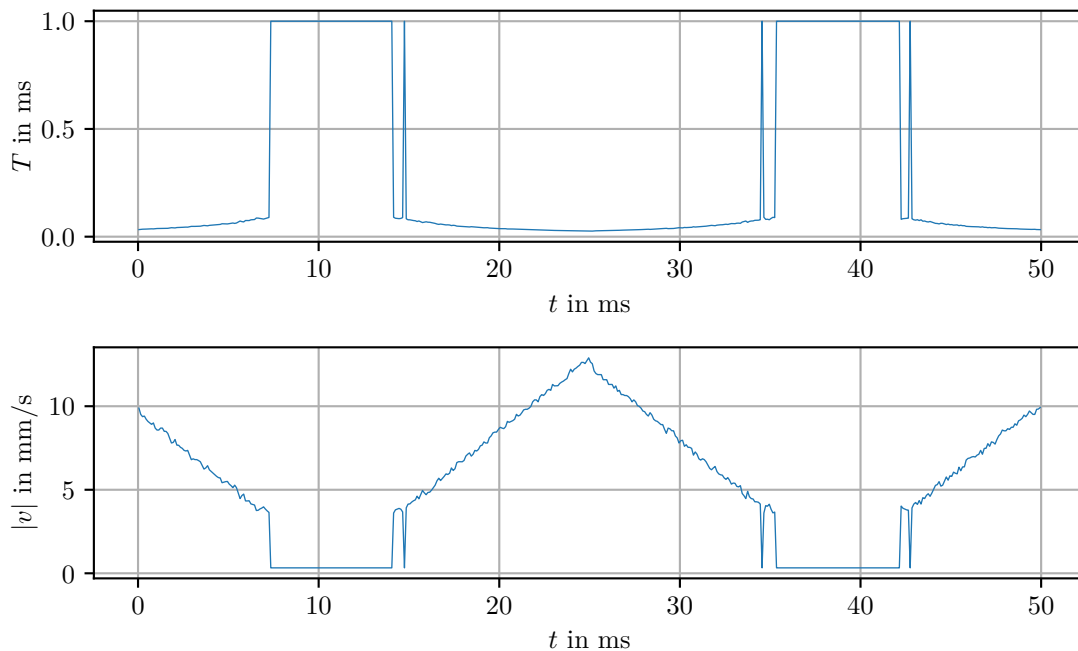
Mit dieser Bedingung ergibt sich Abbildung 4.7. Das Video zu Abbildung 4.7 findet sich unter <https://www.youtube.com/watch?v=K-YeHafXnSE>.

Es lässt sich in Abbildung 4.7 erkennen, dass das Verfahren grundsätzlich funktioniert, allerdings ist die Geschwindigkeit stark verrauscht. Für langsame Geschwindigkeiten versagt das Verfahren, wie es die relativ großen Balken bei  $t \approx 10$  ms und  $t \approx 40$  ms zeigen. Um diese Balken herum ist die Detektion eines Minimums nicht stabil, dort kann es sein, dass ein Minimum gefunden wird, oder es kann sein, dass kein Minimum gefunden wird.

#### 4.5.3.5. Modifikation: Gleitende Frames

Um das Rauschen genauer untersuchen zu können, werden die Frames mit mehr Zwischenschritten gerechnet. Bis jetzt hat jedes Frame die Länge 0,1 ms oder 12500 Samples, und jedes Frame ist zum nächsten Frame um dieselbe Zeit verschoben.

Zur genaueren Untersuchung werden die Frame-Länge und die Frame-Verschiebung ab



**Abbildung 4.7.:** Autokorrelationsmessung mit erweiterten Bedingungen

jetzt getrennt. Die Frame-Länge bleibt bei 0,1 ms oder 12500 Samples, aber die Frame-Verschiebung wird um einen Faktor 10 reduziert, auf 0,01 ms oder 1250 Samples. Die so erfolgte Berechnung ist in Abbildung 4.8 zu sehen. Das dazugehörige Video findet sich unter <https://www.youtube.com/watch?v=ulpvYEjlopK>

Es lässt sich klar erkennen, dass das Rauschen einer einhüllenden Schwingung folgt. Ein möglicher Grund könnte sein, dass dies ein Artefakt der Autokorrelation ist. Ein anderer möglicher Grund ist, dass der Emmitter, dessen Geschwindigkeit untersucht werden soll, so oszilliert. Schließlich wird ein reelles, oszillierendes physikalisches System betrachtet, bei dem derartige Schwingungen auftreten könnten.

Im Video lassen sich zudem Schwingungen der Autokorrelationsfunktion zwischen einzelnen Frames beobachten, mit dieser Schwingung muss die einhüllende Schwingung im Geschwindigkeitsplot zusammenhängen.

#### 4.5.3.6. Gegenkontrolle: Testsignal

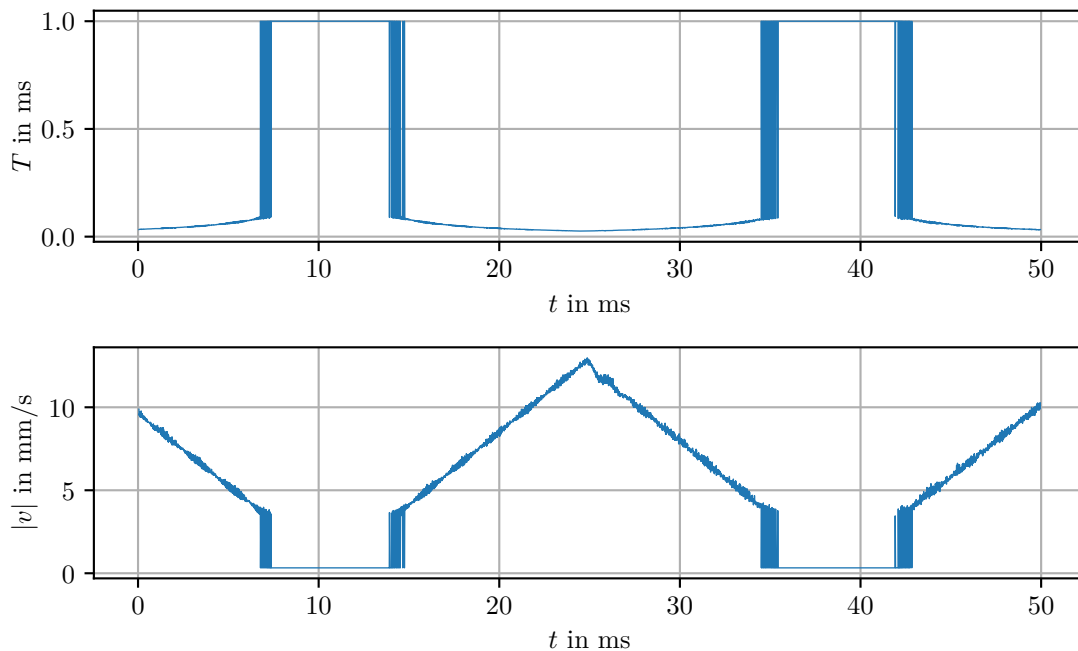
Um herauszufinden, ob die einhüllende Schwingung aus 4.8 ein Artefakt der Auswertung ist, oder wirklich am Experiment auftritt, wird ein Testsignal geschrieben, welches ähnliche Charakteristiken wie das gemessene Chirp-Signal hat.

Grundsätzlich besteht das Testsignal aus einem einzigen linearen Chirp, der in 14 ms von 0 auf 40 kHz läuft. Die Phase am Anfang wird auf  $\phi = 30^\circ$  gesetzt, was relativ willkürlich ist, aber dem ersten Phasensprung am echten Signal ähnelt.

Das so berechnete Chirp-Signal wird dann doppelt gespiegelt. Erstmals gespiegelt wird am am Start, bei  $f = 0$ . Diese Spiegelung ist physikalisch sinnvoll, da das Interferometer im Idealfall auch in beide Richtungen gleich läuft; dieser Übergang ist wegen  $f = 0$  auch in seiner Ableitung stetig.

Das resultierende Signal wird nochmal gespiegelt, aber am Ende mit maximaler Frequenz. Diese Spiegelung ist physikalisch nicht ganz korrekt, da der Übergang in der Ableitung nicht zwangsläufig stetig ist. Das ist auch hier der Fall, beim genauen Betrachten des Signals, z.B. im Video später, fällt dieser Sprung sofort auf.

Die Amplitude des Testsignals ist eigentlich nicht wichtig, da nur die Frequenz von Relevanz



**Abbildung 4.8.:** Autokorrelationsmessung mit gleitenden Frames am echten Signal

ist. Das Testsignal oszilliert auch um die Amplitude null, das ist wichtig. Das so erzeugte Testsignal ist in Abbildung 4.9 zu sehen.

Mit dem so bestimmten Testsignal wird exakt dieselbe Berechnung gemacht, wie zuvor mit dem echten Signal. Die dazugehörigen Plots sind in Abbildung 4.10 zu sehen. Das dazugehörige Video ist unter <https://www.youtube.com/watch?v=m7r9gmAdGkA> zu finden.

Zur Kontrolle ist als gestrichelte Linie der Verlauf der Geschwindigkeit eingezeichnet, der nach dem Chirp-Signal zu erwarten ist. Die gemessene Geschwindigkeit ist immer etwas über der erwarteten Geschwindigkeit, allerdings sind die Abweichungen relativ klein. Der Ausreißer in der Geschwindigkeit in der Mitte des Plots ist durch die Unstetigkeit der Ableitung, die beim Spiegeln des Chirps entsteht, zu erklären.

Es fällt auf, dass das Testsignal und das echte Signal nahezu denselben Geschwindigkeitsplot haben. Bei beiden ist eine einhüllende Schwingung zu erkennen. Folglich muss es sich um ein Artefakt der Autokorrelation handeln.

Im Rückblick auf die Theorie in Kapitel 2.3.1 wird das Problem vermutlich durch die wenigen Perioden verursacht, die im Frame sind. Es befinden sich maximal ca. 4 Perioden in einem Frame. Dadurch kommt es bei angeschnittenen Perioden, die hier durchaus auftreten, stark auf die Phase an, die im Frame ist. Vermutlich ist die Oszillation der Geschwindigkeit an einem Schwingungsbauch, wenn im Frame halbzahlige Schwingungen sind, da dann die Phase viel Spielraum hat. Ist jedoch rund eine ganze Anzahl an Schwingungen im Frame, spielt die Phase eher eine geringere Rolle, und die einhüllende Schwingung im Geschwindigkeitsplot ist an einem Schwingungsknoten.

#### 4.5.3.7. Modifikation: Längere Frames

Die größten Probleme des Autokorrelationsverfahren bis hierhin sind eine einhüllende Schwingung in der Frequenz, vermutlich bedingt durch angeschnittene Perioden, und eine große Messlücke für Geschwindigkeiten bzw. Frequenzen nahe  $v = f = 0$ .

Beide Probleme sollten sich simultan deutlich verbessern lassen, wenn die Framelänge vergrößert wird. Dann befinden sich im Normalfall mehr Perioden im Frame und das Frame

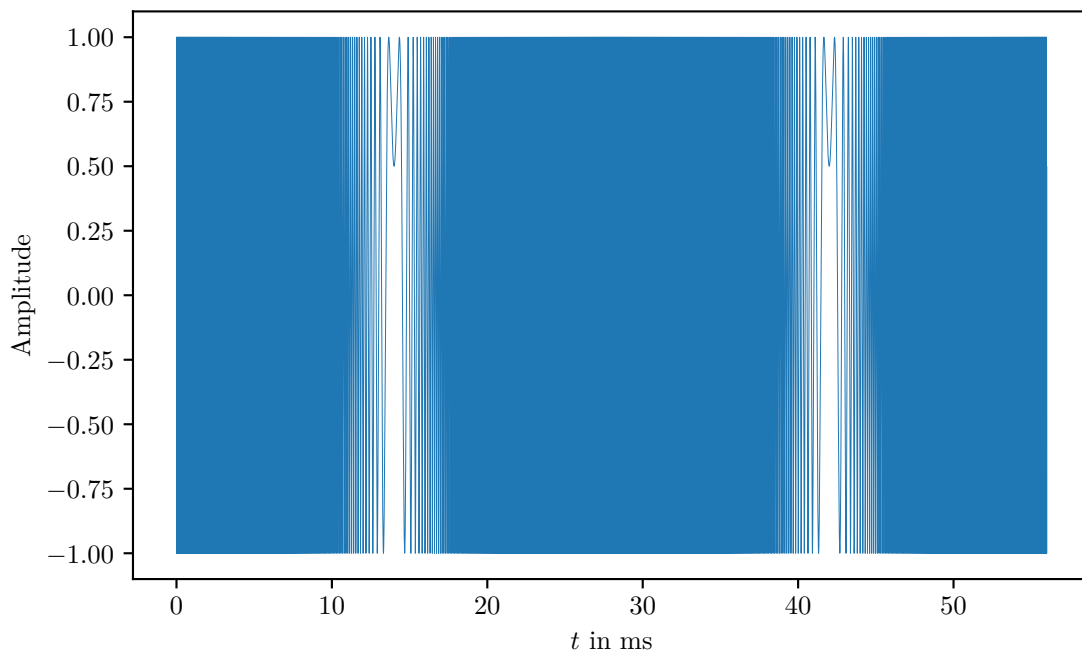


Abbildung 4.9.: Testsignal, doppelt gespiegelter linearer Chirp

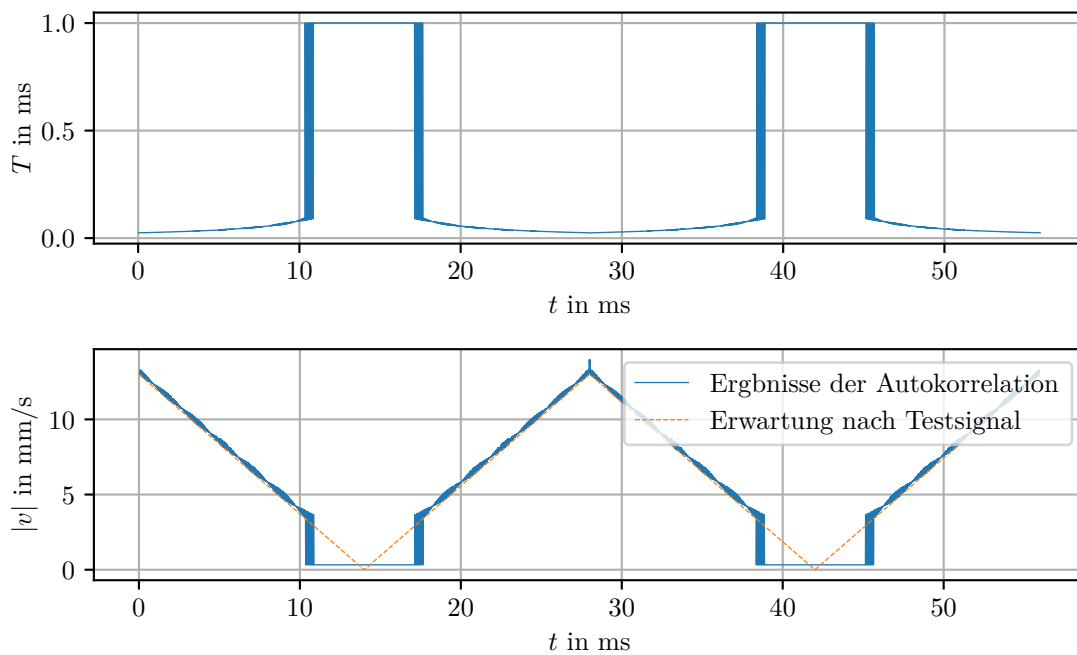


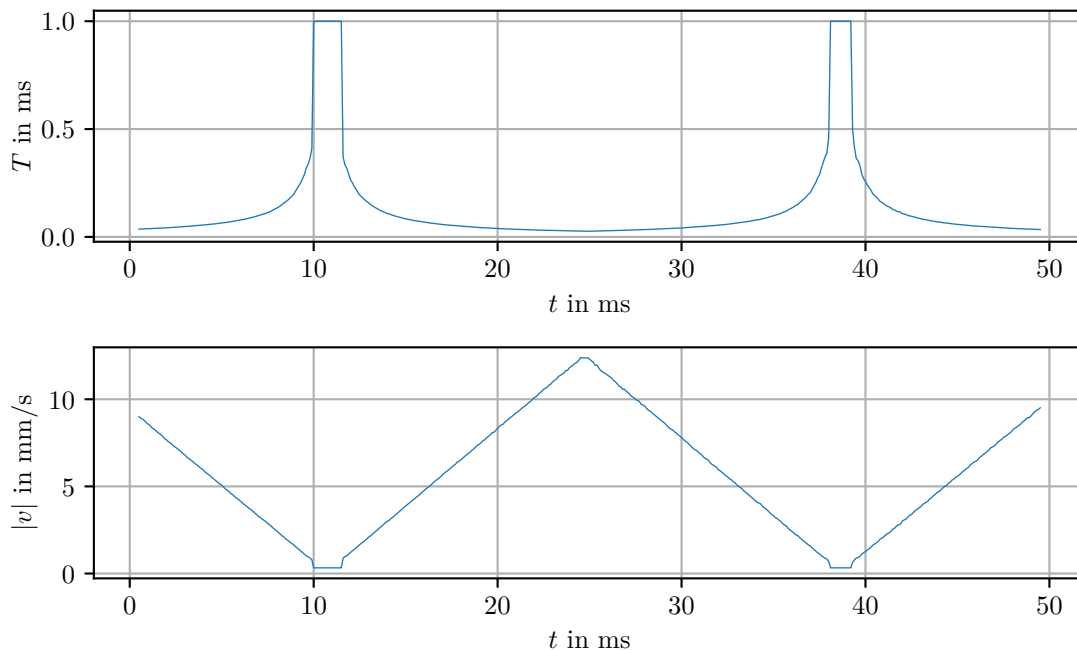
Abbildung 4.10.: Autokorrelationsmessung mit gleitenden Frames am Testsignal



hat noch für kleinere Frequenzen sinnvolle Minima.

Deshalb wird das Verfahren angepasst, mit einer zehnfachen Frame-Länge von 1 ms oder 125000 Samples, die Verschiebung wird wieder auf 0,1 ms oder 12500 Samples gesetzt, was wieder 10% der Frame-Länge entspricht.

Die so mit dem echten Signal berechneten Plots sind in Abbildung 4.11 zu sehen. Das dazugehörige Video ist unter <https://www.youtube.com/watch?v=g0sPW-qhR7w> zu finden.



**Abbildung 4.11.:** Autokorrelationsmessung mit langen Frames am echten Signal

Die erwarteten Verbesserungen sind in Abbildung 4.11 klar zu sehen. Die einhüllende Schwingung ist weg, die Messlücke um  $v = 0$  ist ebenfalls deutlich kleiner.

Zur Referenz wird exakt dieselbe Berechnung noch mit dem Testsignal gemacht. Das Resultat ist in Abbildung 4.12 zu sehen. Das dazugehörige Video ist unter [https://www.youtube.com/watch?v=H\\_5Kf4Hbc1I](https://www.youtube.com/watch?v=H_5Kf4Hbc1I) zu finden.

Wieder liefern das Testsignal und das echte Signal vom Verlauf quasi dasselbe Bild.

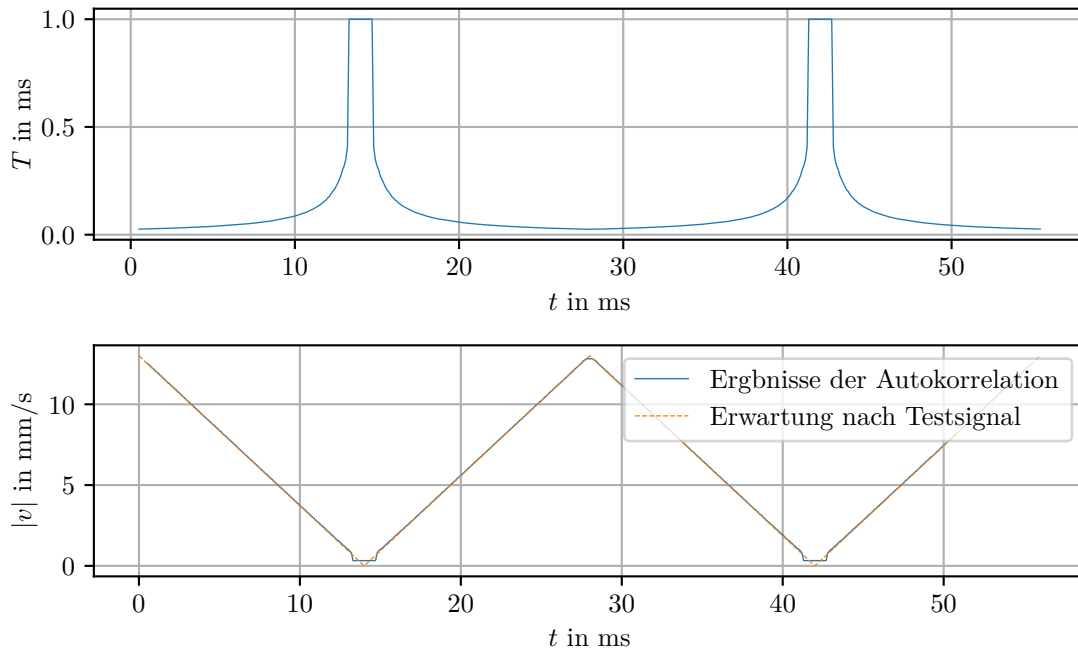
Es fällt bei beiden Berechnungen mit langem Frame noch auf, dass die Spitze am Maximum der Geschwindigkeit etwas rund ist. Das liegt daran, dass im langen Frame am Maximum nicht die wahre Periodendauer bestimmt wird, sondern eine kürzere, da, mit der maximalen Frequenz in der Mitte des Frames, an beiden Rändern des Frames die Frequenzen kleiner sind, weshalb die mittlere Frequenz im Frame zu klein ist.

Bis auf ganz große und ganz kleine Frequenzen liegt der Verlauf der berechneten Geschwindigkeit genau auf dem Verlauf der erwarteten Geschwindigkeit. Diese ist auch in Abbildung 4.12 wieder mit einer gestrichelten Linie eingezeichnet.

#### 4.5.3.8. Anwendung im Versuch und Vergleich zum alten Verfahren

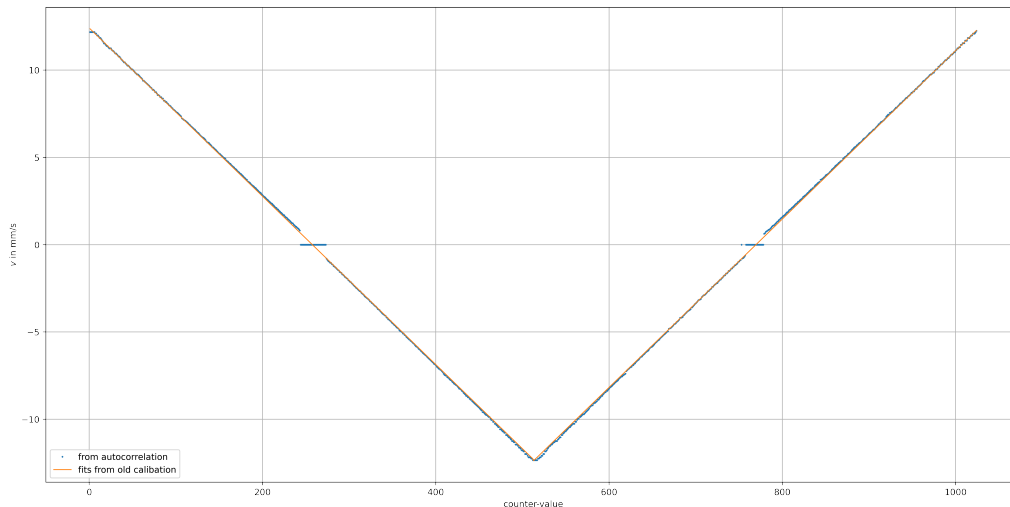
Final soll noch das zuletzt verwendete Verfahren zur Geschwindigkeitsmessung mit dem alten Verfahren verglichen werden.

Dafür wird mit dem Picoscope das Signal erneut aufgezeichnet, allerdings im AC-Modus. Dadurch ist das Signal bereits um  $U = 0$  gemittelt, so kann der Schritt eingespart werden. Zusätzlich wird das START-Signal des DFGs mit aufgezeichnet, um später den Kanalnummern des DFGs je eine Geschwindigkeit zuordnen zu können.



**Abbildung 4.12.:** Autokorrelationsmessung mit langen Frames am Testsignal

Dabei wurde nur mit einer Sample Zeit von 16 ns aufgezeichnet. Entsprechend wurden alle Indexwerte halbiert, um dieselbe Zeit abzudecken. Die Mitte der Frames werden mittig im Zeitfenster der einzelnen Kanal-Fenster platziert. Von dort aus wird für alle 1024 Kanäle eine Autokorrelationsberechnung mit Framelänge 1 ms gemacht. Die Bedingung  $\rho < -0,5$  für die Minima wird weiter verwendet. Die so berechneten Werte, korrekt gespiegelt, sind in Abbildung 4.13 zu sehen.



**Abbildung 4.13.:** Vergleich der neuen Geschwindigkeitsmessung mit dem alten Verfahren

Zum Vergleich ist in Abbildung 4.13 noch die gefittete Geschwindigkeit nach der alten Kalibrierung eingezeichnet. Die Werte der neuen Kalibrierung sind nicht gefittet. Es lässt sich erkennen, dass bis auf die bekannten Lücken um  $v = 0$  beide Verfahren übereinstimmen. Die Lücken um  $v = 0$  ließen sich mittels Interpolation über zwei lineare Fits auch noch schließen. Die genauen Berechnungen finden sich im Musterprotokoll im Anhang. Somit lässt sich sagen, dass die alte Geschwindigkeitsmessung und das neue Verfahren

ziemlich gleichwertig sind. Bei der alten Geschwindigkeitsmessung mussten auch zwei Zählerwerte interpoliert werden, da auf den Kanälen 1 und 2 ein Referenzsignal ist.

Leider ist wegen der notwendigen Interpolation um  $v = 0$  keine Live-Messung der Geschwindigkeit mit dem neuen Verfahren möglich. Trotzdem ist es bemerkenswert, dass die alte Kalibrierung mit extrem vielen DFG-Zyklen und das neue Verfahren mit nur einem Zyklus praktisch identische Werte liefern.

Der Versuch mit der alten Kalibrierung funktioniert auch nur unter der Prämisse, dass die einzelnen Bewegungen alle identisch sind. Gibt es zwischen den einzelnen Zyklen größere Abweichungen, verlieren die Kanalnummern schnell an Bedeutung und die Messung wird unbrauchbar.

#### 4.5.3.9. Fazit

Von allen getesteten Verfahren ist die Autokorrelation definitiv am vielversprechendsten. Die Lücke um  $v = 0$  konnte durch eine große Frame-Länge stark verkleinert werden, auch alle anderen auftretenden Probleme konnten gelöst werden.

Die erstellten Videos geben einen anschaulichen Einblick in das, was passiert. Auch Gegenkontrollen mit einem Testsignal und der alten Geschwindigkeitsmessung am Versuch bestätigen das Verfahren.

Problematisch ist nur, dass die Rechenlücke um  $v = 0$  nicht vollständig geschlossen werden konnte. Deshalb wird in der Anwendung dort immer noch Interpolation notwendig sein. Die erhoffte Live-Messung der Geschwindigkeit (die übrigens auch von den Kanälen unabhängig wäre) ist deshalb leider nicht möglich, zumindest nicht, wenn Geschwindigkeiten um  $v = 0$  relevant sind.

Ein weiterer Nachteil, der bis jetzt nicht angesprochen wurde, ist der große Rechenaufwand, mit dem das Verfahren verbunden ist. Die Autokorrelation eines Arrays der Länge  $n$  besteht aus  $n$  Skalarprodukten, die in ihrer Länge linear von  $n$  bis 1 abnehmen, also eine mittlere Länge von  $n/2$  haben. Das berechnen einer Autokorrelation liegt also in  $\mathcal{O}(n^2)$ . Das Finden des Minimums liegt in  $\mathcal{O}(n)$  und alle weiteren Berechnungen in  $\mathcal{O}(1)$ , diese Einflüsse auf die Rechendauer können also vernachlässigt werden.

Es wird allerdings nicht nur eine Autokorrelation berechnet, sondern z.B.  $m$  hintereinander für  $m$  Frames. Dann liegt die Rechendauer in  $\mathcal{O}(mn^2)$ . In der Praxis haben einige der Berechnungen in dieser Arbeit Stunden gedauert. Auf einem FPGA ließen sich solche Berechnungen sicher optimieren, allerdings muss dann stark auf die Effizienz geachtet werden.



## 5. Fazit

Das Problem der Geschwindigkeitsmessung als Frequenzmessung an einem Chirp wurde im Rahmen dieser Arbeit behandelt und gut verstanden. Die Lösung dieses Problems ist jedoch alles andere als trivial, wie sich gezeigt hat. Alle getesteten Verfahren haben Probleme, kleine Geschwindigkeiten um  $v = 0$  zu messen.

Am besten schneidet ein selbst entwickeltes Verfahren basierend auf Autokorrelation ab. Dieses überzeugt, bis auf die Probleme bei  $v = 0$ , welche alle Verfahren haben, und einen vergleichsweise großen Rechenaufwand. Andere, sonst bewährte Verfahren scheitern an zu viel Rauschen, unzuverlässiger Detektion oder schlechter Frequenzauflösung.

Eine Live-Geschwindigkeitsmessung, wie sie auf dem FPGA praktisch am Versuch zu verwenden wäre, ist leider mit dem Wissen aus dieser Arbeit nicht machbar, da die notwendige Interpolation um  $v = 0$  nicht während der Messung gemacht werden kann.

Somit macht es eigentlich keinen Sinn, das neue Verfahren auf dem FPGA zu implementieren. Womöglich finden sich noch andere Anwendungen, bei denen dieses selbst entwickelte zur Verfahren zur Frequenzmessung eingesetzt werden kann.

Im Rahmen der Arbeit wurde auch die alte Geschwindigkeitsmessung mit dem neuen Autokorrelationsverfahren verglichen. Dabei liefern beide Verfahren praktisch identische Werte, was für beide Verfahren spricht. Vermutlich wäre es für die Zukunft am besten, das alte Verfahren in einer modernisierten Variante zu nutzen. Dieses ist bewährt, für die Studenten einfach verständlich, wenig rechenintensiv und liefert auch für kleine Geschwindigkeiten sinnvolle Werte. Der einzige große Vorteil der anderen Verfahren, eine mögliche Live-Messung, ist, wie bereits beschrieben, mit den Resultaten dieser Arbeit nicht umsetzbar.

Allgemein ist es für den Mößbauer-Versuch noch ein weiter Weg in die Messtechnik des 21. Jahrhunderts. Wie sich während der Arbeit herausgestellt hat, funktioniert der Teil des Versuches, der schon auf einen FPGA portiert wurde, nicht und liefert keine brauchbaren Messwerte. Erste Priorität am Versuch sollte es sein, den gemachten Fortschritt zu sichern. Diese Arbeit ist auch eher als Rückschlag zu sehen im Prozess, den Versuch auf den FPGA zu bringen, da kein wirklich geeignetes, modernes Verfahren gefunden wurde.



# Anhang

## A. Musterprotokoll

Im Rahmen der Bachelorarbeit soll auch ein Musterprotokoll des Versuches geschrieben werden. Da am Versuch oft englischsprachige Tutoren eingesetzt werden, ist das Protokoll auf Englisch verfasst. Das Protokoll ist in der Form eines Jupyter-Notebooks, ein PDF-Export des Notebooks ist im Folgenden zu finden, das Notebook wird allerdings auch direkt bereitgestellt.

Im Protokoll wird die alte Geschwindigkeitskalibrierung durchgeführt. Diese wird mit einer auf den Ergebnissen dieser Arbeit aufbauenden alternativen Geschwindigkeitsmessung mittels Autokorrelation verglichen.

Das Musterprotokoll sollte ursprünglich die neue FPGA-basierte Auswertung des Zählrohrs beinhalten, allerdings war es trotz großem Aufwand nicht möglich, bei dieser Auswertung verwendbare Daten zu erhalten, weshalb der Versuch in der alten Version durchgeführt und ausgewertet wurde. Der einzige Unterschied zur alten Auswertung ist in der FPGA-Auswertung, dass die Energien noch gefiltert werden müssen (macht in der alten Version der Diskriminator) und dann entsprechend für die relevanten Energien die Events für jeden DFG-Kanal gezählt werden müssen. Anschließend ist eine identische Auswertung möglich.

# musterprotokoll\_mb

April 17, 2022

## 1 Report Moessbauer Effect

### 1.1 Chapter 1 - Introduction and theory

Moessbauer spectroscopy is an important tool for many applications in science. It is even used on some [Mars rovers](#) for analysis of iron-bearing minerals on Mars.

The effect, at its core, is about nearly recoil-free emission and absorption of photons. The transitions are nearly recoil-free because the nuclei involved are part of a crystal. When the emission (or absorption) happens, energy and momentum are conserved. Because the nucleus interfering with the photon is part of a solid crystal grid, the momentum transferred to the crystal results in a neglectable movement. In other words the photon is getting nearly all of the energy when emitted. When absorbed, the nucleus gets nearly all of the energy of the photon. Because of that, the energy resolution is very high.

In the effect an oscillating emitter consisting of Co-57 is used. The Co-57 degrades into excited Fe-57 (excited with  $I = 5/2$ , which then drops to  $I = 3/2$ , later drops to the ground state  $I = 1/2$ ). For the experiment the transition from  $I = 3/2$  to  $I = 1/2$  (in the emitter; it is the other way around in the absorber) is particularly interesting.

In addition to just resonance there is three effects that can be observed.

The first effect is the isomer shift (or chemical shift). It is a constant shift in energy caused by different electron configurations within the emitter and the absorber.

The second effect is magnetic hyperfine splitting, a result of the Zeeman effect if there is a magnetic field present in the nucleus. The ground state splits into two states,  $m = \pm 1/2$ , the  $I = 3/2$ -state splits into four states,  $m = \pm 3/2$  or  $m = \pm 1/2$ . Two of the possible transitions are forbidden due to transition rules, so there is six transitions still possible. Magnetic hyperfine splitting is characterized by a splitting into six energies.

The third effect, quadrupole splitting, is present when there is an electric field gradient in the nucleus. The effect needs a non-spherical charge distribution ( $I \neq 1/2$ ). Additionally only  $m^2$  is relevant (sign of  $m$  is not relevant). So for the remaining  $I = 3/2$ -state,  $m = \pm 3/2$  and  $m = \pm 1/2$  only produce a split in two energies. Quadrupole splitting can be recognized by a splitting in two energies.

All three effects can overlay and appear at the same time. In addition splitting in the emitter and absorber could also overlay. In order to better observe those effects the emitter is free of any inner magnetic fields or electric field gradients. This can be achieved by using an alloy with platinum or palladium. The emitter is then known as a one-line-emitter, as any effects splitting the energy states are suppressed.



The absorber is what is analyzed in the experiment. There is a total of four samples which need to be analyzed: pure iron, steel (vacromium) and the salts FeSO<sub>4</sub> and FePO<sub>4</sub>.

In this experiment iron (its isotope Fe-57) is used, as it was historically the first isotope used by Moessbauer to explore the effect, even though Moessbauer Spectroscopy is possible with many other elements. However, temperature is a big factor in Moessbauer Spectroscopy. The Lamb-Moessbauer-Factor (short LMF) (similar to the Debye-Waller-Factor in other cases) is used to describe the ratio of recoil-free to total nuclear resonant absorptions. With higher temperatures there is more movement happening between the involved nuclei. After a certain temperature is reached, the LMF becomes 0; at 0 K temperature the factor is maximal (almost 1). Depending on the material a colder temperature may be needed to display the effect. Iron is special in that regard, as it requires no cooling to show the effect, thus the experiment can be done at room temperature.

## 1.2 Chapter 2 - Doing the experiment

The experiment is done as described in the co-called Blue Book.

At first the new version of the experiment using the FPGA-based analysis is used. However, even after many hours of trying it was not possible to find good parameters, with which the resonance absorption would have been clearly visible. As a consequence, the old, standard version from WissEl is used.

Starting the experiment, the discriminator needs to be set properly so that only events at about 14.4 keV get detected. This can be done looking at the energy spectrum and trying to find the peak that corresponds to the above mentioned energy of 14.4 keV. An alternative for fine tuning is running the experiment and tweaking the discriminator untill the amplitude of the resonance dips becomes maximal.

If the experiment was done via the FPGA the direct signal of the Geiger-Mueller-Tube would need to be plugged into the the red pitaya FPGA board. The energy selection which in the other version is done via the discriminator, would need to be done manually after the measurement. Afterwards, the events for each channel number need to be counted.

The old experiment just outputs the channel number of the function generator and the corresponding number of events counted.

The experiment was run with four samples: vacromium, pure iron, FeSO<sub>4</sub> and FePO<sub>4</sub>.

Regardless of the sample, a velocity calibration using the interferometer needs to be done first.

## 1.3 Chapter 3 - Velocity calibration

This analyis aquires some standard python-libraries for this analysis.

```
[ ]: import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from math import pi, e, log, sqrt
from scipy.optimize import curve_fit
from uncertainties import ufloat
```

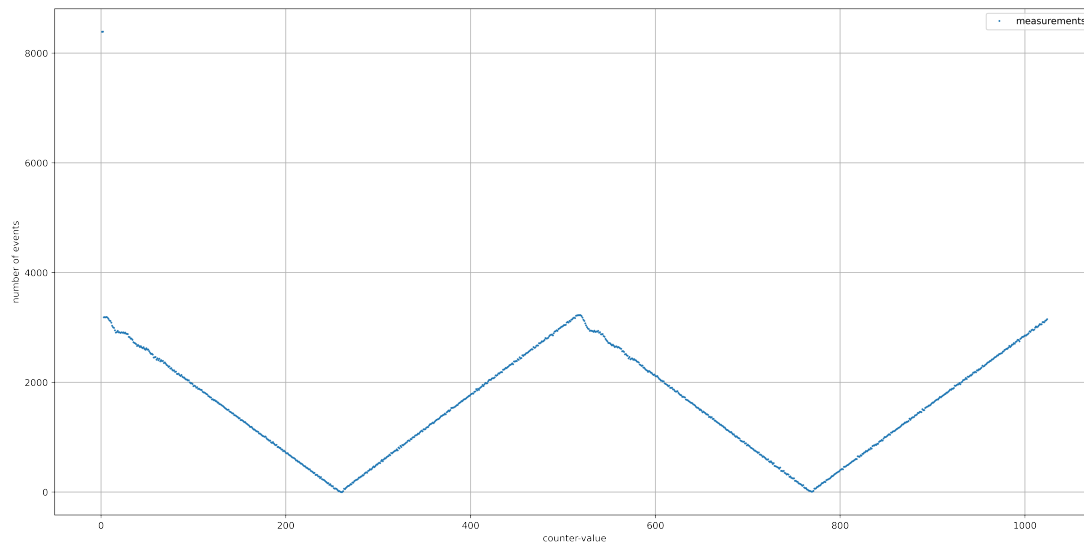
To analyze, first we need to read in the velocity calibration file. We want to use pandas for this analysis, so the data needs to be loaded into a pandas dataframe. We need some rudimentary plotting of the data as well, to get a look of what to do next.

```
[ ]: vel_file = np.genfromtxt('./../data/calibration.txt')
df_vel = pd.DataFrame({'chn':vel_file[:,0], 'count':vel_file[:,1]},dtype=int)
print(df_vel)

plt.figure(figsize=(20,10), dpi=500)
# plot raw data
plt.plot(df_vel['chn'], df_vel['count'], label='measurements',
         linestyle='None', marker='+', markersize=2)
# pretty plotting
plt.grid()
plt.legend(loc='best')
plt.xlabel('counter-value')
plt.ylabel('number of events')
plt.show()
```

	chn	count
0	1	8390
1	2	8391
2	3	3185
3	4	3189
4	5	3185
...	...	...
1019	1020	3097
1020	1021	3095
1021	1022	3123
1022	1023	3132
1023	1024	3151

[1024 rows x 2 columns]



The data needs some processing. Initially, the first two values contain pulses of a reference signal (with  $f = 100\text{kHz}$ ). These values need to be removed. In addition, values in the center need their sign flipped, as this information is lost because of the interferometer.

```
[ ]: # get ref signal (Counter 1 and 2 (0 and 1 here)) and remove from data
ref_count = df_vel['count'][1]
df_vel = df_vel[df_vel['chn'] > 2]

# Flip sign of count
start = 2**0
border_left = 2**8
center = 2**9
border_right = 2**9 + 2**8
end = 2**10
# Call by references
count_ref = df_vel['count'].to_numpy()
count_ref[border_left:border_right] *= -1
# change will be converted to df
print(df_vel)
```

	chn	count
2	3	3185
3	4	3189
4	5	3185
5	6	3191
6	7	3180
...	...	...
1019	1020	3097
1020	1021	3095

```
1021  1022  3123
1022  1023  3132
1023  1024  3151
```

```
[1022 rows x 2 columns]
```

Now, using the reference signal removed earlier we can calculate the velocities.

```
[ ]: def calculate_vel(ref_count, current_count):
      # ref count has f=100kHz
      f_ref = 1e5 # Hz
      lambda_laser = 650e-9 # in m
      v = (lambda_laser / 2) * f_ref * (current_count / ref_count)
      return v # in m/s
df_vel['vel_unfit'] = calculate_vel(ref_count, df_vel['count'])
print(df_vel)
```

```
      chn  count  vel_unfit
2         3   3185   0.012336
3         4   3189   0.012352
4         5   3185   0.012336
5         6   3191   0.012359
6         7   3180   0.012317
...
1019  1020   3097   0.011995
1020  1021   3095   0.011988
1021  1022   3123   0.012096
1022  1023   3132   0.012131
1023  1024   3151   0.012204
```

```
[1022 rows x 3 columns]
```

As a result, we have a velocity for each channel. However, those values are still a bit random and the first two channels are missing because they contained the reference signal. So a linear fit is done (technically two fits; for rising and falling velocity). This can interpolate the missing values and help smooth out the other values.

```
[ ]: # fit two lines for rising and falling velocity
      # split up at center
df_l1 = df_vel[df_vel['chn'] <= center]
chn_l1 = df_l1['chn'].to_numpy()
vel_l1 = df_l1['vel_unfit'].to_numpy()
df_l2 = df_vel[df_vel['chn'] >= center]
chn_l2 = df_l2['chn'].to_numpy()
vel_l2 = df_l2['vel_unfit'].to_numpy()

def lin(x, A, C):
    y = A * x + C
    return y
```

```

[A_l1, C_l1], cov_1 = curve_fit(lin, chn_l1, vel_l1)
l1_chn = np.arange(start, center+1)
l1_vel_fit = lin(l1_chn, A_l1, C_l1)

[A_l2, C_l2], cov_2 = curve_fit(lin, chn_l2, vel_l2)
l2_chn = np.arange(center+1, end+1)
l2_vel_fit = lin(l2_chn, A_l2, C_l2)

# Store results in Dataframe
# add back chn 1 and 2
row_1 = [1,0,0]
row_2 = [2,0,0]
df_vel.loc[0] = row_1
df_vel.loc[1] = row_2
# sort df again
df_vel = df_vel.sort_index()

# append arrays and add as column
vel_fit = np.append(l1_vel_fit, l2_vel_fit)

df_vel['vel_fit'] = vel_fit
print(df_vel)

```

	chn	count	vel_unfit	vel_fit
0	1	0	0.000000	0.012364
1	2	0	0.000000	0.012316
2	3	3185	0.012336	0.012268
3	4	3189	0.012352	0.012219
4	5	3185	0.012336	0.012171
...	...	...	...	...
1019	1020	3097	0.011995	0.012076
1020	1021	3095	0.011988	0.012124
1021	1022	3123	0.012096	0.012173
1022	1023	3132	0.012131	0.012221
1023	1024	3151	0.012204	0.012269

[1024 rows x 4 columns]

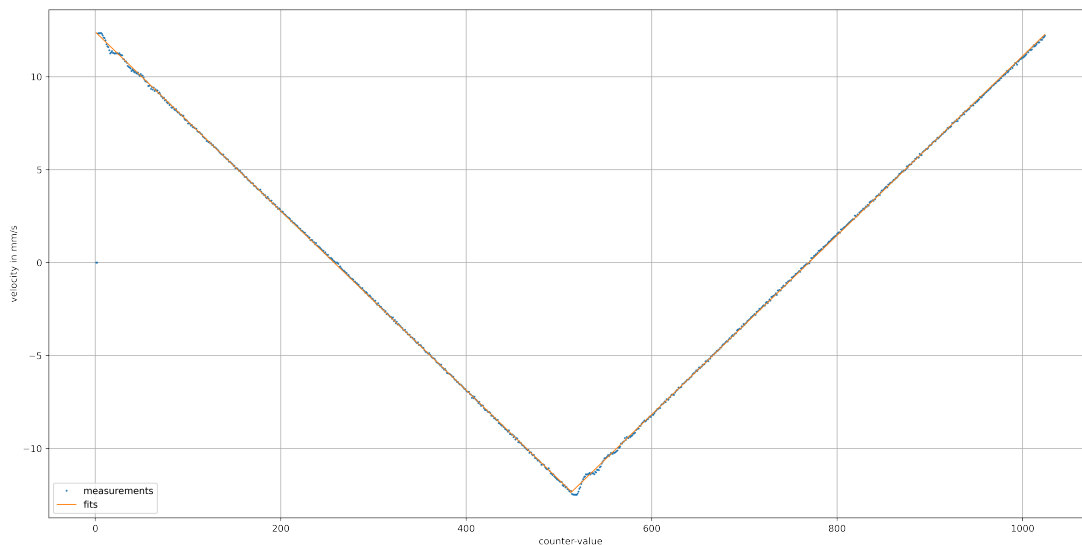
Next, we can plot the measured values together with the fit functions.

```

[ ]: plt.figure(figsize=(20,10), dpi=500)
# plot raw data
plt.plot(df_vel['chn'], df_vel['vel_unfit']*1000, label='measurements',
        linestyle='None', marker='+', markersize=2)
# plot fits
plt.plot(df_vel['chn'], df_vel['vel_fit']*1000, label='fits', linestyle='-',
        marker=None, linewidth=1)

```

```
# pretty plotting
plt.grid()
plt.legend(loc='best')
plt.xlabel('counter-value')
plt.ylabel('velocity in mm/s')
plt.show()
```



As one can see, the linear fits describe the behavior quite well. For further usage we will define a velocity function that can be used to convert a channel number into a velocity.

```
[ ]: def vel_cal(channel):
    '''Provide calibrated velocity for each channel number
    '''
    # channels start from 1
    index = channel - 1
    return vel_fit[index]
```

### 1.3.1 New velocity calibration

In addition to the old velocity calibration, a new velocity calibration based on autocorrelation will be performed. For this calibration, a recording of the interferometer signal from one DAC-cycle is needed. To be able to map channel numbers to the velocities, a recording of the “start”-signal is also needed. First, the data need to be read in. In a second step, the times for the different channel numbers need to be calculated. Afterwards, using autocorrelation, we can determine a frequency and thus a velocity. Finally, the values are compared to the values of the old calibration.

```
[ ]: df_vel_new = pd.read_csv("../data/autocorr_calibration.txt", delimiter="\t",
    ↪ skiprows=3, names=['t[ms]', 'A[mV]', 'B[V]'], decimal=".")
print(df_vel_new.head())
```

	t[ms]	A[mV]	B[V]
0	0.000000	-56.55758	0.433840
1	0.000016	-55.67482	0.390471
2	0.000032	-57.44033	0.390471
3	0.000048	-56.55758	0.433840
4	0.000064	-56.55758	0.390471

After the data is read in, using channel B, the start times of the cycle need to be determined. Using those, times for the center of each frame for the autocorrelation need to be calculated.

```
[ ]: # convert to numpy array
start_signal = df_vel_new['B[V]'].to_numpy()

threshold_voltage = 1 # 1 volt threshold voltage

# detect 2 rising edges of start signal
list_index_found = []
for i in range(len(start_signal)-1):
    if (start_signal[i] < threshold_voltage) and (start_signal[i+1] >=
↳threshold_voltage):
        list_index_found.append(i)

# only 2 detections wanted, start and stop of frame
assert len(list_index_found) == 2
cycle_start_index = list_index_found[0]
cycle_end_index = list_index_found[1]
cycle_len = cycle_end_index - cycle_start_index

step_time = 16e-9 # 16ns step time

print("The whole cycle takes {} ms".format(step_time * cycle_len * 1000))

number_channels = 2*10
channel_length = cycle_len // number_channels
print("Each channel has {} indices".format(channel_length))

# we want to get the center index for each of the channels for autocorrelation
↳(ac) frame
half_channel_len = cycle_len // (2 * number_channels) # half the lengt of a
↳channel
first_ac_index = cycle_start_index + half_channel_len
last_ac_index = cycle_end_index - half_channel_len

ac_indices = np.linspace(first_ac_index, last_ac_index, num=number_channels,
↳dtype=int)
print("Got all indices for frame centers")
print(ac_indices)
```

The whole cycle takes 53.729728 ms

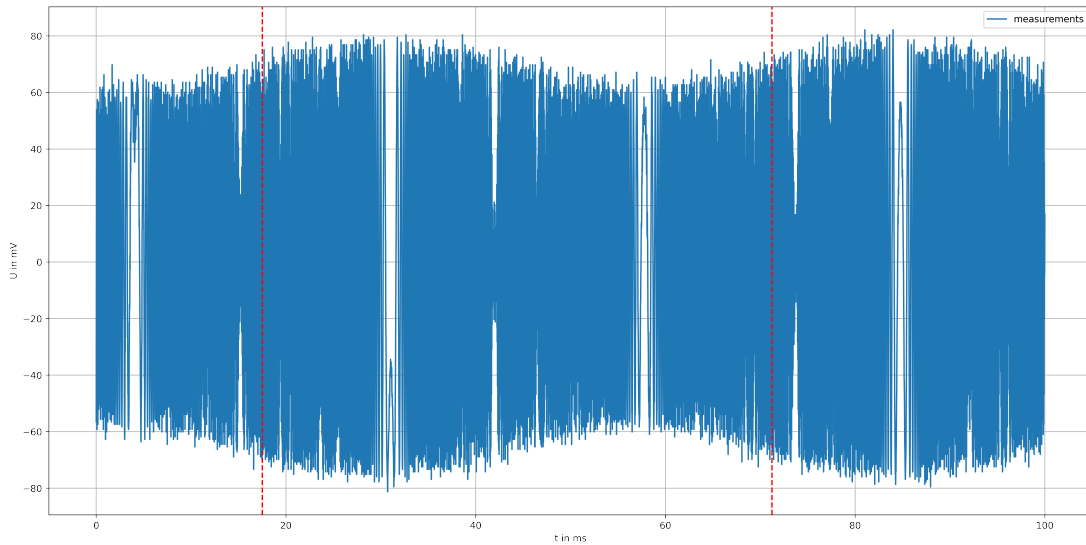
Each channel has 3279 indices

Got all indices for frame centers

```
[1095864 1099143 1102422 ... 4444135 4447414 4450694]
```

Now, the indices in the center for all channels are acquired. Using these, autocorrelation can be performed. To do that, at first a look at the actual interferometer data is needed. The start and stop signal of one cycle are drawn in as vertical red lines.

```
[ ]: plt.figure(figsize=(20,10), dpi=500)
      # plot raw data (downsampled for plotting)
      plt.plot(df_vel_new['t[ms]'][:,::1000], df_vel_new['A[mV]'][:,::1000],
               label='measurements')
      # insert start and stop line
      plt.axvline(x=(cycle_start_index * step_time * 1000), color='red',
                 linestyle='--')
      plt.axvline(x=(cycle_end_index * step_time * 1000), color='red', linestyle='--')
      # pretty plotting
      plt.grid()
      plt.legend(loc='best')
      plt.xlabel('t in ms')
      plt.ylabel('U in mV')
      plt.show()
```



Next, the actual autocorrelation computation is performed. For each frame, the autocorrelation is computed and searched for the first minimum. The first minimum is always the global minimum. If the first minimum is larger than  $\rho = -0.5$ , the minimum is not used. Else, the velocity is given as  $v = \frac{\lambda}{4 \cdot t_{\min}}$  with the laser wavelength  $\lambda$  and the time of the first minimum  $t_{\min}$ .



```
[ ]: from PhyPraKit import autocorrelate

lambda_laser = 650e-9 # 650 nm
interferometer_signal = df_vel_new['A[mV]'].to_numpy()

frame_len = 62500 # 1 ms frame length
half_frame_len = frame_len // 2

# do the actual autocorrelation
v_ac = []
for i in range(number_channels):
    current_index = ac_indices[i]
    frame_start = current_index - half_frame_len
    frame_end = current_index + half_frame_len

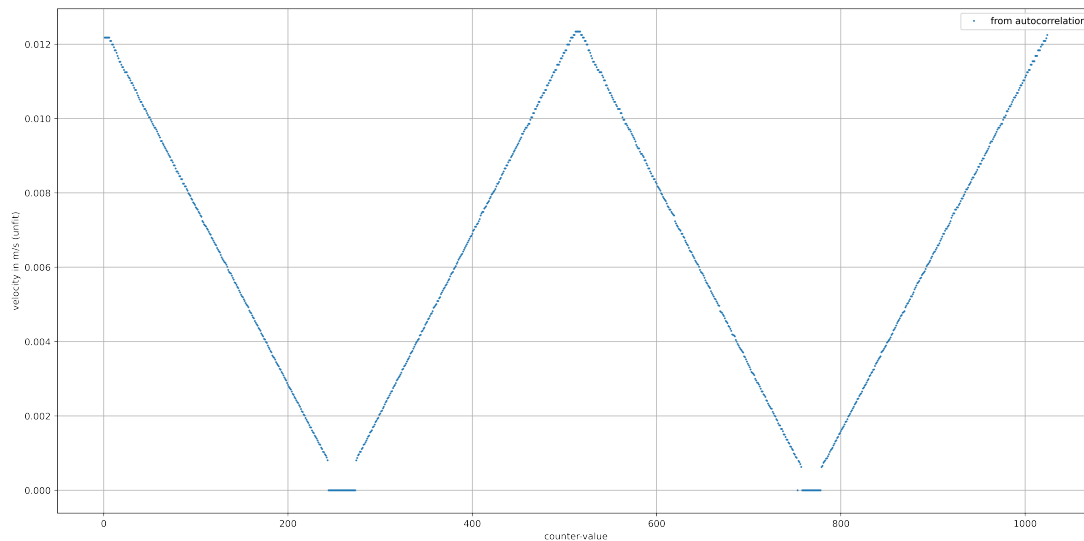
    ac = autocorrelate(interferometer_signal[frame_start:frame_end])

    first_min = np.min(ac)

    if first_min > -0.5:
        # no minimum found
        v_frame = 0
    else:
        # minimum found, calculate v
        first_min_index = np.argmin(ac)
        v_frame = lambda_laser / (4 * first_min_index * step_time)
    v_ac.append(v_frame)
```

This calculation takes some time (for the author, it took about 24 minutes...). With this finally done, plotting the calculated velocities can be done.

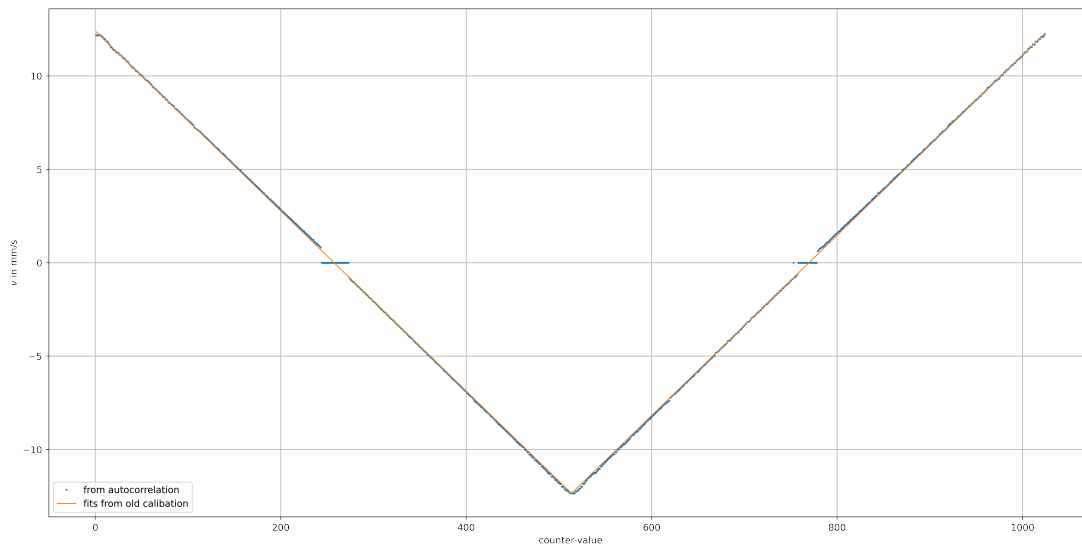
```
[ ]: plt.figure(figsize=(20,10), dpi=500)
# plot raw data
channels = np.arange(number_channels) + 1
plt.plot(channels, v_ac, label='from autocorrelation', linestyle='None',
        marker='+', markersize=2)
# pretty plotting
plt.grid()
plt.legend(loc='best')
plt.xlabel('counter-value')
plt.ylabel('velocity in m/s (unfit)')
plt.show()
```



The velocity plot looks quite similar to the first plot of the old calibration. Only for very small velocities, the analysis fails. The next this to do is flip the velocity of the center triangle.

```
[ ]: # convert list to np array
vel_ac = np.array(v_ac)
vel_ac[(border_left-1):(border_right-1)] *= -1

plt.figure(figsize=(20,10), dpi=500)
# plot raw data
plt.plot(channels, vel_ac*1000, label='from autocorrelation', linestyle='None',
        marker='+', markersize=2)
# plot fit from old calibration
plt.plot(df_vel['chn'], df_vel['vel_fit']*1000, label='fits from old
        calibration', linestyle='--', marker=None, linewidth=1)
# pretty plotting
plt.grid()
plt.legend(loc='best')
plt.xlabel('counter-value')
plt.ylabel('$v$ in mm/s')
plt.show()
```



The correctly flipped calculated velocities from the autocorrelation are plotted together with the fit result from the old calibration. As one can see, they line up perfectly. For a full calibration based on autocorrelation, two linear fits would need to be done, like before with the old calibration. Using these linear fits, a velocity for the channels close to  $v = 0$  can be interpolated. The new fit would have very similar results to the old fit, as one can already see in the plot. Because of that, all further calculations will be performed using the old calibration. However, it is clearly possible to show that both methods are in harmony.

## 1.4 Chapter 4 - Analysis of Vacromium

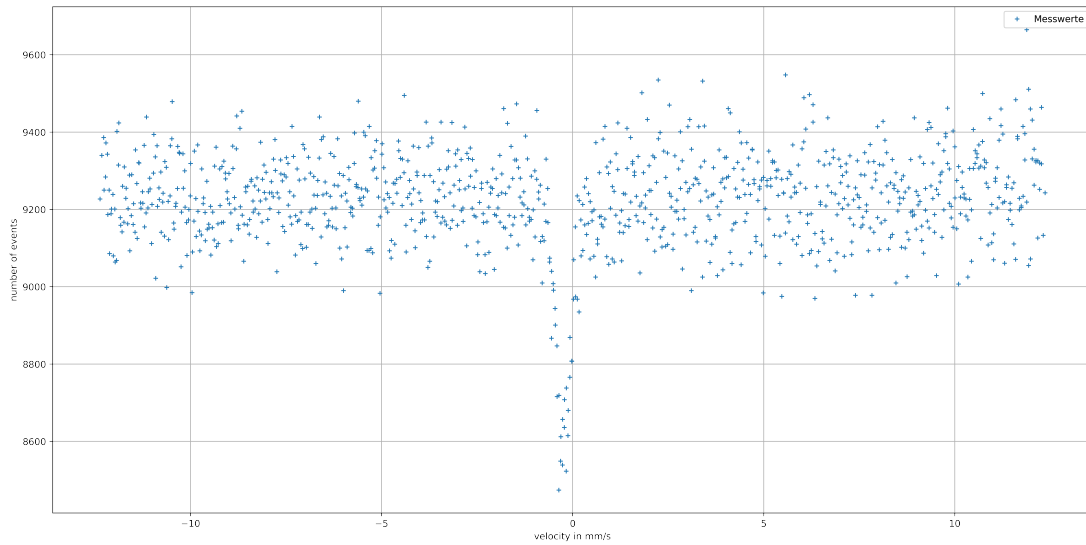
With the velocity analysis done, the analysis for the first of the measured materials, vacromium (a.k.a. stainless steel), can be done. First, the data is read in, the velocity calibration is directly applied. Again, plotting the data first makes sense.

```
[ ]: vc_file = np.genfromtxt('./../data/vc.txt')
df_vc = pd.DataFrame({'chn':vc_file[:,0], 'count':vc_file[:,1]}, dtype=int)
df_vc['vel'] = vel_cal(df_vc['chn'])
print(df_vc)
plt.figure(figsize=(20,10), dpi=500)
plt.plot(1000*df_vc['vel'], df_vc['count'], label='Messwerte',
        linestyle='None', marker='+', markersize=5)
plt.grid()
plt.legend(loc='best')
plt.xlabel('velocity in mm/s')
plt.ylabel('number of events')
plt.show()
```

	chn	count	vel
0	1	9243	0.012364

1	2	9133	0.012316
2	3	9464	0.012268
3	4	9320	0.012219
4	5	9126	0.012171
...	...	...	...
1019	1020	9356	0.012076
1020	1021	9327	0.012124
1021	1022	9326	0.012173
1022	1023	9252	0.012221
1023	1024	9318	0.012269

[1024 rows x 3 columns]



There clearly is a single dip caused by resonance absorption which is visible in the data. To do further analysis, a function will be fit to the data. It is possible to fit the data via a Gaussian function, a Lorentz function or the convolution of both, a Voigt function. To keep the fit and the calculations simple, we will use a Gaussian function of the form

$$f_{\text{Gauss}}(x) = a \cdot e^{-\frac{(x-b)^2}{2c^2}}.$$

Here,  $a$  is the amplitude of the peak,  $b$  is the position of the peak ( $b = \mu$ ) and  $c$  is the standard deviation ( $c = \sigma$ ) of the distribution.

To fit the signal, a constant offset (the base rate) will be assumed, where  $f_{\text{Gauss}}$  will be subtracted. So the actual fit function is

$$f_{\text{fit}}(x) = \text{offset} - f_{\text{Gauss}}(x) = \text{offset} - a \cdot e^{-\frac{(x-b)^2}{2c^2}}.$$

For fitting, the `curve_fit` function from the library `scipy.optimize` is used. To get the fit to converge good initial values need to be passed.

```
[ ]: # modified gauss for fitting
def gauss(x, offset, a, b, c):
    # start at offset, subtract gauss profile
    y = offset - (a * e**((-x-b)**2) / (2*c**2))
    return y

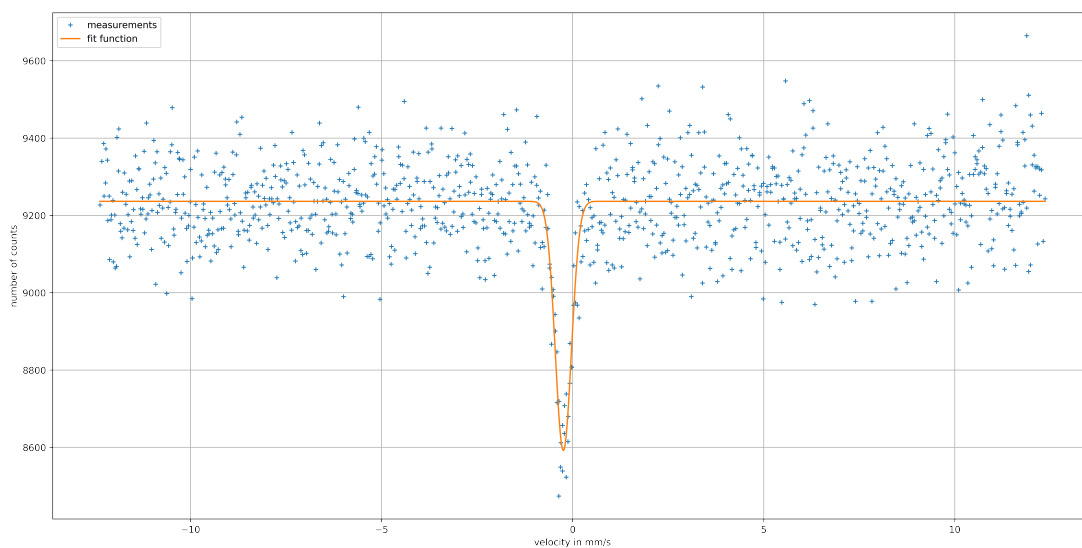
guess_params=[df_vc['count'].mean(), 800, -0.00035, 0.0001]
print('Starting parameters: ', guess_params)

param, cov = curve_fit(gauss, df_vc['vel'].to_numpy(), df_vc['count'].
    ↳to_numpy(), sigma=np.sqrt(df_vc['count'].to_numpy()), p0=guess_params,
    ↳maxfev=1000)
print('Converging parameters: ', param)
# calculate to draw fit-function
fit_vel = np.linspace(df_vc['vel'].min(), df_vc['vel'].max(), int(1e5))
fit_rate = gauss(fit_vel, *param)

plt.figure(figsize=(20,10), dpi=500)
plt.plot(1000*df_vc['vel'], df_vc['count'], label='measurements',
    ↳linestyle='None', marker='+', markersize=5)
plt.plot(1000*fit_vel, fit_rate, label='fit function')
plt.grid()
plt.legend(loc='best')
plt.xlabel('velocity in mm/s')
plt.ylabel('number of counts')
plt.show()
```

Starting parameters: [9223.802734375, 800, -0.00035, 0.0001]

Converging parameters: [ 9.23642904e+03 6.44372908e+02 -2.43353972e-04  
2.10375406e-04]



The fit describes the measured values quite well. With this information further calculations can be done. To automatically calculate uncertainties, the equally called python-library *uncertainties* is used. The statistical uncertainties from the fit are used as uncertainties.

The first variable to be calculated is the isometry shift. In units of velocity it can be directly acquired from the fit values. It is the shift of the Gaussian Peak from 0. To do the conversion, the calculation of the Doppler shifted energy difference  $\Delta E$  is

$$\Delta E = E_0 \cdot \frac{v}{c},$$

for a Photon that already has the energy  $E_0$  and is emitted from a system moving at velocity  $v \ll c$ . For the experiment,  $E_0$  is known to be about 14.4 keV so the calculations will be done in units of eV.

```
[ ]: # Put the fit variables into ufloat-variables to take care of propagation of
      ↪ uncertainties
vc_offset = ufloat(param[0], sqrt(cov[0][0]))
vc_amplitude = ufloat(param[1], sqrt(cov[1][1]))
vc_velocity = ufloat(param[2], sqrt(cov[2][2]))
vc_sigma = ufloat(param[3], sqrt(cov[3][3]))
print('Offset from fit: ', vc_offset, ' counts')
print('Amplitude from fit: ', vc_amplitude, ' counts')
print('Velocity from fit: ', vc_velocity*1000, ' mm/s')
print('Gaussian standard deviation from fit: ', vc_sigma*1000, ' mm/s')

# calculate Doppler shifted energy
def doppler_delta_E_from_v(v):
    E_0 = 14.4e3 # E in eV
    c = 3e8 # c in m/s
    delta_E = E_0 * (v / c)
    return delta_E

# Calculate isometrie-shift:
print('Resonance at ', vc_velocity * 1000, 'mm/s (isometry shift)')
vc_E_isoshift = doppler_delta_E_from_v(vc_velocity)
print('This is equal to ', vc_E_isoshift, 'eV (isometry shift)')
```

```
Offset from fit: 9236.4+/-3.3 counts
Amplitude from fit: 644+/-31 counts
Velocity from fit: -0.243+/-0.012 mm/s
Gaussian standard deviation from fit: 0.210+/-0.012 mm/s
Resonance at -0.243+/-0.012 mm/s (isometry shift)
This is equal to (-1.17+/-0.06)e-08 eV (isometry shift)
```

Another variable that can be calculated with the Vacromium measurement is the mean lifetime  $\tau$  of the 14.4 keV-state. Therefore the natural energy width  $\Gamma_0$  of the resonance dip is used. Heisenberg's

uncertainty theorem gives a relation with the mean lifetime  $\tau$  of

$$\Gamma_0 \cdot \hbar = \tau \implies \tau = \frac{\hbar}{\Gamma_0}.$$

For a Gaussian distribution, the width  $\Gamma_{\text{Gauss}}$  is given as

$$\Gamma_{\text{Gauss}} = 2\sqrt{2\log 2}\sigma.$$

However, as this is a transmission measurement the  $\Gamma_{\text{Gauss}}$  needs to be halved, because the transmission and emission both contribute to this curve (it's the convolution of both). To get the natural width  $\Gamma_0$  an additional conversion from the velocity difference to energy difference needs to be done with the Doppler formula. Where the conversion is done does not matter, as it is linear in  $v$ .

```
[ ]: # Calculate lifetime of 14.4-keV-state
hbar = 6.58e-16 # in eV * s

Gamma_Gauss = 2 * sqrt(2 * log(2)) * vc_sigma
Gamma_v = Gamma_Gauss * 0.5
Gamma_0 = doppler_delta_E_from_v(Gamma_v)

tau = hbar / Gamma_0
print("Gamma from Gauss plot: ", Gamma_Gauss * 1000, "mm/s")
print("Gamma_0 as energy width:", Gamma_0, 'eV')
print("Tau:", tau, 's')
```

```
Gamma from Gauss plot: 0.495+/-0.028 mm/s
Gamma_0 as energy width: (1.19+/-0.07)e-08 eV
Tau: (5.53+/-0.32)e-08 s
```

In the blue book,  $\Gamma_0$  is given with  $\Gamma_0 = 5 \cdot 10^{-9}$  eV, which corresponds to  $\tau = 1.316 \cdot 10^{-7}$  s.

The values calculated here are off by a factor of 2-3. To achieve a better analysis, more effects should probably be considered.

## 1.5 Chapter 5 - Analysis of pure Iron

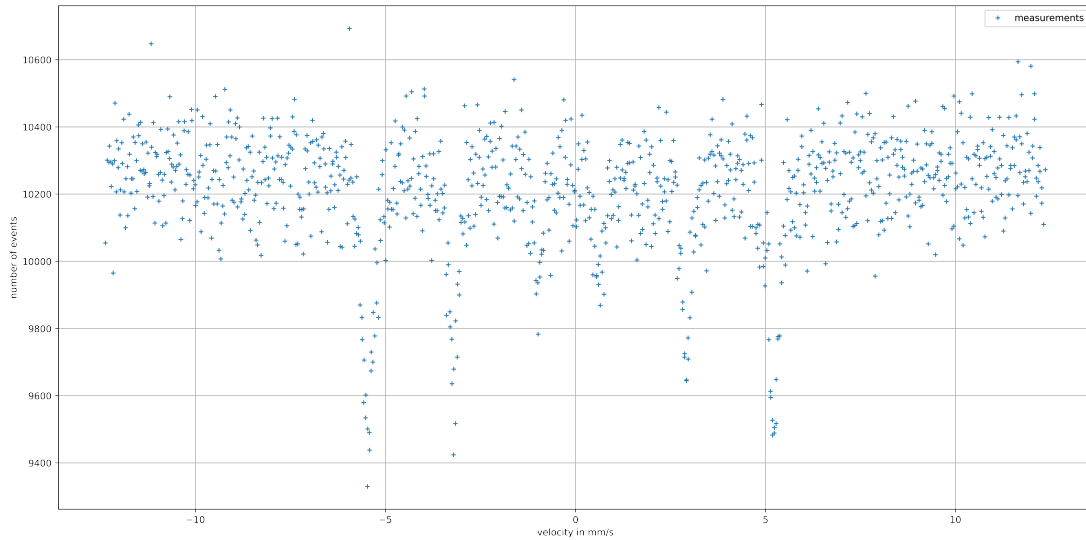
Next, the analysis of pure iron (Fe) will be done. Again, first the data needs to be read in and plotted.

```
[ ]: fe_file = np.genfromtxt('../data/Fe.txt')
df_fe = pd.DataFrame({'chn':fe_file[:,0], 'count':fe_file[:,1]}, dtype=int)
df_fe['vel'] = vel_cal(df_fe['chn'])
print(df_fe)
plt.figure(figsize=(20,10), dpi=500)
plt.plot(1000*df_fe['vel'], df_fe['count'], label='measurements',
        linestyle='None', marker='+', markersize=5)
plt.grid()
plt.legend(loc='best')
plt.xlabel('velocity in mm/s')
plt.ylabel('number of events')
```

```
plt.show()
```

	chn	count	vel
0	1	10273	0.012364
1	2	10110	0.012316
2	3	10173	0.012268
3	4	10339	0.012219
4	5	10285	0.012171
...	...	...	...
1019	1020	10423	0.012076
1020	1021	10194	0.012124
1021	1022	10237	0.012173
1022	1023	10268	0.012221
1023	1024	10219	0.012269

```
[1024 rows x 3 columns]
```



Again, there needs to be a function fit to these measurements. There will again be a constant offset of which Gaussian functions will be subtracted. However, as it is known that these dips are symmetrical around the isometry-shifted center, there will be six Gaussian functions subtracted. But as they are symmetrical around the isometry shift, two will share all parameters except for being mirrored around the isometry-shifted center. So the fit function looks like

$$f(x) = \text{offset} - \sum_{i=1}^3 f_{\text{SingleGauss}}(x, a_i, \text{isoshift} \pm b_i, c_i).$$

There are eleven free parameters in this fit (three sets of three Gaussian parameters, offset and isoshift), so it is quite important to set good starting values for the fit to converge nicely.



```
[ ]: # single gauss for fitting
def single_gauss(x, a, b, c):
    # start at offset, subtract gauss profile
    y = (a * e**((-x-b)**2) / (2*c**2))
    return y
# more complex function
def fit_func_fe(x, offset, isoshift, a_1, b_1, c_1, a_2, b_2, c_2, a_3, b_3,
    ↪c_3):
    # offset is base rate
    # isoshift is shifted velocity 0
    # then, three pairs of gaussian distributions are subtracted, symmetrically
    ↪around the isoshift.
    y = offset
    y -= single_gauss(x, a_1, isoshift+b_1, c_1)
    y -= single_gauss(x, a_1, isoshift-b_1, c_1)
    y -= single_gauss(x, a_2, isoshift+b_2, c_2)
    y -= single_gauss(x, a_2, isoshift-b_2, c_2)
    y -= single_gauss(x, a_3, isoshift+b_3, c_3)
    y -= single_gauss(x, a_3, isoshift-b_3, c_3)
    return y

guess_params=[df_fe['count'].mean(), -0.0001, 400, 0.0008, 0.0001, 600, 0.003,
    ↪0.0001, 800, 0.0055, 0.0001]
print('Starting parameters: ', guess_params)

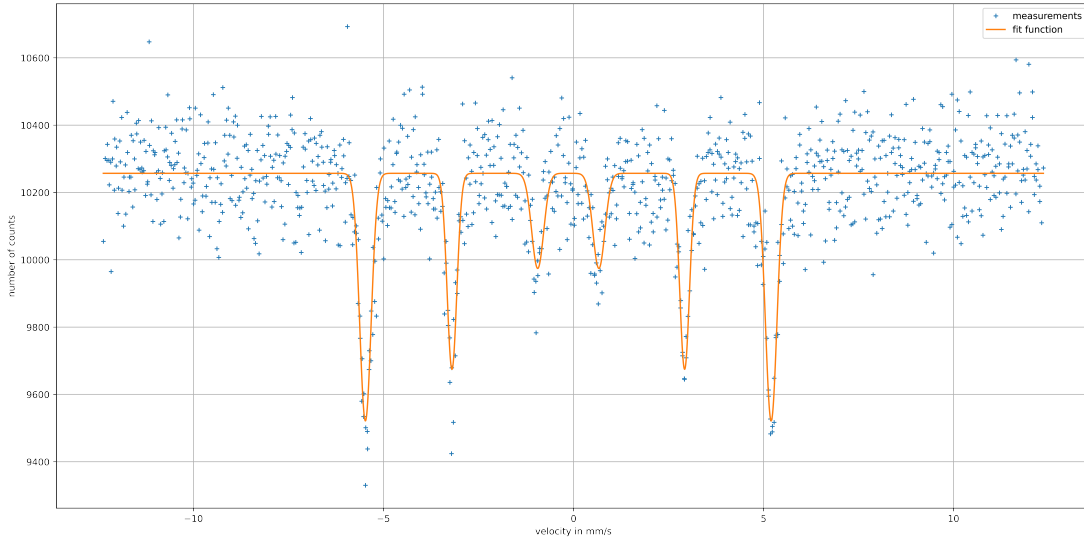
param, cov = curve_fit(fit_func_fe, df_fe['vel'].to_numpy(), df_fe['count'].
    ↪to_numpy(), sigma=np.sqrt(df_fe['count'].to_numpy()), p0=guess_params,
    ↪maxfev=10000)
print('Converging parameters: ', param)

# calculate to draw fit-function
fit_vel = np.linspace(df_fe['vel'].min(), df_fe['vel'].max(), int(1e5))
fit_rate = fit_func_fe(fit_vel, *param)

plt.figure(figsize=(20,10), dpi=500)
plt.plot(1000*df_fe['vel'], df_fe['count'], label='measurements',
    ↪linestyle='None', marker='+', markersize=5)
plt.plot(1000*fit_vel, fit_rate, label='fit function')
plt.grid()
plt.legend(loc='best')
plt.xlabel('velocity in mm/s')
plt.ylabel('number of counts')
plt.show()
```

Starting parameters: [10211.33203125, -0.0001, 400, 0.0008, 0.0001, 600, 0.003, 0.0001, 800, 0.0055, 0.0001]

Converging parameters: [ 1.02569346e+04 -1.36303705e-04 2.82646639e+02  
8.04617420e-04  
1.58859038e-04 5.82252491e+02 3.06055312e-03 1.27779318e-04  
7.35572006e+02 5.33831844e-03 1.50873933e-04]



With the fit done, the parameters which the fit delivered can be used for further analysis. The offset and the amplitudes  $a_i$  and standard deviations  $c_i$  are not relevant, only the isoshift and the positions of the peaks  $b_i$  (already corrected for the isoshift) will be used.

```
[ ]: isoshift = ufloat(param[1], sqrt(cov[1][1]))
      # Will order the velocities and energys now in descending order
      b_3 = ufloat(param[3], sqrt(cov[3][3]))
      b_2 = ufloat(param[6], sqrt(cov[6][6]))
      b_1 = ufloat(param[9], sqrt(cov[9][9]))
      fe_E_isoshift = doppler_delta_E_from_v(isoshift)
      fe_E_1 = doppler_delta_E_from_v(b_1)
      fe_E_2 = doppler_delta_E_from_v(b_2)
      fe_E_3 = doppler_delta_E_from_v(b_3)

      print('Isometry shift: ', isoshift*1000, 'mm/s or ', fe_E_isoshift, 'eV')
      print('Positon b_1: +/-', b_1*1000, 'mm/s or +/-', fe_E_1, 'eV')
      print('Positon b_2: +/-', b_2*1000, 'mm/s or +/-', fe_E_2, 'eV')
      print('Positon b_3: +/-', b_3*1000, 'mm/s or +/-', fe_E_3, 'eV')
```

```
Isometry shift: -0.136+/-0.005 mm/s or (-6.54+/-0.23)e-09 eV
Positon b_1: +/- 5.338+/-0.007 mm/s or +/- (2.5624+/-0.0032)e-07 eV
Positon b_2: +/- 3.061+/-0.008 mm/s or +/- (1.469+/-0.004)e-07 eV
Positon b_3: +/- 0.805+/-0.018 mm/s or +/- (3.86+/-0.09)e-08 eV
```

From there, the four conditions for  $A$  and  $G$  need to be checked.

```
[ ]: # One of these four conditions should be (close to) 0
fe_C_1 = fe_E_1 - (2 * fe_E_2) - fe_E_3
fe_C_2 = fe_E_1 - (2 * fe_E_2) + fe_E_3
fe_C_3 = fe_E_1 - fe_E_2 - (2 * fe_E_3)
fe_C_4 = fe_E_1 + fe_E_2 + (2 * fe_E_3)
print("Condition 1 is ", fe_C_1, "eV (valid if 0)")
print("Condition 2 is ", fe_C_2, "eV (valid if 0)")
print("Condition 3 is ", fe_C_3, "eV (valid if 0)")
print("Condition 4 is ", fe_C_4, "eV (valid if 0)")
```

Condition 1 is (-7.62+/-0.12)e-08 eV (valid if 0)  
Condition 2 is (1.0+/-1.2)e-09 eV (valid if 0)  
Condition 3 is (3.21+/-0.18)e-08 eV (valid if 0)  
Condition 4 is (4.804+/-0.018)e-07 eV (valid if 0)

It looks like condition 2 is satisfied. So we have the case

$$A > 0; \quad G < 0; \quad |A| < |G|.$$

Now,  $A$  and  $G$  can be calculated

```
[ ]: # Calculations for case 2
fe_A = fe_E_1 - fe_E_2
fe_G = - fe_E_2 - fe_E_3
print("A =", fe_A, "eV")
print("G =", fe_G, "eV")
```

$A = (1.093+/-0.005)e-07$  eV  
 $G = (-1.855+/-0.009)e-07$  eV

Using  $G$  and the given  $\mu_g$ , the magnetic field  $B$  can be calculated.

```
[ ]: I_g = 1/2 # nuclear angular momentum quantum number of ground state
mu_k = 3.1524512550e-8 # eV / T
mu_g = ufloat(0.0903, 0.0007) * mu_k
B = fe_G * I_g / mu_g
print("Magnetic field B =", B, "T")
```

Magnetic field  $B = -32.59+/-0.30$  T

The literature value for  $B$  is about 33 T which was confirmed in this measurement. With knowledge about the magnetic field  $B$ ,  $\mu_a$  can finally be calculated.

```
[ ]: I_a = 3/2 # nuclear angular momentum quantum number of excited state
mu_a = fe_A * I_a / B
print("mu_a = ", mu_a, "eV/T = ", (mu_a / mu_k), "mu_k = ", (mu_a / mu_g),
      "\n      ↳ mu_g")
```

$\mu_a = (-5.03+/-0.06)e-09$  eV/T =  $-0.1596+/-0.0017$   $\mu_k = -1.768+/-0.014$   $\mu_g$

It is rather interesting to see  $\mu$  flip its sign when transitioning from the ground state  $\mu_g$  to the excited state  $\mu_a$ .

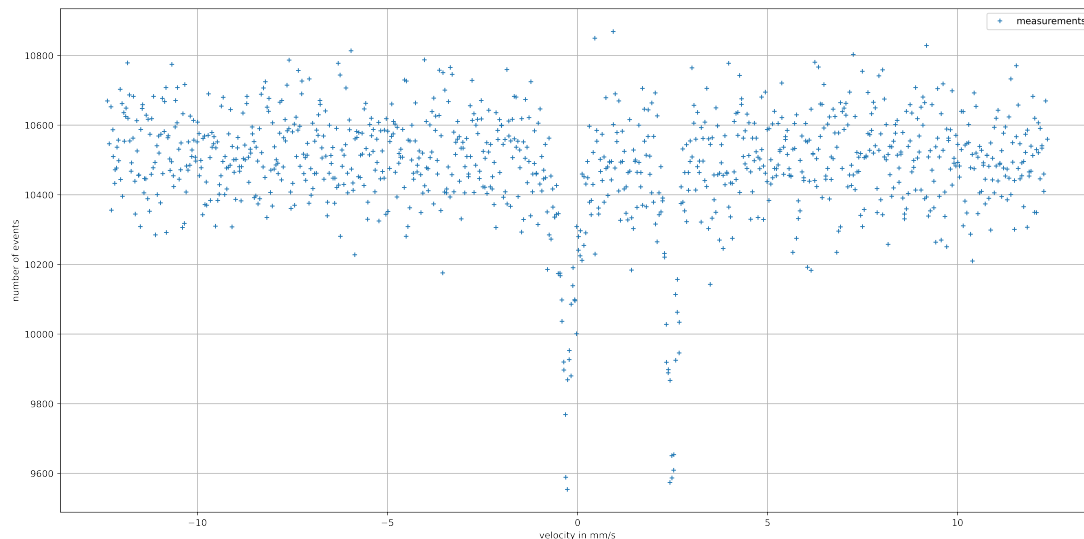
## 1.6 Chapter 6 - Analysis of FeSO4

Next, the analysis of FeSO4 comes in. Again, the first step is reading in the data.

```
[ ]: feso4_file = np.genfromtxt('../data/FeSO4.txt')
df_feso4 = pd.DataFrame({'chn':feso4_file[:,0], 'count':feso4_file[:,1]}, dtype=int)
df_feso4['vel'] = vel_cal(df_feso4['chn'])
print(df_feso4)
plt.figure(figsize=(20,10), dpi=500)
plt.plot(1000*df_feso4['vel'], df_feso4['count'], label='measurements',
        linestyle='None', marker='+', markersize=5)
plt.grid()
plt.legend(loc='best')
plt.xlabel('velocity in mm/s')
plt.ylabel('number of events')
plt.show()
```

	chn	count	vel
0	1	10560	0.012364
1	2	10670	0.012316
2	3	10460	0.012268
3	4	10534	0.012219
4	5	10446	0.012171
...	...	...	...
1019	1020	10530	0.012076
1020	1021	10607	0.012124
1021	1022	10591	0.012173
1022	1023	10541	0.012221
1023	1024	10410	0.012269

[1024 rows x 3 columns]



The data clearly shows two dips caused by resonance absorption. So the fit function will be two Gaussian functions subtracted from a constant offset. The fit function is identical to the one from iron, besides two dips instead of six.

```
[ ]: # more complex fit function
def fit_func_feso4(x, offset, isoshift, a, b, c):
    # offset is base rate
    # isoshift is shifted velocity 0
    # then, three pairs of gaussian distributions are subtracted, symmetrically
    ↪ around the isoshift.
    y = offset
    y -= single_gauss(x, a, isoshift+b, c)
    y -= single_gauss(x, a, isoshift-b, c)
    return y

guess_params=[df_feso4['count'].mean(), 0.001, 1800, 0.001, 0.0001]
print('Starting parameters: ', guess_params)

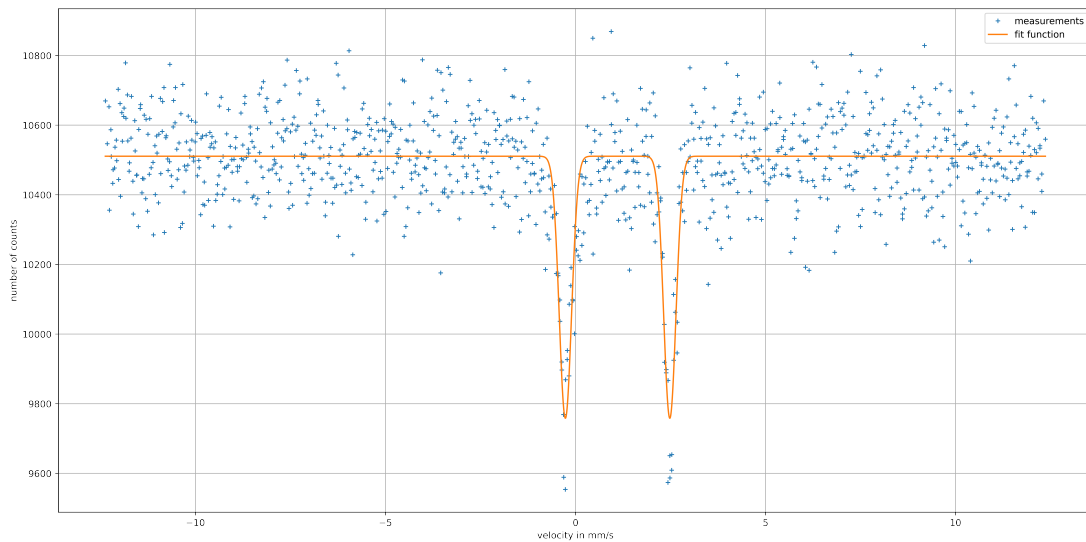
param, cov = curve_fit(fit_func_feso4, df_feso4['vel'].to_numpy(),
    ↪ df_feso4['count'].to_numpy(), sigma=np.sqrt(df_feso4['count'].to_numpy()),
    ↪ p0=guess_params, maxfev=10000)
print('Converging parameters: ', param)

# calculate to draw fit-function
fit_vel = np.linspace(df_feso4['vel'].min(), df_feso4['vel'].max(), int(1e5))
fit_rate = fit_func_feso4(fit_vel, *param)

plt.figure(figsize=(20,10), dpi=500)
```

```
plt.plot(1000*df_feso4['vel'], df_feso4['count'], label='measurements',
        linestyle='None', marker='+', markersize=5)
plt.plot(1000*fit_vel, fit_rate, label='fit function')
plt.grid()
plt.legend(loc='best')
plt.xlabel('velocity in mm/s')
plt.ylabel('number of counts')
plt.show()
```

Starting parameters: [10487.2958984375, 0.001, 1800, 0.001, 0.0001]  
 Converging parameters: [1.05108131e+04 1.10828055e-03 7.52460939e+02  
 1.37390054e-03  
 1.62162410e-04]



The fit describes the data quite well. Again, only the isoshift and the position  $b$  are relevant,  $a$  and  $c$  are not relevant.

```
[ ]: isoshift = ufloat(param[1], sqrt(cov[1][1]))
      b = ufloat(param[3], sqrt(cov[3][3]))
      feso4_E_isoshift = doppler_delta_E_from_v(isoshift)
      feso4_delta_E = doppler_delta_E_from_v(b)

      print('Isometry shift: ', isoshift*1000, 'mm/s or ', feso4_E_isoshift, 'eV')
      print('Position b: +/-', b*1000, 'mm/s or +/-', feso4_delta_E, 'eV')
```

Isometry shift: 1.108 $\pm$ 0.007 mm/s or (5.320 $\pm$ 0.034)e-08 eV  
 Position b:  $\pm$  1.374 $\pm$ 0.007 mm/s or  $\pm$  (6.595 $\pm$ 0.034)e-08 eV

From the preparation, it is known that the energy shift  $\Delta E$  is

$$\Delta E_Q(m) = \frac{eQ}{4} \left( \frac{\partial^2 V}{\partial z^2} \right) \frac{3m^2 - I(I+1)}{3I^2 - I(I+1)}.$$

Here,  $Q$  is the known quadrupole moment,  $e$  is the elementary charge and  $V$  the electric field. The quantum number  $I$  has to be  $3/2$ , as the state  $I = 1/2$  does not show quadrupole splitting. Because of that the magnetic quantum number  $m$  can take the values  $\pm 3/2$  and  $\pm 1/2$ ; as  $m$  is squared, the sign does not matter. For  $m = \pm 3/2$ , the last fraction becomes  $+1$ ; for the case  $m = \pm 1/2$ , the last fraction is  $-1$ .

The fit already takes care of the  $\pm$ -symmetry and returns only one  $\Delta E$  for FeSO<sub>4</sub>. This can be resolved for the unknown gradient  $V_{zz}$

$$\Delta E = \frac{eQ}{4} \left( \frac{\partial^2 V}{\partial z^2} \right) = \frac{eQ}{4} V_{zz} \implies V_{zz} = \frac{4 \cdot \Delta E}{eQ}.$$

```
[ ]: Q = ufloat(0.21e-28, 0.01e-28) # in m^2; value known
      # Division by e makes eV to V
      V_zz = 4 * feso4_delta_E / Q
      print("Gradient of electric field: V_zz =", V_zz, "V/m^2")
```

Gradient of electric field: V\_zz = (1.26+/-0.06)e+22 V/m<sup>2</sup>

Finally, it is possible to calculate the wanted field gradient.

## 1.7 Chapter 7 - Analysis of FePO<sub>4</sub>

The last measured sample was FePO<sub>4</sub> which is chemically very similar to the last analyzed sample, FeSO<sub>4</sub>. Again, at first the data need to be read in.

```
[ ]: fepo4_file = np.genfromtxt('../data/FePO4.txt')
      df_fepo4 = pd.DataFrame({'chn':fepo4_file[:,0], 'count':fepo4_file[:,
      ↪,1]}, dtype=int)
      df_fepo4['vel'] = vel_cal(df_fepo4['chn'])
      print(df_fepo4)
      plt.figure(figsize=(20,10), dpi=500)
      plt.plot(1000*df_fepo4['vel'], df_fepo4['count'], label='measurements',
      ↪ linestyle='None', marker='+', markersize=5)
      plt.grid()
      plt.legend(loc='best')
      plt.xlabel('velocity in mm/s')
      plt.ylabel('number of events')
      plt.show()
```

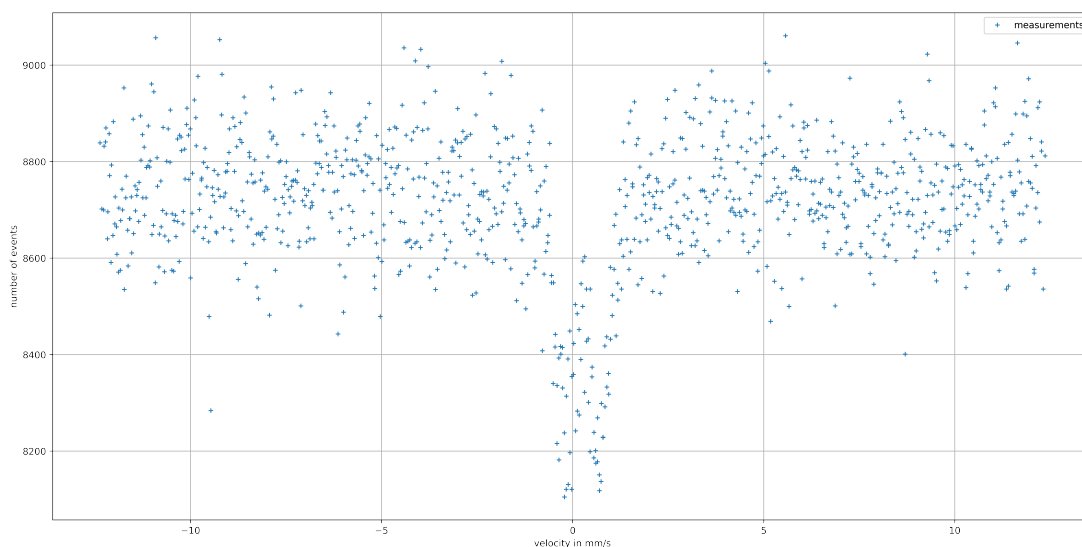
	chn	count	vel
0	1	8812	0.012364
1	2	8536	0.012316
2	3	8841	0.012268
3	4	8924	0.012219

```

4          5    8736  0.012171
...      ...      ...
1019  1020    8569  0.012076
1020  1021    8792  0.012124
1021  1022    8912  0.012173
1022  1023    8675  0.012221
1023  1024    8822  0.012269

```

[1024 rows x 3 columns]



Just as FeSO<sub>4</sub>, FePO<sub>4</sub> has two dips of resonance absorption caused by quadrupole splitting. However, with FePO<sub>4</sub> the dips are much closer, making the fitting more difficult.

```

[ ]: # more complex fit function
def fit_func_fepo4(x, offset, isoshift, a, b, c):
    # offset is base rate
    # isoshift is shifted velocity 0
    # then, three pairs of gaussian distributions are subtracted, symmetrically
    ↪ around the isoshift.
    y = offset
    y -= single_gauss(x, a, isoshift+b, c)
    y -= single_gauss(x, a, isoshift-b, c)
    return y

guess_params=[df_fepo4['count'].mean(), 0.0005, 400, 0.0005, 0.0001]
print('Starting parameters: ', guess_params)

```



```

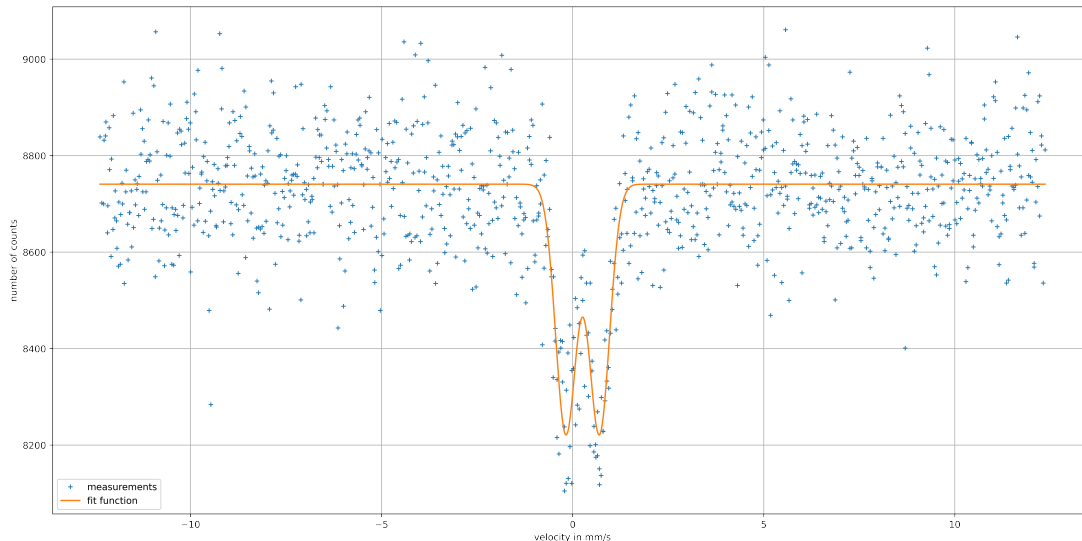
param, cov = curve_fit(fit_func_fepo4, df_feso4['vel'].to_numpy(),
↳df_fepo4['count'].to_numpy(), sigma=np.sqrt(df_fepo4['count'].to_numpy()),
↳p0=guess_params, maxfev=10000)
print('Converging parameters: ', param)

# calculate to draw fit-function
fit_vel = np.linspace(df_fepo4['vel'].min(), df_fepo4['vel'].max(), int(1e5))
fit_rate = fit_func_fepo4(fit_vel, *param)

plt.figure(figsize=(20,10), dpi=500)
plt.plot(1000*df_fepo4['vel'], df_fepo4['count'], label='measurements',
↳linestyle='None', marker='+', markersize=5)
plt.plot(1000*fit_vel, fit_rate, label='fit function')
plt.grid()
plt.legend(loc='best')
plt.xlabel('velocity in mm/s')
plt.ylabel('number of counts')
plt.show()

```

Starting parameters: [8713.958984375, 0.0005, 400, 0.0005, 0.0001]  
Converging parameters: [8.74089018e+03 2.60800733e-04 5.17052252e+02  
4.37100459e-04  
2.68756348e-04]



With good starting parameters, the fit function describes the measured values quite well. The following analysis is completely identical to that of FeSO<sub>4</sub>.

```
[ ]: isoshift = ufloat(param[1], sqrt(cov[1][1]))
      b = ufloat(param[3], sqrt(cov[3][3]))
      fepo4_E_isoshift = doppler_delta_E_from_v(isoshift)
      fepo4_delta_E = doppler_delta_E_from_v(b)

      print('Isometry shift: ', isoshift*1000, 'mm/s or ', fepo4_E_isoshift, 'eV')
      print('Positon b: +/-', b*1000, 'mm/s or +/-', fepo4_delta_E, 'eV')

      Q = ufloat(0.21e-28, 0.01e-28) # in m2; value known
      # Division by e makes eV to V
      V_zz = 4 * fepo4_delta_E / Q
      print("Gradient of electric field: V_zz =", V_zz, "V/m2")
```

```
Isometry shift: 0.261+/-0.015 mm/s or (1.25+/-0.07)e-08 eV
Positon b: +/- 0.437+/-0.011 mm/s or +/- (2.10+/-0.05)e-08 eV
Gradient of electric field: V_zz = (4.00+/-0.22)e+21 V/m2
```

At last, the field gradient for FePO<sub>4</sub> can be calculated. As expected it is smaller than that of FeSO<sub>4</sub> because the quadrupole splitting is much less severe with FePO<sub>4</sub> than it is with FeSO<sub>4</sub>.

# Literaturverzeichnis

- [1] NASA, “Mössbauer spectrometer (mb).” Website. <https://mars.nasa.gov/mer/mission/instruments/mb/>; letzter Zugriff am 27. März 2022.
- [2] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, pp. 357–362, Sept. 2020.
- [3] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [4] Wes McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.
- [5] F. K. Schmidt and J. Wolf, *Einführung in das Kern- und Teilchenphysikalische Praktikum*. 2021.
- [6] M. R. Bhat, “Nuclear data sheets for a = 57 \*,” 1998. DOI: <https://doi.org/10.1006/ndsh.1998.0021>.
- [7] E. W. Weisstein, “Autocorrelation.” Website. From MathWorld—A Wolfram Web Resource. <https://mathworld.wolfram.com/Autocorrelation.html>; letzter Zugriff am 31. März 2022.
- [8] A. Becker, “Digitale Datenerfassung und Verarbeitung mit FPGA für das Fortgeschrittenpraktikum.” Bachelorarbeit, 2019.
- [9] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Í. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.
- [10] G. Quast, “Phyprakit.” Python-Library, 2022. DOI: <https://doi.org/10.5281/zenodo.6433316>.
- [11] S. Tomar, “Converting video formats with ffmpeg,” *Linux Journal*, vol. 2006, no. 146, p. 10, 2006.