## Fortgeschrittenenpraktikum

# Messung des Fermi-Impulses durch die Winkelkorrelation von $\gamma$-Strahlung aus der Annihilation von Elektron-Positron-Paaren


**Verfasser:** Eduard Koller (03702415)
Michael Labenbacher (03697519)
Anna Oberbauer (03670019)

**Gruppe:** 97

**Betreuer:** Josef Ketels



Versuchsdatum: 17. Juni 2019
Abgabedatum: 21. Juli 2019

## INHALTSVERZEICHNIS

**1. Einführung und physikalischer Background** **1**
1.1. Positronenquelle und Verhalten von Positronen im Festkörper . . 1
1.2. Ideales Fermigas . . . . . 1

**2. Experimentelle Methoden** **2**
2.1. ACAR-Spektroskopie . . . . . 2
2.2. Versuchsaufbau . . . . . 2

**3. Experimentelles Vorgehen** **3**

**4. Auswertung und Ergebnisse** **4**

**5. Zusammenfassung** **4**

**Verzeichnisse** **6**

**A. Herleitung der Winkelabweichung** **8**

**B. Fit-Parameter** **9**

**C. Python-Code** **9**

## 1. EINFÜHRUNG UND PHYSIKALISCHER BACKGROUND

Ziel ist die Messung des Fermi-Impulses einer polykristallinen Kupferprobe. Dieser lässt sich aus der Fermi-Fläche extrahieren, deren Projektion mit Hilfe einer ein dimensionalen ACAR[1]-Spektroskopie bestimmt wird. Hierbei wird die Winkelkorrelation von $\gamma$-Strahlung aus der Annihilation von Elektron-Positron-Paaren in der Probe untersucht.

### 1.1. POSITRONENQUELLE UND VERHALTEN VON POSITRONEN IM FESTKÖRPER

Als Positronenemitter in diesem Versuch dient eine $^{22}$Na-Quelle, deren relativ große Halbwertszeit, auf eine, im Sinne des Experiments, konstante Rate führt. Die Positronenaktivität ist groß genug und das sich beim $\beta^+$-Zerfall ergebende Spektrum kontinuierlich mit einer Endpunktenergie von ca. 545 keV. Eine weitere Möglichkeit nutzbare Positronen zu erzeugen ist die Paarbildung, wobei die benötigten hochenergetischen $\gamma$-Quanten durch Bremsstrahlung oder in Reaktorquellen, durch Neutroneneinfang, z.B. NEPOMUC[2] gewonnen werden. Dabei ergibt sich ein Spektrum mit scharfer Bandbreite und ermöglicht es in bestimmten Tiefen zu messen und tiefer ins Material einzudringen, was in diesem Versuch untergeordnete Relevanz hat.

Trifft ein Positron auf einen Festkörper (die Probe), so kommt es zu verschiedenen Prozessen. Ein wichtiger ist die *Thermalisierung*, Abbremsung des Teilchens im Medium durch inelastische Stöße bis sich dieses im thermischen Gleichgewicht befindet. Nach dem Energieverlust auf $\frac{3}{2}\mathcal{O}(25\,\mathrm{meV})$ diffundiert das Positron innerhalb seiner Lebensdauer frei im Festkörper umher. Wird das Positron durch eine in der Probe vorliegende Haft- bzw. Defektstelle *eingefangen*, so lokalisiert sich dort das Wellenpaket und das attraktive Potential macht ein, im klassischen Sinne gesehen, Entkommen unmöglich und erhöht die Lebensdauer der Positronen. Dieser Effekt ist in diesem Versuch unerwünscht, da der ungestörte Kristall untersucht werden soll, womit die verwendete Kupferprobe dementsprechend wenig Defekte aufweisen muss. Trifft ein Positron auf ein Elektron im Festkörper, so *annihiliert* es in den meisten Fällen nach einer mittleren Lebensdauer von $\mathcal{O}(100\,\mathrm{ps})$ zu zwei $\gamma$-Quanten. Auf Grund der Viererimpulserhaltung im Schwerpunktsystem mit jeweils 511 keV im Winkel von 180°. Mit Hilfe der Winkelkorrelation dieser beiden $\gamma$-Quanten im Laborsystem kann auf den Fermi-Impuls geschlossen werden. Notwendig dafür ist, dass die Prozessdauer der Abbremsung auf thermische Energien gering ist und da typische Energien der Valenzelektronen, Fermi-Energien, in der Größenordnung von $\mathcal{O}(10\,\mathrm{eV})$ liegen, die Positronen als in Ruhe angenommen werden können.

### 1.2. IDEALES FERMIGAS

Zur Beschreibung der Valenzelektronen in polykristallinem Kupfer wird das Modell des freien Elektronengases verwendet. Dabei wird die Wechselwirkung zwischen den Elektronen vernachlässigt und nach dem Pauli-Verbot werden die Zustände von unten bis zur Fermi-Energie $E_\mathrm{F}$ aufge-

[1] **A**ngular **C**orrelation of **A**nnihilation **R**adiation

[2] **NE**utron induced **PO**sitronsource **MU**ni**C**h

füllt. Mit der Dispersionsrelation $E \propto p^2/(2m_\mathrm{e})$ definiert man den Fermi-Impuls mit

$$E_\mathrm{F} = \frac{p_\mathrm{F}^2}{2m_\mathrm{e}}. \tag{1}$$

Betrachtet man die Fläche konstanter Energie, der Fermi-Energie, im Impulsraum so erhält man aus Isotropie eine Kugel und durch Projektion auf eine Ebene eine zentral invertierte Parabel. Im dreidimensionalen Potentialkasten bei einer Temperatur $T = 0\,\mathrm{K}$ ergibt sich

$$E_\mathrm{F} = \frac{\hbar^2}{2m_\mathrm{e}}(3\pi^2 n_\mathrm{e})^{\frac{2}{3}}, \tag{2}$$

mit $n_\mathrm{e}$ der Elektronendichte. Im Experiment wird Kupfer mit einer Dichte von $n_\mathrm{e} = 8.5 \cdot 10^{22}\,\mathrm{cm}^{-3}$ verwendet [1].

## 2. EXPERIMENTELLE METHODEN

### 2.1. ACAR-SPEKTROSKOPIE

Wie bereits in Abschnitt 1 erwähnt, kommt es bei der Zerstrahlung des Positrons an einem Elektron zur Annihilation zweier $\gamma$-Quanten. Die Energie der Elektronen führt auf eine Dopplerverschiebung und auf Grund von Isotropie ergibt sich eine symmetrische Verbreiterung der Annihiationslinie. Durch den Transversalimpuls senkrecht zur Emissionsrichtung der $\gamma$-Quanten geht die Kollinearität im Laborsystem verloren und führt auf eine Abweichung von 180° um

$$\Delta\theta \approx \frac{p_\perp}{m_e c}, \tag{3}$$

mit $m_e$ der Elektronenmasse und $c$ der Lichtgeschwindigkeit. Die Herleitung findet sich im Anhang A. Die Messung der Winkelabweichung erlaubt die Bestimmung des Elektronenimpulses in transversaler Richtung zum Detektor und mit dem Modell eines idealen Fermi-Gases lässt sich auf den Fermi-Impuls schließen, da die Koinzidenzzählrate $I(\Delta\theta)$ nach (3) proportional zum Schwerimpuls und damit zur eindimensionalen Projektion der Elektronenimpulsdichte ist.

Zur Bestimmung eines 1D-ACAR-Spektrums sind zwei Detektoren für die beiden $\gamma$-Quanten notwendig. Für dieses Spektroskopie-Verfahren wird eine hohe Nachweiswahrscheinlichkeit benötigt und die Energieauflösung spielt eine geringere Rolle, weshalb BGO-Kristalle für die Szintillationsdetektoren verwendet wurden. Die Messung bei unterschiedlichen Winkeln für gleiche Zeiten liefert, mit entsprechender Kalibrierung der Messelektronik sowie der Verwendung eines Zählers, die Anzahl gleichzeitig detektierter Ereignisse, welche dann direkt proportional zur eindimensionalen Projektion der Fermi-Fläche ist. Die Probe selbst befindet sich unter einer Bleiburg, zur Minimierung der Strahlungsbelastung der Experimentatoren, welche zwischen den beiden Detektoren errichtet wurde.

### 2.2. VERSUCHSAUFBAU

Der Versuchsaufbau besteht im wesentlichen aus den beiden Detektoren und NIM[3]-Komponenten zur Signalverarbeitung. Der Aufbau ist in Abbildung 1 schematisch dargestellt. Die Detektoren im Abstand $R = 1\,\mathrm{m}$ von der Probe bestehen aus je einem BGO-Kristall, welcher 82 mm hoch, 30 mm breit und 10 mm lang ist, und einem Photomultiplier im selben Wellenlängenbereich. Der Schwächungskoeffizient $\eta$ für BGO und $\gamma$-Energien von 511 keV beträgt $0.906\,\mathrm{cm}^{-1}$ [2], womit die Quantendetektionseffizienz eines Detektors durch

$$\eta = 1 - \mathrm{e}^{-\mu \cdot d}, \tag{4}$$

mit $d$ der Dicke, gegeben ist. Zusätzlich wird hinter der Probe und vor dem Detektor ein Kollimator mit einer Spaltbreite von 1 mm aufgestellt, um eine dementsprechende Winkelauflösung zu gewährleisten. Der Winkel eines Detektors wird mit Hilfe eines Schrittmotors im Abstand $R = 1.25\,\mathrm{m}$ zur Probe, der durch ein Phyton-Programm gesteuert wird, verstellt. Die Anzahl der möglichen Anordnungen des fixen Detektors beträgt $\frac{\Delta\omega}{2\pi}$, mit $\Delta\omega$ dem Raumwinkel, und die Wahrscheinlichkeit für eine Interaktion in einem Szintillator ergibt sich mit dem Absorptionskoeffizienten von BGO nach Gleichung (4). Die Messung koinzidenter Ereignisse mit zwei unabhängigen Detektoren führt auf den Zusammenhang

$$N = \lambda \cdot \eta^2 \cdot \frac{\Delta\omega}{2\pi} \tag{5}$$

[3] **N**uclear **I**nstrumentation **M**odule

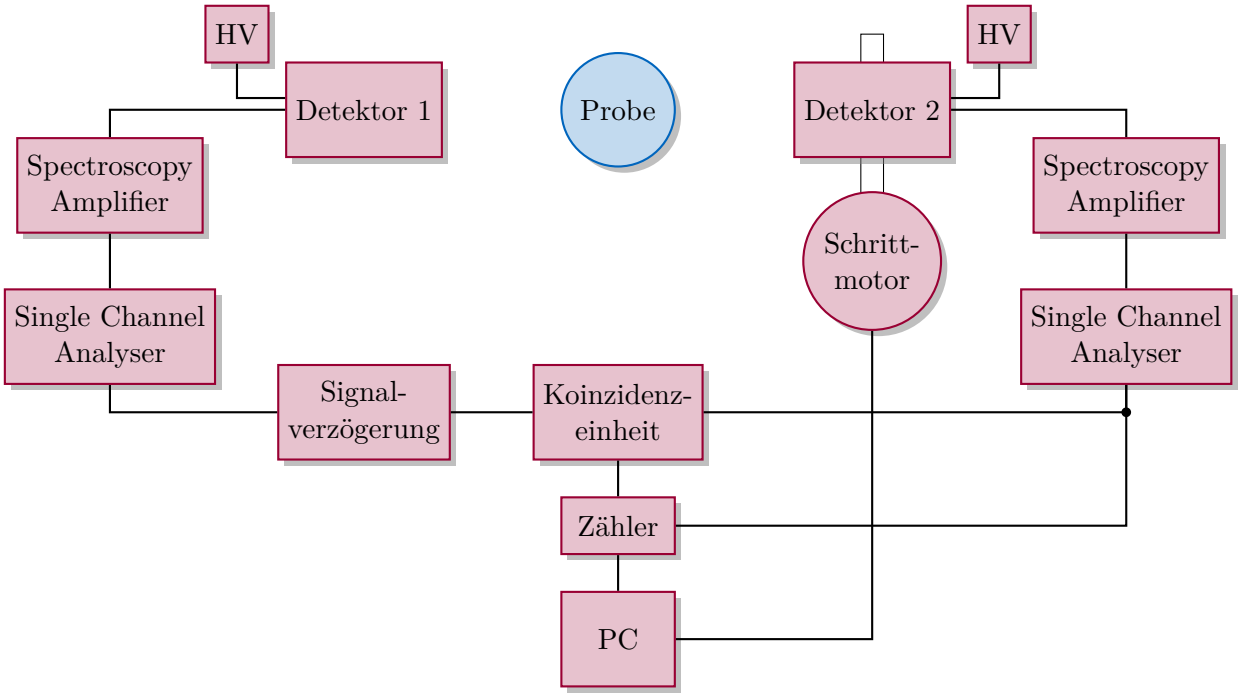


**Abbildung 1:** Schematischer Aufbau und Schaltplan des Versuchs.

der gemessenen Zählrate $N$ und Positronenannihilationsrate $\lambda$ in der Probe.

Der Spectorscopy Amplifier (SA) hinter einem Detektor verstärkt das „gepeakte“ Signal vom Detektor und wandelt es in eine Gaußkurve um, deren Amplitude proportional zur Energie des $\gamma$-Quants ist. Der Single Channel Analyser (SCA) filtert Ereignisse in einem Bereich um die gesuchte Energie heraus und gibt ein TTL-Signal auf die Koinzidenzeinheit weiter. Auf Grund unsymmetrischer Verkabelung muss zusätzlich die zeitliche Verschiebung der Signale korrigiert werden. Hierfür wird zwischen einem Single Channel Analyser und der Koinzidenzeinheit ein Linear Gate Strecher eingebaut. Die Koinzidenzeinheit überprüft anschließend die Ereignisse auf Gleichzeitigkeit und steuert einen Zähler an, welcher an den PC angeschlossen ist.

## 3. EXPERIMENTELLES VORGEHEN

Die Kalibrierung der Elektronik, skizziert in Abb. 1, erfolgt für beide Detektoren einzeln mit Hilfe eines Oszilloskop.

Das Ausgangssignal eines Detektors ist exponentiell abfallend, da die Lebensdauer angeregter Zustände im Szintillatorkristall analog zum exponentiellen Zerfallsgesetz beschrieben wird. Dieses Signal wird durch den SA in eine Gaußkurve (idealisiert) umgewandelt, das Signal dabei verstärkt und die Amplitude ist direkt proportional zu der Photonenenergie. Anschließend wird mittels des SCA's ein Energiebereich gefiltert, der gesuchte Bereich entspricht dem Photopeak bei 511 keV. Die niedrigeren Energiebereiche (Compton-Effekt) werden durch einen Bereich niedriger Intensität (Compton-Kante) vom nächsten Bereich hoher Intensität, dem Photopeak, getrennt. Der SCA gibt im Falle eines Ereignisses im richtigen Energiebereich ein Rechtecksignal aus.

Anschließend werden beide Signale mit der Koinzidenzeinheit verbunden. Dabei wird mit der Signalverzögerungs-Komponente der Zeitunterschied beider Signale ausgeglichen, so dass die beiden Rechtecksignale der SCA's übereinander liegen. Bei gleichzeitiger Detektion eines Ereignisses wird ein Impuls an den Zähler übergeben. Mithilfe eines Python-Skripts 1 werden die Daten am PC erfasst und gespeichert.

Des Weiteren wird der Ausgang des SCA's vom beweglichen Detektor mit einem separaten Eingang an den Zähler angeschlossen. Damit kann auf die Abhängigkeit der Detektionswahrscheinlichkeit von der Position rückgeschlossen und die koinzidenten Ereignisse normiert werden.

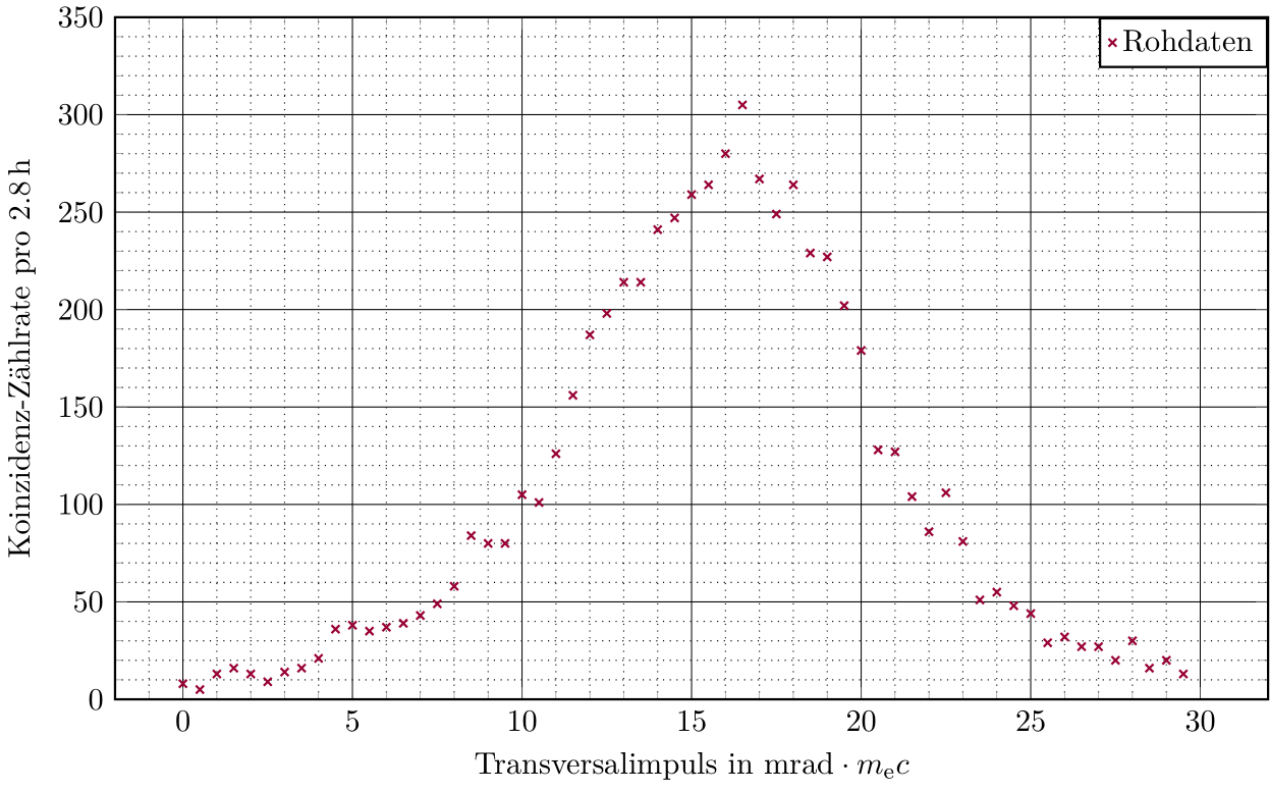


**Abbildung 2:** Rohdaten des gemessenen Acar-Spektrums von polykristallinem Kupfer. Die Anzahl der gleichzeitig detektierten Ereignisse $N$ sind gegen den Transversalimpuls $p_\perp$ aufgetragen und die Startposition wurde willkürlich auf 0 mrad festgelegt.

## 4. AUSWERTUNG UND ERGEBNISSE

Die Rohdaten des aufgezeichneten Spektrums sind in Abbildung 2 dargestellt. Es lässt sich eine Verschiebung der Symmetrieachse um ca. 2 mrad feststellen und ein Peak mit in etwa 100 gleichzeitig registrierten Ereignissen pro Stunde. Wie aus Gründen der Isotropie verlangt ist der Verlauf aus statistischer Sicht symmetrisch und die Form gleicht, neben dem überlagerten Background, einer invertierten Parabel, wie in Kapitel 1.2 für einen idealen Fermi-Körper erläutert. Die breite Kurve, die der Parabel überlagert ist, rührt von den Positronenannihilationen mit gebundenen Elektronen her und lässt sich gut mit einer Gaußkurve annähern [3].

Nach der Normierung der koinzidenten Ereignisse, wie in Kap. 3 erläutert, wurde iterativ gefittet. Dies setzt sich aus einer Zentrierung der Daten, einem Gauß-Fit mit den Datenpunkten außerhalb des Fermi-Impulses und anschließendem Fit einer Parabel innerhalb zusammen. Das Ergebnis ist der Abbildung 3 zu entnehmen und der vollständige Python-Code dem Anhang C. Aus den Fit-Parametern für die Parabel, siehe Tab. 1, und Formel (1) folgt eine Fermi-Energie von $E_\mathrm{F} = 7.189\,\mathrm{eV}$.

Integration der gefitteten Kurve in Abb. 3 über die Halbkugel resultiert in einer Zählrate pro Sekunde von $N = 3\,136$ und mit Gleichung (5) folgt eine Positronenannihilationsrate in der Probe von $\gamma = 2.78 \cdot 10^7/\mathrm{s}$.

## 5. ZUSAMMENFASSUNG

Mit der Formel (2) folgt ein theoretischer Wert der Fermi-Energie für Kupfer von $E_\mathrm{F} = 7.05\,\mathrm{eV}$. Der experimentelle Wert weicht um etwa 2 % davon ab, wobei statistische Schwankungen durch die relativ lange Messzeit und im Fit berücksichtigt sind. Systematische Abweichungen vom Aufbau, der Abstand des Schrittmotors, des Detektors (und geradlinigen Bewegung des Detektors) und der Winkelverstellung, überwiegen und führen auf einen Fehler von $\approx 0.1\,\mathrm{eV}$, abgeschätzt durch den Detektorabstand $\Delta R = \pm 1\,\%$ und ei-

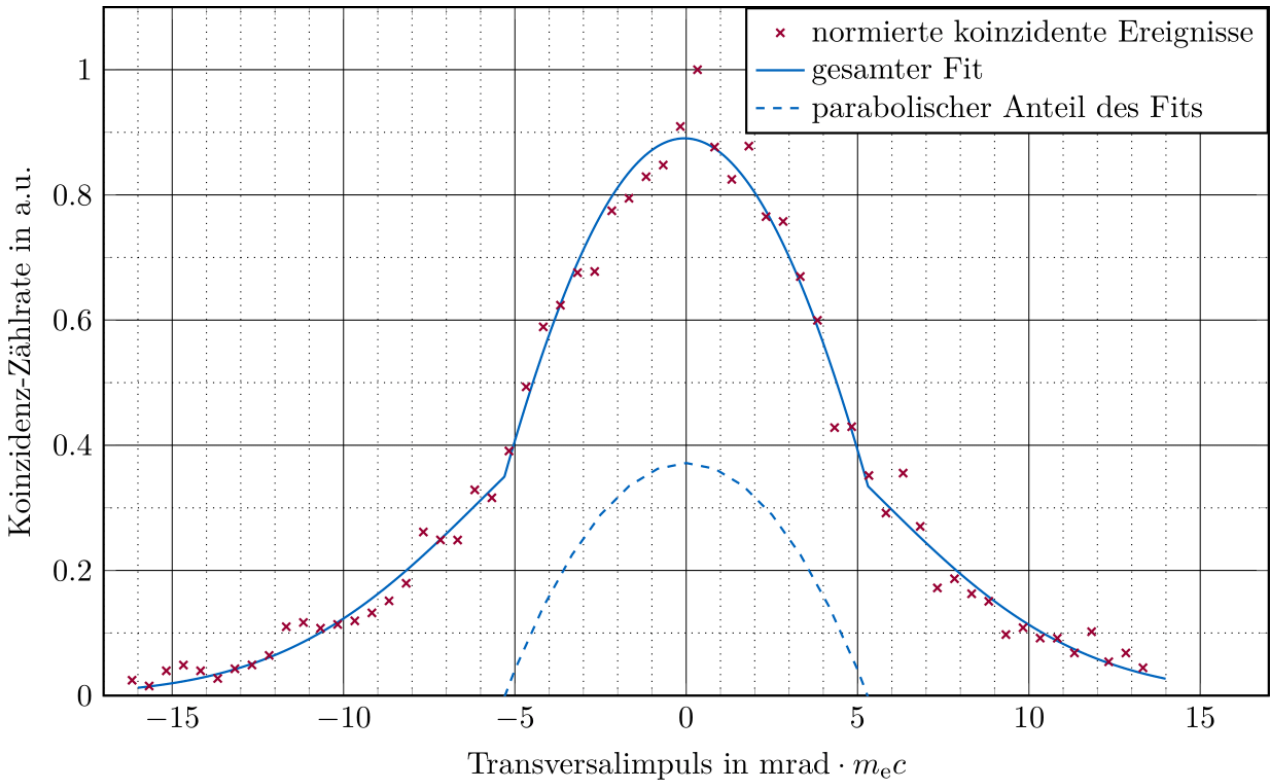


**Abbildung 3:** Normierte koinzidente Ereignisse des gemessenen Acar-Spektrums von polykristallinem Kupfer und zusammengesetzter Fit aus einer Gaußkurve und Parabel.

nem Fehler bei der Winkeländerung von 1 %. Des Weiteren wurde das Experiment bei Raumtemperaturen durchgeführt, was aber bei Metallen, keinen nennenswerten Einfluss zeigt und sich die Fermi-Energie dadurch verringern würde. Zusammengefasst zeigt sich, dass die Anwendung des Fermigas-Modells für polykristallines Kupfer eine gute Approximation für die Fermi-Energie liefert, sofern die Anzahl an Defektstellen gering ist. Der Einfluss von Defekten, Orten mit verringerter Leitungselektronendichte, ist die Erhöhung der Positronenlebensdauer, erläutert in Kap. 1.1, und der Fermi-Impuls nimmt lokal ab, womit sich die Parabel in Abb. 3 verschmälert.

Die in Kap. 1.2 angegebene Elektronendichte für Kupfer lässt sich (z.B.) aus dem Gitterabstand von $a = 0.359\,6\,\mathrm{nm}$ berechnen [4]. Die Elektronenkonfiguration ist $[\mathrm{Ar}]3d^{10}4s$, Kupfer hat fcc-Struktur und vier Kupferatome pro Elementarzelle mit je einem Leitungselektron, womit folgt $n_\mathrm{e} = \frac{4}{a^3}$.

Bei der in Kap. 4 abgeschätzten Annihilationsrate handelt es sich um die 511 keV-Annihilationen, welche den überwiegenden Anteil bilden. Der theoretisch erwartete Wert liegt bei $6.35 \cdot 10^8/\mathrm{s}$ für die Annihilationsrate für nicht wechselwirkende freie Elektronen, nach $\lambda_0 = \pi r_0^2 c n_\mathrm{e}$ berechnet, mit $r_0$ dem klassischen Elektronenradius, wobei dieser Wert um ca. eine Größenordnung zu klein ist, wegen der Elektron-Positron-Wechselwirkung [3]. Ein Vergleich zeigt damit Abweichungen zum Messwert von $\mathcal{O}(10^2)$. Ein Grund dafür liegt in der Messmethode, da wenn der Energiebereich, in dem die SCA's filtern, zu strikt eingestellt ist, zu wenige koinzidente Ereignisse detektiert werden.

## LITERATURVERZEICHNIS


[1] Universität Hamburg. *Struktur der Materie. Festkörperphysik*. Universität Hamburg, 2014. URL: https://uniexp.desy.de/sites/site_uniexp/content/e47434/e229885/e232348/sdm-fk.pdf (besucht am 10. Juli 2019) (siehe S. 2).

[2] Christoph Pascal Hugenschmidt. *Userguide-86.de*. 1994. URL: https://www.ph.tum.de/academics/org/labs/fopra/docs/userguide-86.de.pdf (besucht am 12. Juli 2019) (siehe S. 2).

[3] Günter Dlubek. „Positronenannihilation“. In: *Ausgewählte Untersuchungsverfahren in der Metallkunde*. Vienna: Springer Vienna, 1983, S. 266–287. ISBN: 978-3-7091-9503-1 (siehe S. 4–5).

[4] Philipp Haas, Fabien Tran und Peter Blaha. *Calculation of the lattice constant of solids with semilocal functionals*. 2009. URL: https://pdfs.semanticscholar.org/188c/7de160c4c419b28547f7445cceb2fab843a2.pdf (besucht am 14. Juli 2019) (siehe S. 5).


## ABBILDUNGSVERZEICHNIS


1. Schematischer Aufbau und Schaltplan des Versuchs. . . . . 3
2. Rohdaten des gemessenen Acar-Spektrums von polykristallinem Kupfer. . . . . 4
3. Normierte koinzidente Ereignisse des gemessenen Acar-Spektrums von polykristallinem Kupfer. . . 5

## TABELLENVERZEICHNIS

1. Fit-Parameter für den Fit in Abbildung 3. . . . . 9

## LISTINGVERZEICHNIS

1. Python-Code für die Bewegung des Detektors, Datenaufnahme und -speicherung. . . . . 9
2. Python-Code für die Auswertung der Daten, Fits in Abb. 3 und Berechnungen. . . . . 10

## A. HERLEITUNG DER WINKELABWEICHUNG

Eine Lorentz-Transformation vom CMS[4]-System mit Geschwindigkeit des Schwerpunkts des Elektron-PositronPaares $v$ in das Labor-System kann mit

$$\begin{pmatrix} E \\ p_\parallel \\ p_\perp \end{pmatrix} = \begin{pmatrix} \gamma & \gamma\beta & 0 \\ \gamma\beta & \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} E^{\mathrm{CM}} \\ p_\parallel^{\mathrm{CM}} \\ p_\perp^{\mathrm{CM}} \end{pmatrix}, \qquad \gamma = \frac{1}{\sqrt{1-\beta^2}}, \qquad \beta = \frac{v}{c} \tag{6}$$

geschrieben werden. Im CMS-System ist die Energie der beiden $\gamma$-Quanten gleich $E_i^{\mathrm{CM}} = \frac{1}{2}Mc^2$, mit $i \in \{1,2\}$ und $M = 2m_{\mathrm{e}}$. Die Photonen besitzen eine Ruhemasse von Null, also $p_i^{\mathrm{CM}}c = E_i^{\mathrm{CM}}$ und im CMS hat wegen der Impulserhaltung z.B. das Photon 1 den Winkel $\theta_1^{\mathrm{CM}}$ und Photon 2 somit $\pi - \theta_1^{\mathrm{CM}} = \theta_2^{\mathrm{CM}}$. Unter Ausnutzung der Invarianz des Skalarproduktes von Vierer-Vektoren folgt für die beiden Photonen

$$p_1 \cdot p_2 = \frac{E_1 E_2}{c^2}(1 - \cos(\alpha)) = p_1^{\mathrm{CM}} \cdot p_2^{\mathrm{CM}} = \frac{E_1^{\mathrm{CM}} E_2^{\mathrm{CM}}}{c^2} - \mathbf{p}_1^{\mathrm{CM}} \mathbf{p}_2^{\mathrm{CM}} = \frac{(Mc)^2}{2},$$

mit $\alpha = \pi - \Delta\theta$ dem Winkel im Laborsystem zwischen den beiden Photonen, welcher eine kleine Abweichung von $\pi$ zeigt. Mit Taylor-Entwicklung um Null für $\Delta\theta$ folgt:

$$\begin{aligned} &\Rightarrow 1 - \cos(\alpha) = \frac{M^2c^4}{2E_1E_2} \\ &\Rightarrow \cos(\Delta\theta) = \frac{M^2c^4}{2E_1E_2} - 1 \approx 1 - \frac{\Delta\theta^2}{2} \\ &\Rightarrow \Delta\theta^2 = 4 - \frac{M^2c^4}{E_1E_2} \end{aligned}$$

Rücktransformation mit Gleichung (6) führt auf:

$$\begin{aligned} E_1 &= \gamma E_1^{\mathrm{CM}} + \gamma\beta p_{1,\parallel}^{\mathrm{CM}} = \gamma\frac{Mc^2}{2}(1 + \beta\cos(\theta_1^{\mathrm{CM}})) \\ E_2 &= \gamma E_2^{\mathrm{CM}} + \gamma\beta p_{2,\parallel}^{\mathrm{CM}} = \gamma\frac{Mc^2}{2}(1 - \beta\cos(\theta_1^{\mathrm{CM}})) \end{aligned}$$

und mit der nicht-relativistischen Näherung für den Schwerpunktimpuls $\gamma^2\beta^2 \approx \frac{p^2}{M^2c^2}$ und $\gamma^2 \approx 1 + \frac{p^2}{M^2c^2}$ folgt:

$$\begin{aligned} E_1 &= E + \Delta E \\ E_2 &= E - \Delta E \end{aligned} \quad \text{mit } E = m_{\mathrm{e}}c^2, \Delta E = \frac{pc}{2}\cos(\theta_1^{\mathrm{CM}}) = \frac{p_\parallel c}{2}$$

$$\begin{aligned} \Rightarrow E_1E_2 = E^2 - \Delta E^2 &= \frac{M^2c^4}{4}\gamma^2(1 + \beta\cos(\theta_1^{\mathrm{CM}}))(1 - \beta\cos(\theta_1^{\mathrm{CM}})) = \frac{M^2c^4}{4}\gamma^2(1 + \beta^2\cos^2(\theta_1^{\mathrm{CM}})) \\ &= \frac{M^2c^4}{4}(1 + \frac{p^2}{M^2c^2}\sin^2(\theta_1^{\mathrm{CM}}))) = \frac{M^2c^4}{4} + \frac{p^2c^2}{4}\sin^2(\theta_1^{\mathrm{CM}}) \end{aligned}$$

[4] **C**enter of **M**ass **S**ystem

Damit folgt für die Winkelabweichung im Laborsystem:

$$\Delta\theta^2 = \frac{4E_1E_2/c^2 - M^2c^2}{M^2c^2/4 + p^2\sin^2(\theta_1^{\mathrm{CM}})/4} = \frac{4p^2\sin^2(\theta_1^{\mathrm{CM}})}{M^2c^2 + p^2\sin^2(\theta_1^{\mathrm{CM}})} \approx \frac{4p^2\sin^2(\theta_1^{\mathrm{CM}})}{M^2c^2} = \frac{p_\perp^2}{m_e^2c^2}$$

Damit ist Gleichung (3) hergeleitet, mit $p_\perp$ der Transversalkomponente des Schwerpunktimpulses des Elektron-Positron-Paares. Selbes Ergebnis erhält man direkt mit der Dopplerverschiebung (Taylor-Entwicklung).

## B. FIT-PARAMETER

**Tabelle 1:** Fit-Parameter für den Fit in Abbildung 3.

| Funktion $y(x) =$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $c(-(x-a)^2 + b^2)$ | $2.15 \cdot 10^{-7}$ | 5.30 | $1.32 \cdot 10^{-2}$ |
| $\frac{a}{\sqrt{2\pi}c} \cdot \exp\left(-\frac{(x-b)^2}{2c^2}\right)$ | 7.56 | −0.14 | 5.81 |

## C. PYTHON-CODE

**Listing 1:** Python-Code für die Bewegung des Detektors, Datenaufnahme und -speicherung.

```
1  from pylab import *
2  import time
3  import numpy as np
4  import steprocker
5  import counter
6
7  #Anzahl von schritten bis zur Mitte
8  mitSchritt=5884486
9  #Anzahl der Messpunkte
10 anzahl=60
11 #Messzeit pro Messpunkt (auf sinnvollen Wert stellen) 1 week: t=10080s
12 messzeit=10080
13 #Umrechnungsfaktor von Schritten in Weg in mm
14 mmps=1.56e-5
15 #Schrittweite
16 schrittw=round(0.625/mmps)
17
18 #Steuerung für den Schrittmotor
19 sp=steprocker.steprocker()
20 #Steuerung für den Counter
21 co=counter.counter()
22
23 #Erstmal bis zum Endschalter fahren
24 sp.schnell()
25 sp.rotLinks()
26 while sum(sp.statusLimitSwitch())==0: time.sleep(1)
27 sp.stop()
28 sp.langsam()
29
30 #Und dann zum ersten Messpunkt fahren
31 sp.fahreSchritte(mitSchritt-schrittw*anzahl/2)
```

```
#Warten bis die Position erreicht ist
while sp.posEreicht()==0: time.sleep(1)

fp = open('result.txt','w')
fp.write("koinzident single\n")
fp.close()
#Eigentliche Messung

print("start for schleife ")
for idx in range(anzahl):
    print("step "+str(idx))
    #Counter auf null setzen
    co.clear()
    #Counter starten
    co.start()
    #gewisse Zeit warten
    time.sleep(messzeit)
    #Counter stoppen
    co.stop()
    #Counter auslesen
    result = co.counts()
    #Ergebnis speichern
    fp = open('result.txt','a')
    fp.write(str(result[0])+" "+str(result[1])+"\n")
    fp.close()
    #Schrittmotor verfahren
    sp.fahreSchritte(schrittw)
    #Warten bis die Position erreicht ist
    while sp.posEreicht()==0: time.sleep(1)
```

**Listing 2:** Python-Code für die Auswertung der Daten, Fits in Abb. 3 und Berechnungen.

```
import numpy
import math
import matplotlib.pyplot
import scipy.optimize
import scipy.constants
import scipy.stats
import scipy.integrate
import pandas
matplotlib.rcParams['text.usetex'] = True
matplotlib.pyplot.rc('font', family='serif')
matplotlib.rc('figure', figsize=(20, 10))


def gaussian(x, d, mu, sigma):
    return d/(numpy.sqrt(2*numpy.pi)*sigma) * numpy.exp(-numpy.power((x-mu)/
        sigma, 2)/2.0)


def parabolic(x, a, b, c):
    return c*(-(x-a)**2 + b**2)


def f(x, a, b, c, d, mu, sigma):
    xmin = min(a+b, a-b)
```

```
24      xmax = max(a+b, a-b)
25      return parabolic(x, a, b, c)*(1-numpy.heaviside(x-xmax, 1))*numpy.heaviside
            (x-xmin, 1) + gaussian(x, d, mu, sigma)
26
27
28  print("********** START **********")
29  print("1. Theoretical calculations")
30
31  neTheo = 8.5*10**(28)
32  pfermiTheo = scipy.constants.hbar*pow(3*(math.pi**2)*neTheo, 1/3)
33  EfermiTheo = scipy.constants.hbar**2/(2*scipy.constants.m_e*scipy.constants.e)*
        pow(3*(math.pi**2)*neTheo, 2/3)
34
35  print("2. Data analysis")
36
37  inputString = r'C:/Users/micha/TUM/_Ex-FoPra/SS-19/86-Messung-des-Fermiimpulses
        /Daten/Gruppe48-original/result.xlsx'
38  outputString = r'C:/Users/micha/TUM/_Ex-FoPra/SS-19/86-Messung-des-
        Fermiimpulses/Auswertung/result-1.png'
39  yNcoincidenceRaw = numpy.array(pandas.read_excel(inputString, usecols='A'),
        dtype=numpy.float64).ravel()
40  yNsingle = numpy.array(pandas.read_excel(inputString, usecols='B'), dtype=numpy
        .float64).ravel()
41
42  print("N_coincidence raw data:", yNcoincidenceRaw)
43  yNcoincidence = numpy.copy(yNcoincidenceRaw)
44  yNsingleMean = numpy.mean(yNsingle)
45  for i in range(len(yNcoincidence)):
46      # Correction
47      yNcoincidence[i] = yNcoincidence[i]*yNsingleMean/yNsingle[i]
48  yNcoincidenceMax = max(yNcoincidence)
49  y = numpy.copy(yNcoincidence)
50  for i in range(len(y)):
51      # Rescaling to max = 1
52      y[i] = y[i]/yNcoincidenceMax
53  print("N_coincidence after the correction with the probability of detection:",
        yNcoincidence)
54  print("N_single mean: %.1f" % (yNsingleMean))
55
56  dphi = round(0.625/(1.56*10**(-5)))*1.56*10**(-5)/1.25
57  print("angle step size: %.6f" % (dphi))
58  xRaw = numpy.fromiter((dphi*i for i in range(0, len(y), 1)), numpy.float64).
        ravel()
59  x = numpy.copy(xRaw)
60
61  initial = [[0] * 3 for i in range(2)]
62  """ estimatedValues for the first iteration
63      0 = theoretical value of the fermi-momentum
64      1 = estimation of the maximum in the parabolic function
65      2 = estimation of the deviation of the zero point (parabolic) with the
            maximum x value of N_coincidence
66      3 = estimation of the maximum in the gaussian function
67      4 = estimation of the deviation of the zero point (gaussian) = 2
68      5 = estimation of sigma in the gaussian function
69      parabolic: in_a = (2), in_b = (0)-(2), in_c = (1)/((0)-(2))^2
70      gaussian: in_d = (3)/(sqrt(2pi)*(5)), in_mu = (4), in_sigma = (5)
```

```
"""
temp = x[numpy.argmax(y)]
estimatedValues = [5.5, 0.44, temp, 0.55, temp, 5]

""" initial
    [0][0,1,2] := parabolic: a, b, c
    [1][0,1,2] := gaussian: d, mu, sigma
"""
initial[0][0] = estimatedValues[2]
initial[0][1] = estimatedValues[0]-estimatedValues[2]
initial[0][2] = estimatedValues[1]/(initial[0][1]**2)
initial[1][0] = estimatedValues[3]*(numpy.sqrt(2*numpy.pi)*estimatedValues[5])
initial[1][1] = estimatedValues[4]
initial[1][2] = estimatedValues[5]
tempPfermi = [-estimatedValues[0], estimatedValues[0]]

print("Start of the iteration:")
k = 1
xmin = 0
xmax = 0
poptg = [0, 0, 0]
poptp = [0, 0, 0]
while(k <= 100):
    print("step:", k)
    k += 1
    # 1. Centering the data
    x = x - initial[0][0]
    # 2. Split data (gaussian and parabolic part)
    temp = numpy.argwhere((tempPfermi[0] < x) & (x < tempPfermi[1])).ravel()
    tempParabolic = [numpy.take(x, temp), numpy.take(y, temp)]
    temp = numpy.argwhere((tempPfermi[0] >= x) | (x >= tempPfermi[1])).ravel()
    tempGaussian = [numpy.take(x, temp), numpy.take(y, temp)]

    # 3. Gaussian-Fit
    poptg, pcov = scipy.optimize.curve_fit(gaussian, tempGaussian[0],
        tempGaussian[1], p0=tuple(initial[1]))
    # 4. Calculation of the y data for the parabolic-fit
    for i in range(len(tempParabolic[1])):
        tempParabolic[1][i] -= gaussian(tempParabolic[0][i], poptg[0], poptg
            [1], poptg[2])
    initial[1] = list(poptg)

    # 5. Parabolic-Fit
    poptp, pcov = scipy.optimize.curve_fit(parabolic, tempParabolic[0],
        tempParabolic[1], p0=tuple(initial[0]))
    initial[0] = list(poptp)

    xmin = min(poptp[0]+poptp[1], poptp[0]-poptp[1])
    xmax = max(poptp[0]+poptp[1], poptp[0]-poptp[1])
    print("xmin: %.15f, xmax: %.15f" % (xmin, xmax))
    print("new initial values for gaussian-fit:", poptg, ", parabolic-fit:",
        poptp)
    tempPfermi[0] = xmin
    tempPfermi[1] = xmax
# 6. Centering the data
x = x - initial[0][0]
```

```
123
124 # 7. Calculation of the fermi-momentum and -energy
125 pfermi = 10**(-3)*(abs(xmax)+abs(xmin))/2*scipy.constants.m_e*scipy.constants.c
126 Efermi = 0.5/(scipy.constants.m_e*scipy.constants.e)*pfermi**2
127 print("Fermi-momentum: %.3e (theory: %.3e)\nFermi-energy: %.3f eV (theory: %.3f
        eV)" % (pfermi, pfermiTheo, Efermi, EfermiTheo))
128
129 n = numpy.arange(round(x[0]), numpy.ceil(x[-1]), 0.01)
130 nn = numpy.arange(xmin, xmax, 0.01)
131 # matplotlib.pyplot.plot(n, gaussian(n, poptg[0], poptg[1], poptg[2]), label='
        gauss '+str(k))
132 # matplotlib.pyplot.plot(nn, parabolic(nn, poptp[0], poptp[1], poptp[2]), label
        ='fit '+str(k))
133 matplotlib.pyplot.plot(n, f(n, poptp[0], poptp[1], poptp[2], poptg[0], poptg
        [1], poptg[2]), label='fit')
134 matplotlib.pyplot.scatter(x, y, label='symmetrized data')
135 matplotlib.pyplot.xlabel(r'$ p_\perp $ [mrad$\cdot m_\mathrm{e} c$]', fontsize
        =20)
136 matplotlib.pyplot.ylabel(r'$ \frac{N_\mathrm{c}}{N_{\mathrm{c,max}}} $ [a.u.]',
         fontsize=20)
137 matplotlib.pyplot.tick_params(direction='in', right=True, top=True, labelsize
        =20)
138 matplotlib.pyplot.ylim((0, 1.1))
139 matplotlib.pyplot.xlim((round(x[0])-1, abs(round(x[0]))+1))
140 matplotlib.pyplot.legend()
141 matplotlib.pyplot.grid(True)
142 # matplotlib.pyplot.savefig(outputString, bbox_inches='tight', dpi=1000)
143 matplotlib.pyplot.show()
144
145 print("3. Positron annihilation rate")
146
147 attenuationCoeff = 0.906  # cm^-1
148 thickness = 3  # cm
149 detectorAngle = 0.001*0.82  # approximation (length*height) in rad
150 Nmeasured = scipy.integrate.quad(f, -math.pi/2*10**(3), math.pi/2*10**(3), args
        =(poptp[0], poptp[1], poptp[2], poptg[0], poptg[1], poptg[2]))[0]*
        yNcoincidenceMax
151 rate = Nmeasured*pow(detectorAngle/(2*math.pi)*(1-math.exp(-attenuationCoeff*
        thickness))**2, -1)
152 print("Nmeasured (per second): %.3f, positron annihilation rate (per second):
        %.3f" % (Nmeasured, rate))
153
154 print("4. Data output")
155
156 outputString = r'C:/Users/micha/LaTeX/TUM/FoPra/86/files/result-raw.txt'
157 numpy.savetxt(outputString, numpy.transpose([xRaw, yNcoincidenceRaw]),
        delimiter=", ", fmt='%.6f')
158 outputString = r'C:/Users/micha/LaTeX/TUM/FoPra/86/files/result-N.txt'
159 numpy.savetxt(outputString, numpy.transpose([x, y]), delimiter=", ", fmt='%.6f'
        )
160 outputString = r'C:/Users/micha/LaTeX/TUM/FoPra/86/files/result-fit.txt'
161 numpy.savetxt(outputString, numpy.transpose([n, f(n, poptp[0], poptp[1], poptp
        [2], poptg[0], poptg[1], poptg[2])]), delimiter=", ", fmt='%.6f')
162
163 print("********** END **********")
```