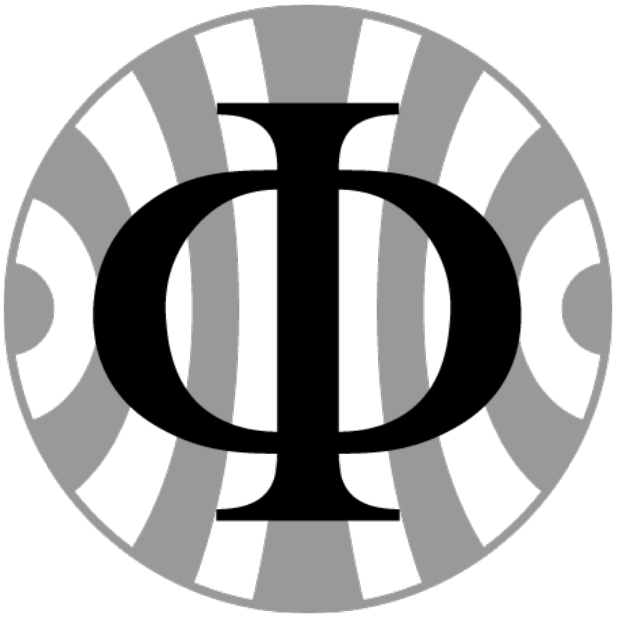
Versuch 255

PAP 2.2, [1]

06.03.2025



[2]

Teilnehmender Student: **Jonathan Rodemers**

Gruppe des Teilnehmenden: 1

Inhaltsverzeichnis

1. Einleitung

1.1 Motivation

Das Ziel dieses Experiments ist die Untersuchung der Röntgenspektren einer Molybdän-Anode mittels Bragg-Reflexion. Durch die Messung der Reflexionswinkel an Kristallen (LiF und NaCl) können die charakteristischen K_α- und K_β-Linien bestimmt werden. Zudem wird die Planck-Konstante h über die Analyse des kurzwelligen Endes der Bremsstrahlung abgeschätzt. Schließlich ermöglich die Bestimmung der Gitterkonstante des NaCl-Kristalls die Berechnung der Avogadro-Zahl N_A.

1.2 Messverfahren

Das Experiment nutzt eine Röntgenröhre mit einer Molybdän-Anode, ein Goniometer zur präzisen Winkelmessung sowie ein Zählrohr zur Detektion der Röntgenstrahlung. Der zentrale Messvorgang beruht auf der Bragg-Reflexion:

Der Kristall wird in definierten Winkeln bestrahlt, und die Intensität der reflektierten Strahlung wird als Funktion des Winkels aufgenommen. Über das Bragg'sche Gesetz können daraus die Wellenlängen der Strahlung bestimmt werden.

1.3 Grundlagen aus der Physik

Die Röntgenstrahlung entsteht in einer Röntgenröhre, in der Elektronen mit einer Energie von $E = eU$ auf die Anode treffen. Dadurch entsteht ein kontinuierliches Bremsstrahlungsspektrum mit einer minimalen Wellenlänge

$$\lambda_{gr} = \frac{hc}{eU} \tag{1.1}$$

und überlagerter charakteristischer Strahlung. Die Energien dieser Linien lassen sich mit dem Moseley'schen Gesetz:

$$E_{n\to m} = hcR_\infty (Z - A)^2 \left(\frac{1}{m^2} - \frac{1}{n^2}\right) \tag{1.2}$$

abschätzen. Die Röntgenbeugung an Kristallen folgt dem Bragg'schen Gesetz:

$$2d \sin\theta = n\lambda \tag{1.3}$$

und erlaubt die Bestimmung der Wellenlängen und der Kristallstruktur. Die Avogadro-Zahl ergibt sich schließlich aus

$$N_A = \frac{1}{2}\frac{M_{Mol}}{\rho d^3} \tag{1.4}$$

Diese Zusammenhänge bilden die theoretische Grundlage für die Durchführung und Auswertung des Experiments.

2. Durchführung

2.1 Messprotokol

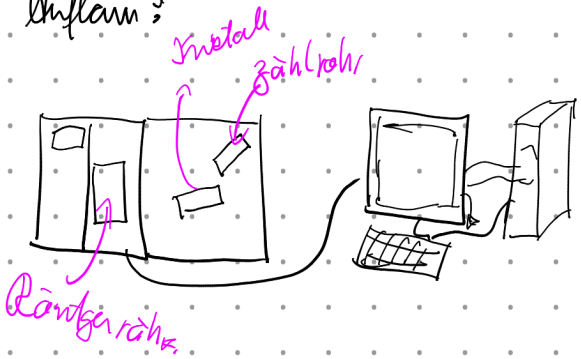
06.04.25 Versuch 255 Jonathan Rodemers
Marvel Sargenfrei

Geräte:
-> Röntgengerät mit Röntgenröhre (Molybdän-Anode)
-> Goniometer
-> Zählrohr
-> LiF- und NaCl-Kristalle
-> Computer
-> Kristallhalter mit CCD-Sensor

Aufg. 1-2
↳ txt. dateien

Aufg 3
Tabelle siehe Python.

Aufbau:



3. Auswertung

3.1 Bestimmung der Grenzfrequenz

Um die Grenzfrequnez bestimmen zu können müssen wir zunächst aus den gemessenen Spektrum den Grenzwinkel exptrapolieren.

Dazu fitten wir eine Lineare Funktion an den ersten Anstieg im Diagramm und Etrapolieren zur Null.

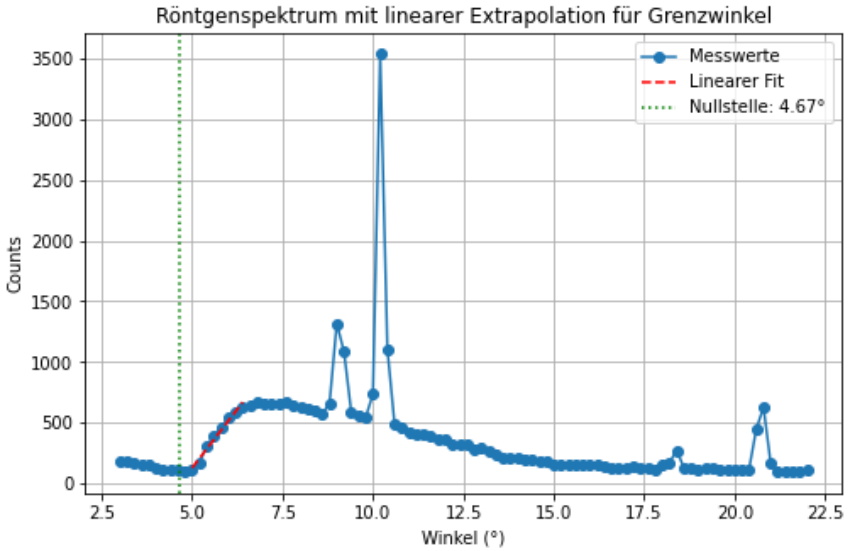


Abbildung 3.1: Counts Vs. Winkel

Dabei ergibt sich für die lineare Steigung:

$$m = (387 \pm 25)\frac{\text{Counts}}{\text{grad}} \quad a = (-1800 \pm 150)\text{counts}$$

Wobei m die Steigung des Fits ist und a der Y-Achsenabschnitt.

Wir berechnen also den Grenzwinkel mit:

$$\beta_g = -\frac{a}{m} \tag{3.1}$$

mit dem Fehler:

$$\Delta\beta_g = \sqrt{\left(\frac{-1}{m}\Delta a\right)^2 + \left(\frac{a}{m^2}\Delta m\right)^2} \tag{3.2}$$

daraus ergibt sich:

$$\beta_g = (4,7 \pm 0,5)^\circ$$

Die Grenzwellenlänge erhält man nun indem man die Formel 1.3 verwendet und n=1 setzt und den Fehler errechnet mit:

$$\Delta\lambda_g = 2d\cos(\beta_g)\Delta\beta_g \tag{3.3}$$

Dabei ist d der Netzebeneabstand von LiF mit $d = 201,4$ pm [2].

Daraus ergibt sich nun die Grenzfrequenz:

$$\lambda_g = (30 \pm 3)\ \text{pm}$$

Nun kann man mit der Formel 1.1 auf das Planksche Workungsqunatum schließen.

$$h = \frac{\lambda_g e U}{c} \tag{3.4}$$

Mit dem Fehler:

$$\Delta h = \frac{Ue}{c} \Delta\lambda_g \tag{3.5}$$

so ergibt sich für h:

$$h = (5,6 \pm 0,6) \cdot 10^{-34}\ \text{Js}$$

Verglichen mit dem Literaturwert aus dem Skript entspricht das einer Abweichung von 1,7 σ. Also noch in einem nicht signifikanten Bereich.

Nun können wir erneuet Formel 1.3 benutzen um für n = 2, den Winkel zu bestimmen, ab dem das Speektrum zweiter Ordnung einsetzen sollte.

$$\beta_2 = \arcsin\left(\frac{2\lambda_g}{2d}\right) \tag{3.6}$$

Mit dem Fehler:

$$\Delta\beta_2 = \frac{1}{d\sqrt{1 - \frac{\lambda_g^2}{d^2}}} \tag{3.7}$$

Daraus ergibt sich:

$$\beta_2 = (8,584 \pm 0,015)^\circ$$

3.2 Bestimmung der Peaks und FWHM

Im weiteren wurden die zuvor schon grop vermessenen Peaks genauer Untersucht. Dabei wurden die Peaks gefunden und anschließend in die jeweilige Frequenz übersetzt außerdem wurde die das FHWM für die K_α Linie der Ersten Ordnung bestimmt. Dabei ergeben sich folgende Bilder:

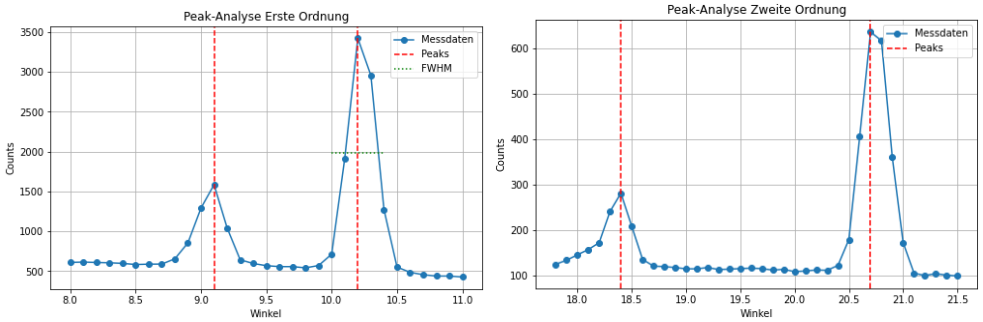


Abbildung 3.2: Peaks der K-Linien

Es wurde analog wie davor aus dem Winkel die jeweiligen Frequnezen bestimmt, es ergibt sich insgesammt also folgende Tabelle:

Linie	λ [pm]	FWHM [pm]
K_{β_1}	$57 \pm 0,6$	-
K_{α_1}	$64,1 \pm 0,6$	$1,3 \pm 0,8$
K_{β_2}	$114,6 \pm 0,6$	-
K_{α_2}	$128,4 \pm 0,6$	-

Tabelle 3.1: Position der K-Linien

Dabei wurden die Fehler für die Position der Peaks sinnvoll mit der maximalen Auflösung (Stepsize) abgeschätzt.

3.3 Erneute Berechnug der Plankkosntate

Diesmal lassen wir den Winkel fest bei 7.5° und variieren die Spannung. Wir tragen die Counts gegen die Spannun auf uns extrapolierne wieder den Wert der Spannung bei dem theoretisch keine Counts gemessene werden würden. Damit haben wir U_{grenz} gefunden, woraus wir wiederrum h berechnen können.

Für den Plot ergibt sich: Den Fehler des Fits erhalten wir analog zu dem Fehler für β_g. Es ergibt

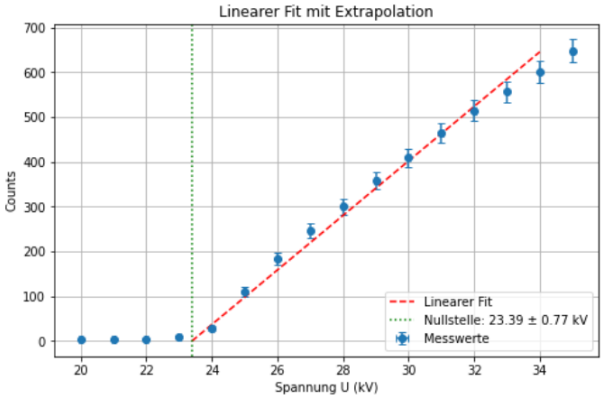


Abbildung 3.3: Counts vs. Spannugn bei festem Winkel

sich also:

$$U_{grenz} = (23,39 \pm 0,77)\ \text{kV}$$

Wir können nun also Gleichung 1.3 in die Gleichung 1.1 einsetzen und erhält eine weitere Gleichung für h:

$$h = \frac{eU_{grenz}}{c} \cdot 2d\sin(7,5°) \tag{3.8}$$

Mit dem Fehler:

$$\Delta h = \frac{2ed\sin(\beta)}{c}\Delta U_{grenz} \tag{3.9}$$

Damit ergibt sich ein h von:

$$h = (5,91 \pm 0,19) \cdot 10^{-34}\ \text{Js}$$

Welches eine Abweichung von 3,7 σ zum Literaturwert aufweißt. Dies ist vermutlich zurückzuführen auf den kleinen Fehler, der auch schon bei U_{grenz} sehr gering war, welches vllt darauf

zurückzuführen ist, welchen Fitbereich man wählt.

3.4 Analyse des NaCl- Kritalls

Im weiteren haben wir den NaCl Kristal untersucht und auch hier eine Peak analyse durchgeführt:

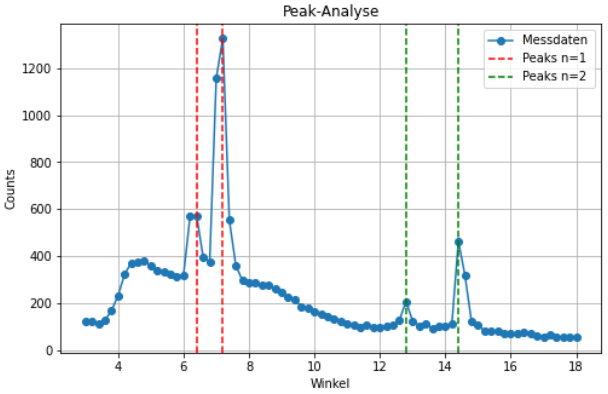


Abbildung 3.4: Peaks des NaCl Kristals

Daraus ergibt sich dann folgende Tabelle:

Linie	β [pm]	d_2 [pm]
K_{β_1}	$6,4 \pm 0,2$	285 ± 9
K_{α_1}	$7,2 \pm 0,2$	284 ± 8
K_{β_2}	$12,8 \pm 0,2$	287 ± 5
K_{α_2}	$14,4 \pm 0,2$	286 ± 4

Tabelle 3.2: Position der K-Linien vom Nacl

Dabei wurden analog zum LiF Kristall die Fehler auf die gleiche Weise abgeschätzt.

Aus der Formel 1.3 kann man eine Beziehung zwischen den Gitterkonstanten und den Winkel herstellen, für zwei verschiedene Kristalle, es gilt:

$$d_2 = d_1 \frac{\sin(\beta_1)}{\sin(\beta_2)} \tag{3.10}$$

mit dem Fehler:

$$\Delta d_2 = \sqrt{\left(\frac{d_1 cos\left(\beta_1\right)}{sin\left(\beta_2\right)} \Delta\beta_1\right)^2 + \left(\frac{-d_1 sin\left(\beta_1\right) cos\left(\beta_2\right)}{sin\left(\beta_2\right)^2} \Delta\beta_2\right)^2} \tag{3.11}$$

So wurden die Werte für die Gitterkonstanten in 3.2 errechnet.

Die gemittlete Gitterkonstate d_2 ist somit:

$$d_2 = (286 \pm 7)\ \text{pm}$$

Nun kann man mit dem Molekulargewicht und Dichte von NaCl und der Formel 1.4 noch die Avogadrozahle bestimmen.

Man benutzt hierfür den Fehler von:

$$\Delta N_A = \frac{-3M_{Mol}}{2d_2^4\rho}\Delta d_2 \tag{3.12}$$

Man erhält so eine Avogadrozahl von:

$$N_A = (5,8 \pm 0,4) \cdot 10^{23}\ \text{mol}^{-1}$$

Welches lediglich um 0.6σ vom Literaturwert [2] abweicht.

4. Zusamenfassung und Diskussion

Im Rahmen dieses Experiments wurde das Röntgenspektrum einer Molybdän-Anode mittels Bragg-Reflexion untersucht. Durch die Messung der Reflexionswinkel an LiF- und NaCl-Kristallen konnten die charakteristischen K_α- und K_β-Linien detektiert und ihre Wellenlängen bestimmt werden. Zudem wurde durch die Extrapolation des kurzwelligen Endes der Bremsstrahlung die Grenzfrequenz berechnet und daraus die Planck-Konstante h bestimmt. Eine weietre Bestimmung von h wurde durch die Messung der Intensität der Röntgenstrahlung in Abhängigkeit von der Beschleunigungsspannung durchgeführt, indem die Einsatzspannung extrapoliert wurde. Zusätzlich wurde die Gitterkonstante des NaCl-Kristalls ermittelt und daraus die Avogadro-Zahl N_A berechnet. Die Ergebnisse zeigen eine hohe Übereinstimmung mit Literaturwerten, wobei die berechnete Planck-Konstante im ersten Ansatz eine Abweichung von 1,7 σ und im zweiten Ansatz von 3,7 σ aufweist. Die Bestimmung der Gitterkonstante des NaCl-Kristalls führte zu einem Mittelwert von $d_2 = (286 \pm 7)$ pm, wodurch sich eine Avogadro-Zahl von $N_A = (5,8 \pm 0,4) \cdot 10^{23}$ mol^{-1}ergab, was lediglich eine Abweichung von 0,6 σ zum Literaturwert darstellt.

Mögliche Fehlerquellen ergeben sich insbesondere durch die begrenzte Auflösung des Goniometers, da die Schritte in der Winkelmessung eine systematische Ungenauigkeit in der Bestimmung der Reflexionswinkel verursachen können. Ebenso könnte die Wahl des Fitbereichs zur Extrapolation des kurzwelligen Endes der Bremsstrahlung einen Einfluss auf die Genauigkeit der Planck-Konstanten-Bestimmung haben. Kleine Variationen in der Auswahl der Datenpunkte können dabei systematische Abweichungen hervorrufen. Zudem könnten Materialunreinheiten oder Oberflächenbeschaffenheiten der Kristalle die Reflexionsbedingungen beeinflussen (vorallem weil das ein PAP Versuch ist lol) und zu kleinen Abweichungen in den berechneten Gitterkonstanten führen. Ein weiterer Unsicherheitsfaktor ist die Annahme einer idealen Bragg-Reflexion ohne zusätzliche Streu- oder Absorptionseffekte im Kristall.

Trotz dieser potenziellen Fehlerquellen war das Experiment insgesamt erfolgreich. Die wichtigsten physikalischen Größen wurden mit einer Genauigkeit bestimmt, die innerhalb akzeptabler Abweichungen vom Literaturwert liegt.

5. Anhang

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

```
U = np.array([20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35 ])  # Kv
Delta_U = 0.05 #kv maybe
# Fehler von Counts ist einfach Wurzel aus den Cunts
Counts = np.array([3.05,4.35,5.0,10.2,27.45,110.3,184.9,246.2,300.4,357.9,409.7,4
```

```
def linear_model(x, a, b):
    return a * x + b

def process_xray_data(filename, fit_min, fit_max):
    nts)
    with open(filename, 'r') as file:
        lines = file.readlines()

    # Ersetzen von Kommas durch Punkte und Konvertierung in numpy-Array
    cleaned_lines = [line.replace(',', '.') for line in lines]
    data = np.loadtxt(cleaned_lines)
    angles, counts = data[:, 0], data[:, 1]

    # Plot der Daten
    plt.figure(figsize=(8, 5))
    plt.plot(angles, counts, marker='o', linestyle='-', label='Messwerte')

    # Auswahl des Bereichs f r die lineare Regression
    fit_mask = (angles >= fit_min) & (angles <= fit_max)
    fit_angles, fit_counts = angles[fit_mask], counts[fit_mask]

    # Linearen Fit durchf hren
    popt, pcov = curve_fit(linear_model, fit_angles, fit_counts)
    a, b = popt
    print("Steigung m und abschnitt a=", popt)
    print("Fehler von m, a =",np.sqrt(np.diag(pcov)))
    #fehler = np.sqrt(pcov[0,0])
```

```
    # Nullstellenberechnung (Extrapolation bis Counts = 0)
    angle_zero = -b / a
    print(f"Extrapolierter Winkel f r Null Counts: {angle_zero:.2f}  ")

    # Plot des Fits
    fit_x = np.linspace(min(fit_angles), max(fit_angles), 100)
    fit_y = linear_model(fit_x, a, b)
    plt.plot(fit_x, fit_y, 'r--', label='Linearer Fit')

    # Markierung des extrapolierten Nullpunkts
    plt.axvline(angle_zero, color='g', linestyle=':', label=f'Nullstelle: {angle_

    # Plot formatieren
    plt.xlabel('Winkel (  )')
    plt.ylabel('Counts')
    plt.legend()
    plt.grid()
    plt.title('R ntgenspektrum mit linearer Extrapolation f r Grenzwinkel')
    plt.show()
    return angle_zero



filename = "data2/Sherm_aufgabe_1.txt"



process_xray_data(filename, 5, 6.5)
```

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import find_peaks, peak_widths

def process_peak_data(filename):
    # Datei einlesen und Kommas durch Punkte ersetzen
    with open(filename, 'r') as file:
        lines = [line.replace(',', '.') for line in file]

    # Daten laden
    data = np.loadtxt(lines)
    x, y = data[:, 0], data[:, 1]
```

```
    # Plot der Daten
    plt.figure(figsize=(8, 5))
    plt.plot(x, y, label='Messdaten', linestyle='-', marker='o')

    # Peaks finden
    peaks, _ = find_peaks(y, height=np.max(y) * 0.25)

    # Halbwertsbreiten berechnen
    widths, width_heights, left_ips, right_ips = peak_widths(y, peaks, rel_heigh

    # Peaks und FWHM anzeigen
    for i, peak in enumerate(peaks):
        plt.axvline(x[peak], color='r', linestyle='--', label=f'Peaks ' if i ==

        print(f"Peak {i+1}: x = {x[peak]:.2f}, y = {y[peak]:.2f}, FWHM = {x[int(
    plt.hlines(width_heights[1], x[int(left_ips[1])-1], x[int(right_ips[1])+1],
    # Plot anpassen
    plt.xlabel('Winkel')
    plt.ylabel('Counts')
    plt.legend()
    plt.title('Peak-Analyse Erste Ordnung')
    plt.grid()
    plt.show()


process_peak_data('data2/sherm2_1.txt')
```

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

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

def process_voltage_data(U, Counts, Delta_U, fit_min, fit_max):
    # Fehler der Counts als Wurzel der Counts
    Errors_Counts = np.sqrt(Counts)

    # Plot der Daten
    plt.figure(figsize=(8, 5))
```

```
plt.errorbar(U, Counts, yerr=Errors_Counts, xerr=Delta_U, fmt='o', label='Me

# Auswahl des Bereichs f r die lineare Regression
fit_mask = (U >= fit_min) & (U <= fit_max)
fit_U, fit_Counts = U[fit_mask], Counts[fit_mask]
fit_Errors = Errors_Counts[fit_mask]

# Linearen Fit mit Fehlerber cksichtigun
popt, pcov = curve_fit(linear_model, fit_U, fit_Counts, sigma=fit_Errors, ab
a, b = popt
perr = np.sqrt(np.diag(pcov))  # Fehler der Fit-Parameter

# Nullstellenberechnung (Extrapolation bis Counts = 0)
U_zero = -b / a

# Fehlerfortpflanzung f r Nullstelle
U_zero_error = U_zero * np.sqrt((perr[0] / a) ** 2 + (perr[1] / b) ** 2)

print(f"Extrapolierte Nullstelle: {U_zero:.2f}    {U_zero_error:.2f} kV")
print(f"Steigung: a = {a:.4f}    {perr[0]:.4f}")
print(f"Achsenabschnitt: b = {b:.4f}    {perr[1]:.4f}")

# Fit plotten
fit_x = np.linspace(U_zero, max(fit_U), 100)
fit_y = linear_model(fit_x, a, b)
plt.plot(fit_x, fit_y, 'r--', label='Linearer Fit')

# Markierung des extrapolierten Nullpunkts
plt.axvline(U_zero, color='g', linestyle=':', label=f'Nullstelle: {U_zero:.2

# Plot formatieren
plt.xlabel('Spannung U (kV)')
plt.ylabel('Counts')
plt.legend()
plt.grid()
plt.title('Linearer Fit mit Extrapolation')
plt.show()

return U_zero, U_zero_error
```

```
process_voltage_data(U, Counts, Delta_U, fit_min=24, fit_max=34)
```

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import find_peaks, peak_widths

def process_peak_data(filename):
    # Datei einlesen und Kommas durch Punkte ersetzen
    with open(filename, 'r') as file:
        lines = [line.replace(',', '.') for line in file]

    # Daten laden
    data = np.loadtxt(lines)
    x, y = data[:, 0], data[:, 1]

    # Plot der Daten
    plt.figure(figsize=(8, 5))
    plt.plot(x, y, label='Messdaten', linestyle='-', marker='o')
    y1 = y[0:27]
    y2 = y[46:70]
    # Peaks finden
    peaks1, _ = find_peaks(y1, height=np.max(y1) * 0.3)
    peaks2, _ = find_peaks(y2, height=np.max(y2) * 0.3)  # Peaks mit min. 50% de
    print(peaks2)
    # Halbwertsbreiten berechnen
    #widths, width_heights, left_ips, right_ips = peak_widths(y, peak1, rel_heigl
    x2 = x[46:70]
    # Peaks und FWHM anzeigen
    for i, peak in enumerate(peaks1):
        plt.axvline(x[peak], color='r', linestyle='--', label=f'Peaks n=1 ' if i
        print(f"Peak {i+1}: x = {x[peak]:.2f}")

    for i, peak in enumerate(peaks2):
        plt.axvline(x2[peak], color='g', linestyle='--', label=f'Peaks n=2 ' if
        print(f"Peak {i+1}: x = {x2[peak]:.2f}")
    #plt.hlines(width_heights[1], x[int(left_ips[1])-1], x[int(right_ips[1])+1],
    # Plot anpassen
    plt.xlabel('Winkel')
    plt.ylabel('Counts')
    plt.legend()
    plt.title('Peak-Analyse')
```

```
    plt.grid()
    plt.show()


process_peak_data('data2/Aufgabe 4.txt')
```

Quellen- und Literaturverzeichnis

[1] Captain Joni: *pap1-tex-vorlage*. https://github.com/captain-joni/pap1-tex-vorlage. – [Online; Stand 28.08.2024]

[2] Dr. J.Wagner: *Physikalisches Praktikum 1 f¨ur Studierende der Physik B.Sc.* https://www.physi.uni-heidelberg.de/Einrichtungen/AP/info/Corona/PAP1.pdf. – [Online; Stand 01/2014]