

Physikalisches Anfängerpraktikum II

Sommersemester 2023

Versuch 234

Tutor: Henri Valentin Lurtz

Lichtquellen und Gitterspektroskopie

1 Einleiung¹

1.1 Ziel des Versuchs

Das Ziel dieses Experiments ist es, die Sonnenlichtspektren sowohl mit als auch ohne eine Fensterscheibe aufzuzeichnen, um das gestreute Himmelslicht zu untersuchen sowie durch den Vergleich des Sonnenlichts mit und ohne Glasscheibe die Absorption von Fensterglas zu bestimmen. Außerdem soll die markanten Fraunhoferlinien im Spektrum identifiziert werden.

Es wurden die Spektren mehrerer Lichtquellen wie Sonnenlicht, Glühlampe, LEDs und Energiesparlampen erfasst und miteinander verglichen. Zusätzlich erfolgte eine spektroskopische Untersuchung durch die detaillierte Aufzeichnung des Spektrums einer Natriumlampe mit hoher Auflösung, dabei sollen die Serienübergänge der Hauptserie sowie zweier Nebenserien zugeordnet werden. Des Weiteren wurden die korrigierten Energieniveaus im Natrium ermittelt, welche von der Drehimpulsabhängigkeit beeinflusst werden.



Abbildung 1: Versuchsaufbau

¹Dr. J.Wagner - Physikalisches Anfängerpraktikum - V. 2.0β Stand 04/2023 - Python Edition

1.2 Spektroskopie²

Spektroskopie ist die Analyse von Licht oder elektromagnetischer Strahlung, die von Materie absorbiert oder emittiert wird. Es umfasst verschiedene Methoden wie Absorption, Emission, Massen- und NMR-Spektroskopie. Diese Techniken werden genutzt, um Materialzusammensetzungen zu bestimmen und Molekülstrukturen zu untersuchen, in verschiedenen wissenschaftlichen und technologischen Bereichen.

1.3 Temperaturstrahler

Ein Körper mit Temperatur größer als 0K sendet elektromagnetische Strahlung in Form von Wärmestrahlung aus, deren Intensität abhängig von der Temperatur ist. Ein typisches Beispiel dafür wird ein idealisiertes Schwarzkörper angenommen, der die gesamte einfallende elektromagnetische Strahlung vollständig absorbiert und die dann mit dem maximalen Emissionskoeffizienten $\epsilon = 1$ abstrahlt. Die Intensitätsverteilung der abgestrahlten Wärmestrahlung eines Schwarzkörpers wird durch das **Plancksche Strahlungsgesetz** beschrieben:

$$M_\lambda(\lambda, T)dAd\lambda = \frac{2\pi hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda kT}} - 1} dAd\lambda \quad (1)$$

Hierbei steht $E_\lambda(\lambda, T)$ für die spektrale Strahlungsdichte in Abhängigkeit von der Wellenlänge λ und der Temperatur T des Schwarzkörpers. h ist das Plancksche Wirkungsquantum, c ist die Lichtgeschwindigkeit, und k die Boltzmann-Konstante. Die spektrale Intensitätsverteilung bei reinen Temperaturstrahlern ist nur proportional zu der Temperatur; die temperaturabhängige Verschiebung des Intensitätsmaximums wird quantitativ durch das **Wiensche Verschiebungsgesetz** beschrieben:

$$\lambda_{max} = \frac{2897,8\mu m \cdot K}{T} \quad (2)$$

Mit steigender Temperatur emittiert ein Körper Licht mit höheren Frequenzanteilen und sein entsprechendes Spektrum zeigt immer verschiedene Farbenmischung von Lichtanteilen, was als Farbtemperatur bekannt ist. Bei 0 Kelvin ist der Körper absolut schwarz, während bei etwa 5500 Kelvin alle sichtbaren Wellenlängen mit ähnlicher Intensität emittiert werden, wodurch das Licht weiß erscheint.

Nach Rayleigh-Streuung ist der Wirkungsquerschnitt proportional zur vierten Potenz der Frequenz des Lichts, weshalb das gestreute Sonnenlicht (Himmelslicht) bläulich kalt (höhere Frequenz) ist während das direkte Sonnenlicht "warm" (niedrigere Frequenz) erscheint.

²<https://de.wikipedia.org/wiki/Spektroskopie>

Fraunhoferlinien sind dunkle Linien im Sonnenspektrum, die durch Absorption in der Sonnen- und Erdatmosphäre von Licht bei bestimmten Wellenlängen entstehen. Fraunhoferlinien sind entscheidend, um die Elementzusammensetzung von Sonnen und anderen Sternen zu verstehen, da sie Informationen basierend auf den charakteristischen Absorptionsspektren über die vorhandenen chemischen Elemente liefern.

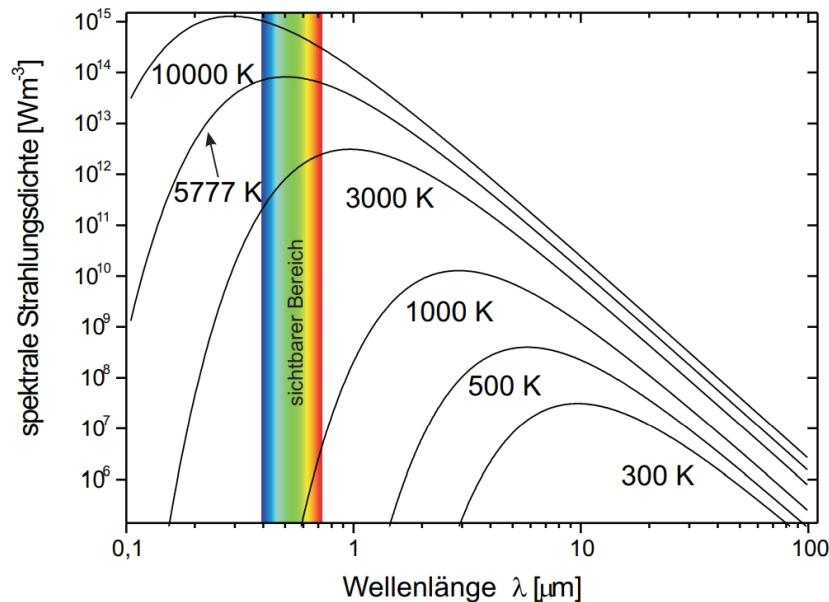


Abbildung 2: Spektrale Intensitätsverteilung eines schwarzen Körpers bei unterschiedlichen Temperaturen

1.4 Nichttemperaturstrahler

Die Lichterzeugung bei Nichttemperaturstrahlern basiert nicht auf Wärmestrahlung, sondern auf der Anregung spezifischer Atomzustände in Gasen oder Festkörpern oder der Rekombination von Elektron-Loch-Paaren in Halbleitern. Im Gegensatz zu Temperaturstrahlern weisen solche Lichtquellen ein diskretes Spektrum auf. Die spektrale Zusammensetzung des emittierten Lichts bei Gasentladungslampen hängt von dem verwendeten "Leuchtgas" (wie Metalldämpfe oder Gase) ab.

1.5 Das Natriumspektrum

Alkaliatome (IA) sind charakterisiert durch ein Leuchtelektron und zeigen dementsprechend "wasserstoffartige Spektren". Das Leuchtelektron im Natrium wird nicht direkt von der vollen Kernladung beeinflusst, sondern es erfährt ein kugelsymmetrisches Potential $V(r)$, das teilweise durch den Neonrumpf mit 10 Elektronen abgeschirmt wird. Dieses Potential zeigt in der Nähe des Kerns das Coulombpotential während für große Radien ein Wasserstoffpotential besitzt:

$$V_{\text{nah}}(r) = \frac{Ze^2}{4\pi\epsilon_0 r} \quad V_{\text{fern}}(r) = \frac{e^2}{4\pi\epsilon_0 r} \quad (3)$$

In der Entfernung hängen die Energieniveaus nicht mehr nur von der Hauptquantenzahl n ab sondern auch von der Bahndrehimpulsquantenzahl l ab:

$$E_{n,l} = -13.6eV \frac{1}{(n - \Delta_{n,l})^2} \quad (4)$$

Für die Korrektur gilt hierbei $\Delta_{l,n} \approx \Delta_l$. Nach dem chemischen **Aufbauprinzip**³ sind s-Elektronen näher am Kern als p-Elektronen, während d-Elektronen weiter entfernt sind. Im Grundzustand ist das Valenzelektron des Natriums ($1s^2 2s^2 2p^6 3s^1$) im 3s-Zustand. Die Hauptserie umfasst Übergänge $np \rightarrow 3s$ zurück zum Grundzustand, wobei die bekannteste Linie das gelbe Dublett bei 589 nm ist, während zum 3p-Zustand die Nebenserien $nd \rightarrow 3p$ (1. Nebenserie) und $ns \rightarrow 3p$ (2. Nebenserie) ebenfalls Linien im beobachtbaren Spektralbereich erzeugen.

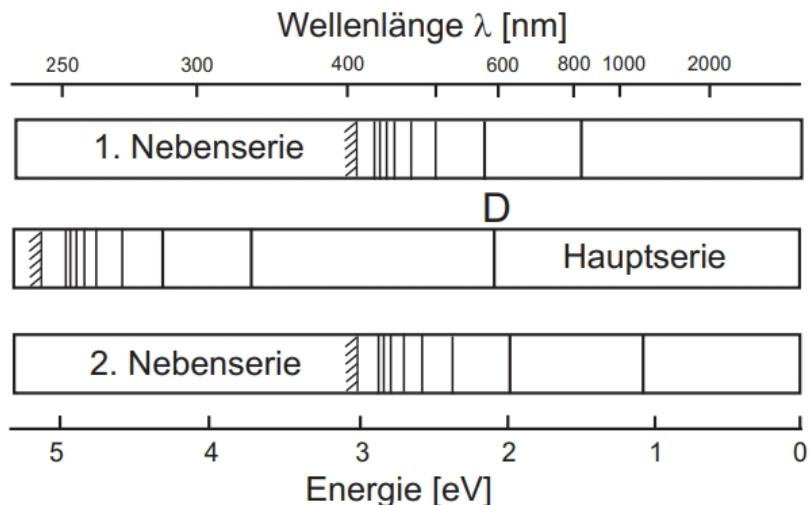


Abbildung 3: Spektren der Natriumserien

2 Versuchsdurchführung

2.1 Versuchsaufbau, Versuchsdurchführung und Messprotokoll

Siehe folgende Seiten.

³siehe Aufbauprinzip und Das Madelung-Energieschema

- Messgeräte:
- Gitterspektrometer, Ocean Optics L1SB 4000
 - verschiedene Lichtquellen: LED, Laser, Energiesparlampe, Halogenlampe, Glühlampe, Natriumdampflampe
 - PC mit Drucker

Vor der Aufnahme des Spektrums stellen wir die Integrationszeit und die Zahl der Scans zur Mittelung ein, wir werden die folgenden Messungen mit dem Programm "Spectra Suite" durchführen und im Scope-Modus.

1. Sonnenspektrum

Wir speichern nacheinander die auf den Dunkelstrom korrigierten Spektren von:

- Himmellicht bei geöffneten Fenster.
- Himmellicht hinter Fensterglas.

2. Qualitativer Vergleich einfacher Lichtquellspektren

Wir führen die Messungen von [Glühlampe, Halogenlampe, Energiesparlampe, LED's, Laser] nur qualitativ durch.

3. Natriumspektrum

Wir warten bis die Natriumlampe stabil brennt, dann richten das Objektiv auf die Lichtquelle aus. Wir wählen die Blendeöffnung s.d. die gelbe Linie bei einer Integrationszeit von 5ms gerade in Sättigung.

- Aufnahme der Linien kleiner Intensitäten

①



扫描全能王 创建

Wir versuchen möglichst viele schwache Linien des Abstreiks zu registrieren, ($400 \text{ nm} \sim 540 \text{ nm}$) damit können wir das Objektiv sorgfältig drehen, bis eine Skala von Linien zwischen 400 nm und 540 nm erscheint.

Wähle die Gesamtaufnahmezeit von etwa 4 s, dadurch wählen wir geeignete Zahl der Scans zur Mittelwertbildung

- Aufnahme des Spektrums der intensiven Linien

kurze Integrationszeit einstellen, Irisblende schließen so weit, dass die Na-D Linie nicht mehr in Sättigung. (zwischen 620 nm und 570 nm)

H. Lutz



扫描全能王 创建

3 Auswertung

Im Versuch wird ein computergesteuertes Gitterspektrometer eingesetzt, das den Spektralbereich einer Lichtquelle von 180 nm bis 950 nm registrieren kann. Mit dem Programm *OceanView* werden kontinuierlich Spektren im Scope- Modus durch Dunkelstrom Korrektur aufgenommen und grafisch dargestellt.

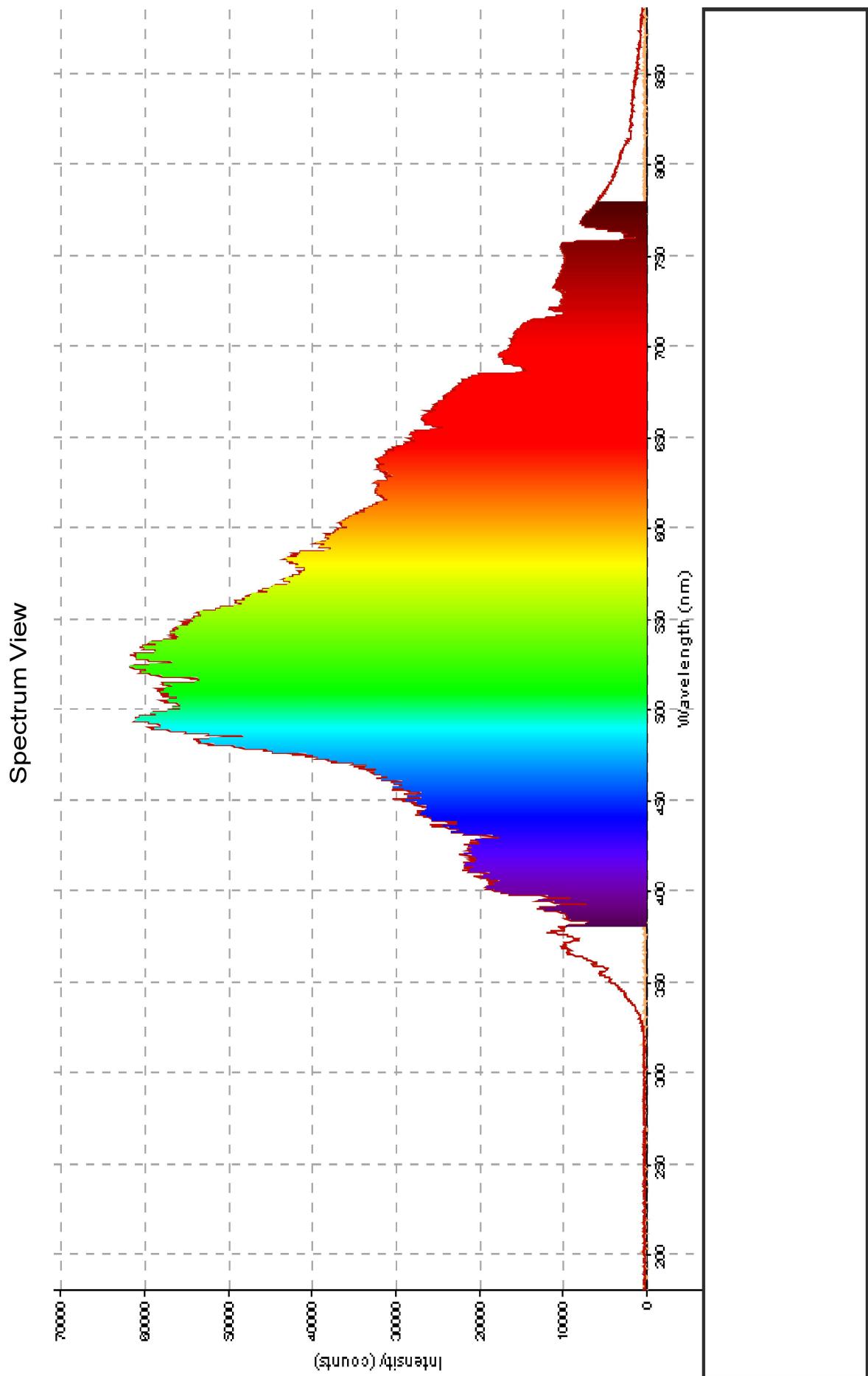
3.1 Sonnenspektrum

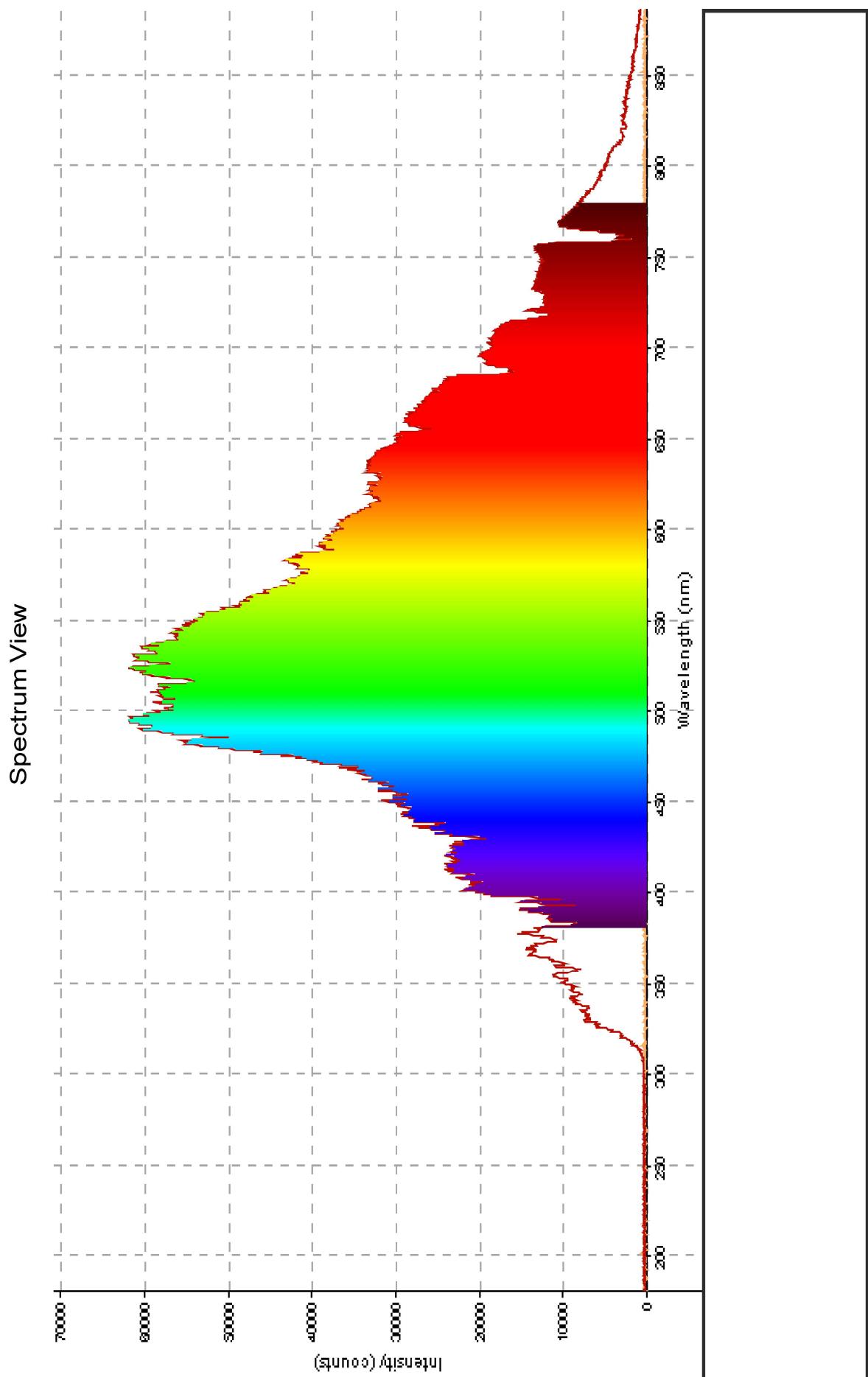
Es wurden Spektren des Himmelslichts wurde geöffneten Fenster hinter dem Fensterglas aufgenommen (Das erste Spektrum mit Fenster und das zweite ohne Fenster). Die Absorption des Glases kann nun mithilfe der folgenden Formel aus den Graphen berechnet werden:

$$A_G = 1 - \frac{I_m(\lambda)}{I_o(\lambda)} \quad (5)$$

Hierbei steht I_m für die Intensität mit der Fensterscheibe und I_o für die Intensität ohne Fensterscheibe.⁴:

⁴Quelle: Python Code 1





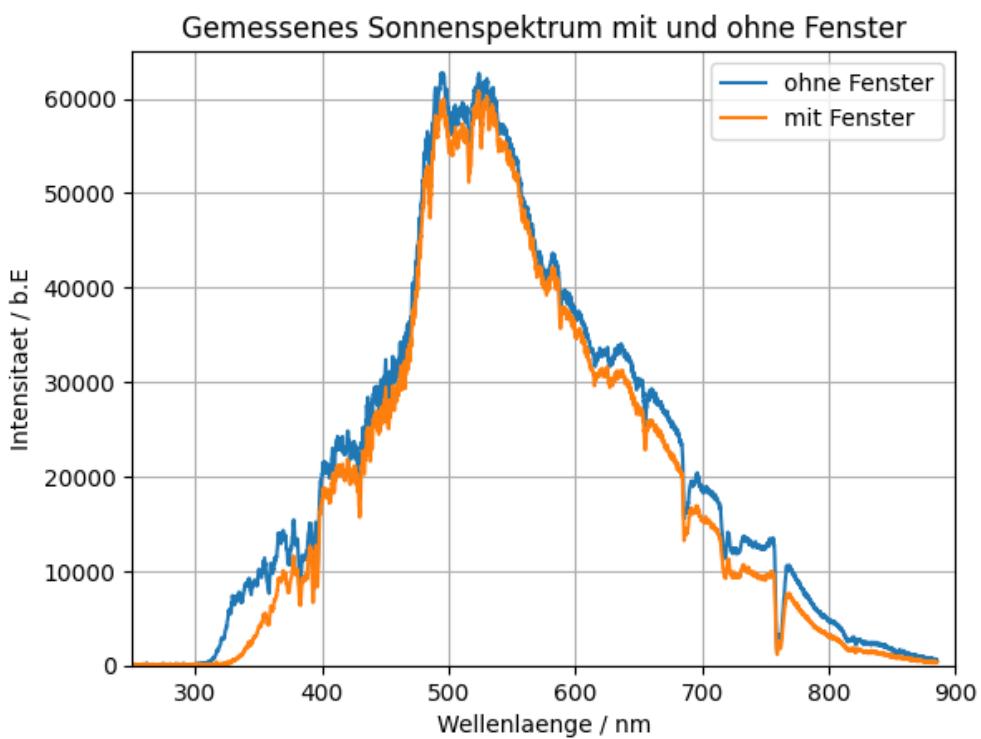


Abbildung 4: Gemessenes Sonnenspektrum mit und ohne Fenster

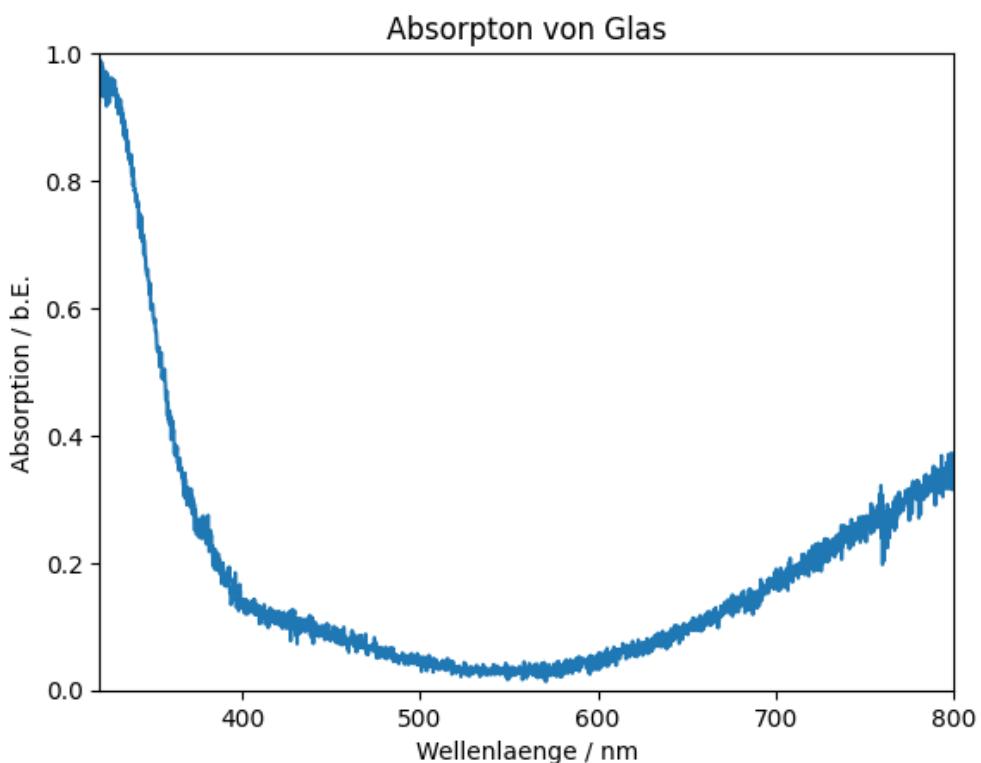


Abbildung 5: Absorptio von Glas

Das Sonnenlicht zeigt ein kontinuierliches Spektrum mit zahlreichen durchgezogenen "Fraunhoferlinien". Diese Linien entstehen durch die Absorption von Gasen in der Atmosphäre insbesondere wie H_2O , O_2 und CO_2 , die gut erkennbare Absorptionslinien oder -banden im Lichtspektrum erzeugen. O_3 absorbiert breitbandig im Bereich von 200 bis 700 nm und filtert dadurch einen erheblichen Anteil der UV-Strahlung aus dem Sonnenlicht heraus.

Aus Abbildung 5 wird gezeigt, dass kurze Wellenlängen, insbesondere im UV-Bereich, sehr stark absorbiert werden, was darauf hinweist, dass Glas einen ausgezeichneten UV-Schutz bietet. Im sichtbaren Bereich ist die Absorption vergleichsweise gering. Jedoch nimmt die Absorption im Infrarotbereich wieder etwas zu.

Nun wird ein neues Diagramm des Sonnenspektrums (Direktlicht oder Himmelslicht ohne Glas) im Wellenlängenbereich von 350 nm bis 800 nm gezeichnet, um die wichtigsten Fraunhoferlinien zu identifizieren⁵:

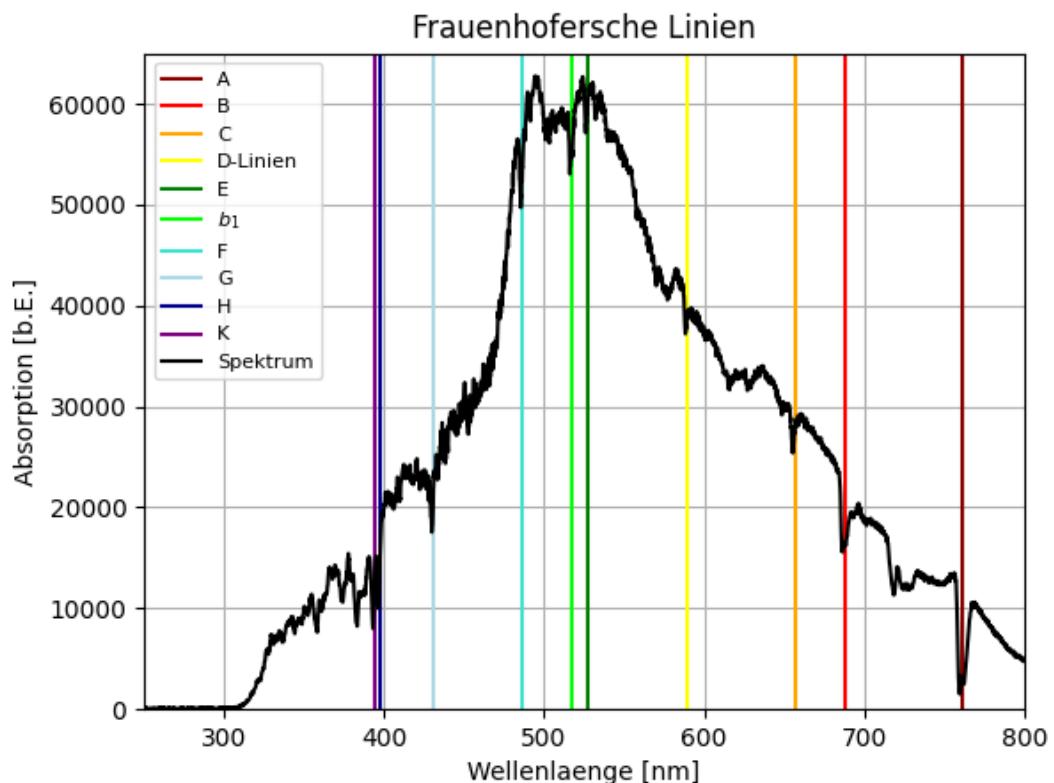


Abbildung 6: Identifikation der Fraunhoferlinien

⁵Python Code 2

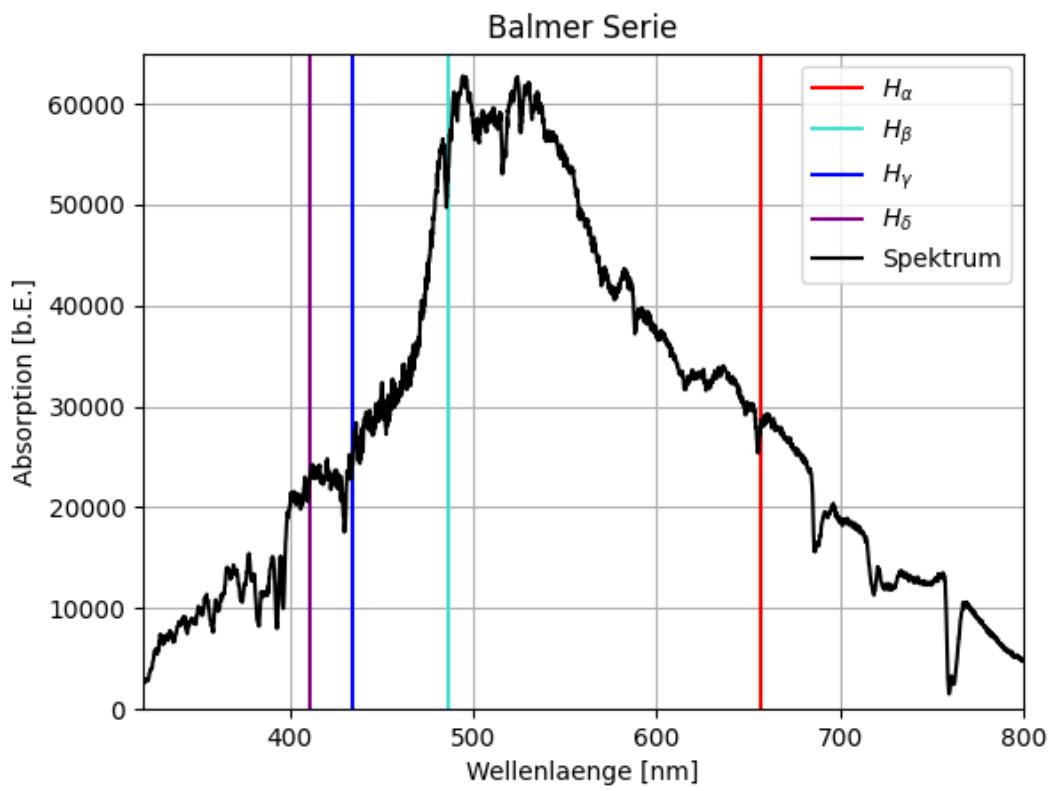


Abbildung 7: Balmerserie

Es werden die gemessenen Wellenlängen der Fraunhoferlinien und der Balmerserie in der folgenden Grafik mit den theoretischen Werten verglichen, die in der Praktikumsanleitung angegeben sind:

Symbol	λ_{theo} [nm]	λ_{exp} [nm]	$\Delta\lambda$ [nm]	Elemente/Molekül
A	759,4	756,2	3,2	telluric oxygen
B	686,7	686,2	0,5	telluric oxygen
C	656,3	657,4	1,1	hydrogen
D_1	589,6	591,9	2,3	sodium
D_2	589,0	589,9	0,9	sodium
D_3	587,6	585,3	2,3	helium
E	527,0	526,4	0,6	iron and calcium
b_1	518,4	518,9	0,5	magnesium
F	486,1	485,6	0,5	hydrogen
G	430,8	432,6	1,8	iron and calcium
H	396,8	395,1	1,7	calcium
K	393,4	390,8	2,6	calcium

Tabelle 1: Fraunhoferlinien und Balmerserie im Sonnenspektrum

Es ist erkennbar, dass die Abweichungen zwischen den theoretisch berechneten und

experimentell ermittelten Werten im Langwellen- und Kurzwellenbereich deutlich signifikanter sind als die Abweichungen im mittleren Bereich. Die genauen Gründe für diese Abweichungen werden später in der Diskussion erläutert.

3.2 Auswertung der unterschiedlichen Lichtquellen⁶

Nun sollen unterschiedliche Lichtquellen qualitativ beobachtet, und geforscht werden. Zu Begin werden die Aufnahmen des Spektrums von LED mit 7 verschiedenen Signalen durchgeführt:

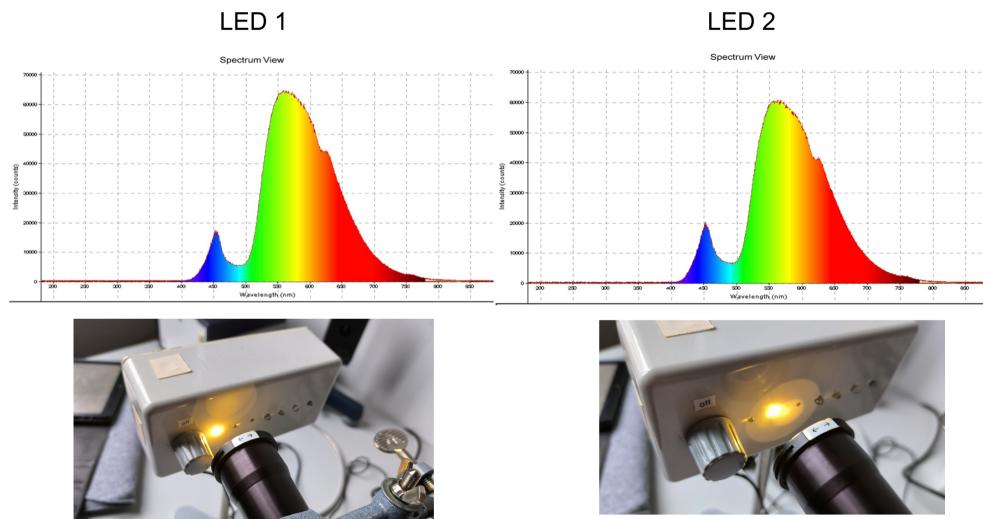


Abbildung 8: LED 1 und 2 sowie ihre Spektren

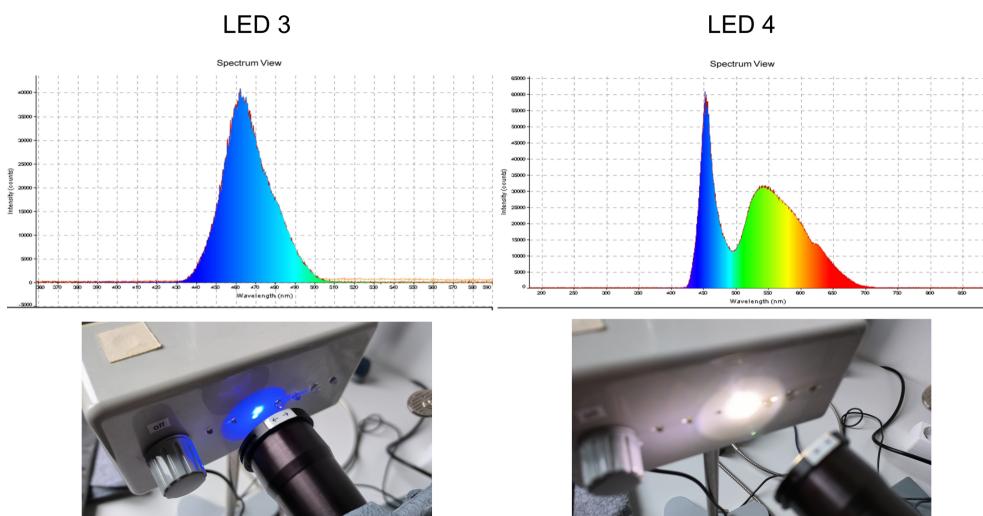


Abbildung 9: LED 3 und 4 sowie ihre Spektren

⁶Python-Code 4

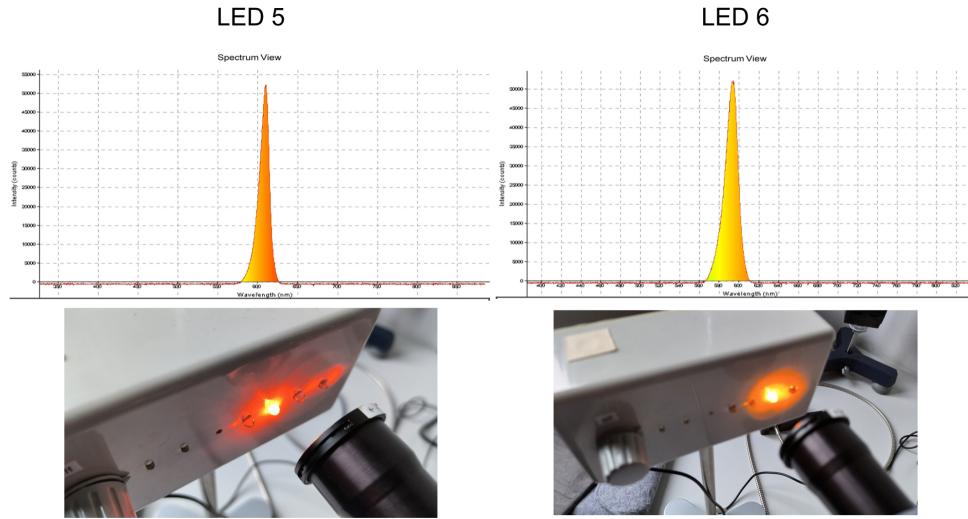


Abbildung 10: LED 5 und 6 sowie ihre Spektren

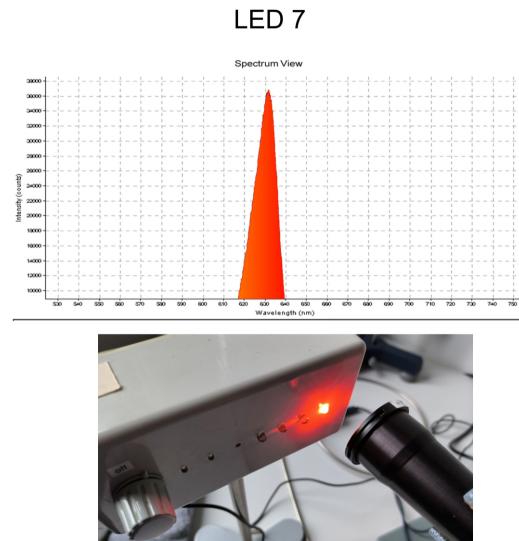


Abbildung 11: LED 7 sowie sein Spektrum

Die Lichtemission bei Leuchtdioden (LEDs), abgekürzt für "Light Emitting Diode", entsteht durch die Rekombination von Elektron-Loch-Paaren in einem pn-Übergang. Bei den sogenannten "direkten Halbleitern" wird dabei ein Photon emittiert. Je nach der chemischen Zusammensetzung wird Licht mit unterschiedlicher Farben freigesetzt, dabei manche wichtige Halbleitermaterialien: GaAs (IR), AlGaAs (rot, IR), GaAsP (rot, orange, gelb), GaP (grün), SiC (blau).

Inzwischen gibt es auch LEDs, die weißes Licht abstrahlen (LED 4 wie Abbildung). Solche LED-Lichtquellen bestehen entweder aus drei separaten LEDs in den Farben Rot, Grün und Blau (RGB), deren Licht additiv gemischt wird, um weißes Licht zu erzeugen. Alternativ werden weiße LEDs aus blauen LEDs hergestellt, die ähnlich wie

Leuchtstoffröhren mit einer zusätzlichen Fluoreszenzschicht beschichtet sind.

LED 1,2 und 4 besitzen einen breiten Wellenbereich, setzen sich also aus vielen Materialien zusammen die mittleren Wellenlängen der abgestrahlten Lichts wollen wir nicht bestimmen. Für LED 3 gibt es ein starkes Peak in der Nähe von $\lambda = (467 \pm 5)$ nm, während LED 5, 6, 7 ganz schmale Wellenlängenbereiche haben und besitzen die mittleren Wellen grob geschätzt $\lambda_5 = (620 \pm 5)$ nm, $\lambda_6 = (592 \pm 5)$ nm und $\lambda_7 = (600 \pm 5)$ nm.

Nun vergleichen wir die Spektren der anderen Lichtquellen:

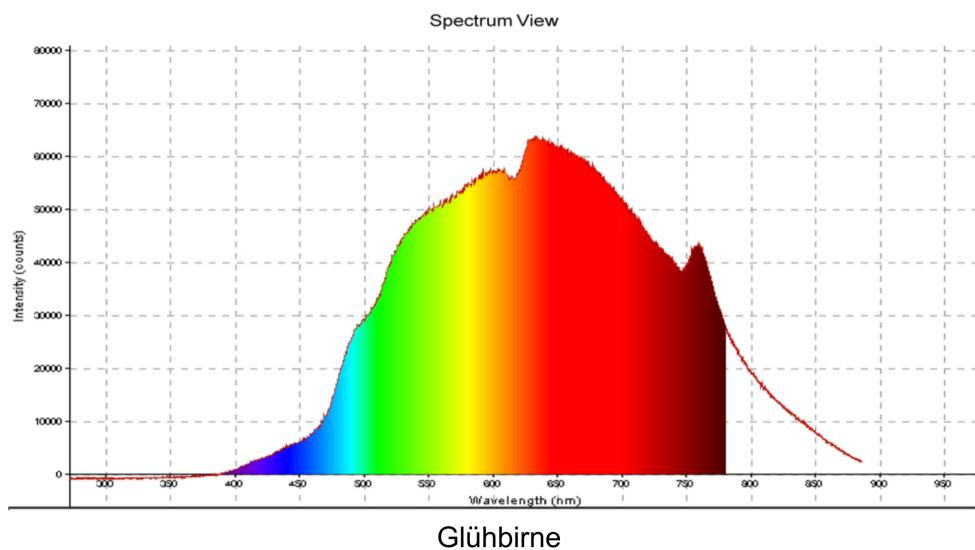


Abbildung 12: Glühbirne

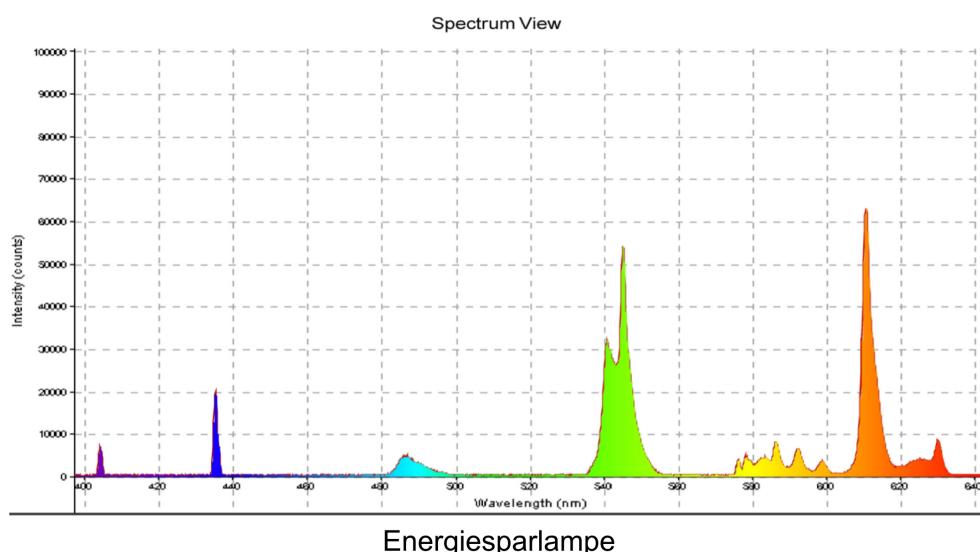


Abbildung 13: Energiesparlampe

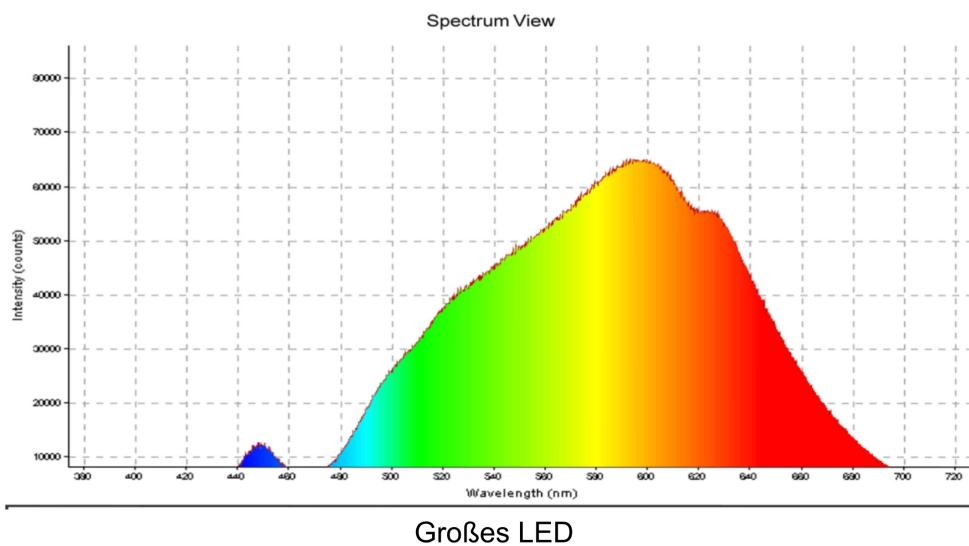


Abbildung 14: Großes LED

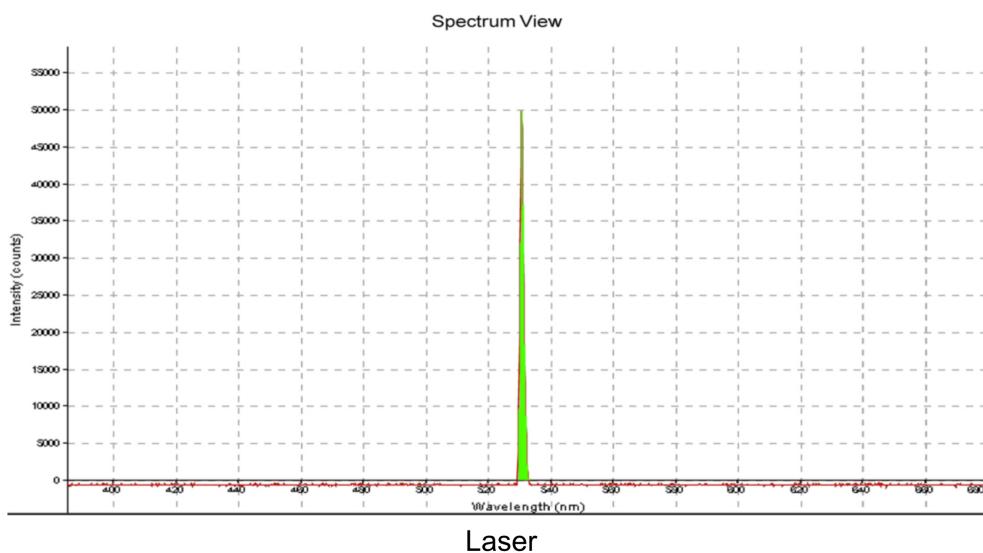


Abbildung 15: Laser

Die Energiesparlampe hat ein diskretes Spektrum mit über 10 Spektrallinien. Im Allgemeinen erzeugen Energiesparlampen Licht mithilfe von Gasentladung und Phosphorbeschichtungen auf der Innenseite der Röhre. Die Lampe enthält eine kleine Menge Quecksilberdampf und Argongas. Beim Einschalten erzeugt die Elektronik in der Lampe eine elektrische Entladung, die das Quecksilbergas zum Leuchten anregt und emittiert ultraviolettes (UV) Licht. Die Innenseite der Röhre ist mit Phosphor beschichtet, der auf das UV-Licht reagiert und sichtbares Licht erzeugt. Die Innenseite erzeugt demnach nur Lichts mit bestimmten Wellenlängen, washalb aus dem Spektrum viele Peaks beobachtet werden können.

Das Spektrum der LED-Lampe hat ein breites Spektrum (440-700nm) im Vergleich zu den LEDs vorher. Sein Höhepunkt liegt in etwa bei 590nm und strahlt ein gelb-weißliches Licht aus.

Das Spektrum einer Glühbirne erreicht seinen Höhepunkt bei ungefähr 640 nm, was rotem Licht entspricht. Es erstreckt sich über einen Bereich von 400 bis 900 nm, wobei der Anteil im roten Bereich (über 600 nm) stärker vertreten ist als im blauen Bereich.

Die schmale Bandbreite des Laserlichts lässt sich darauf zurückführen, dass der Laser eine sehr große Kohärenz besitzt. Diese Kohärenz entsteht durch die Stimulierte Emission im Lasermedium, bei der die emittierten Photonen alle in Phase sind, was zu einer genau definierten Wellenlänge oder Frequenz führt.

3.3 Auswertung des Natriumspektrums

Es wurden Linien kleiner und großer Intensität aufgenommen und zusammen in Python importiert und Daten in ein interaktives Diagramm eingetragen⁷:

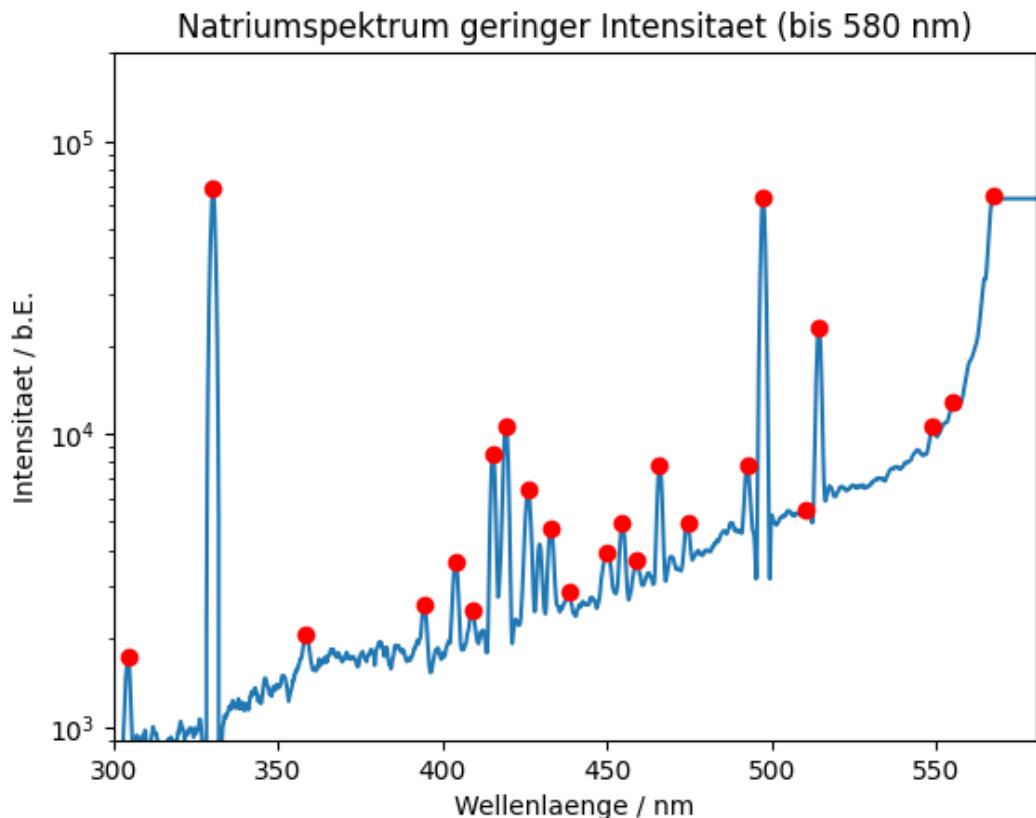


Abbildung 16: Natriumspektrum im Bereich 300 nm bis 580 nm

⁷Python-Code 5

Die daraus ablesbaren Peaks (aus Python) werden nun in ein Diagram eingetragen (Wir sollen jedoch beachten, dass die Wellenlänge nicht zwangsläufig den Peaks entsprechen):

Linie	1	2	3	4	5	6	7	8	9
Wellenlänge [nm]	304,5	330,4	358,4	394,5	404,0	409,1	419,3	426,0	432,8
Fehler [nm]	1,8	1,0	1,0	0,8	1,2	1,0	1,5	0,8	1,0

Linie	10	11	12	13	14	15	16	17	18
Wellenlänge [nm]	438,3	449,8	454,6	458,8	465,8	474,4	492,6	497,3	514,2
Fehler [nm]	1,3	1,2	1,6	1,5	1,1	1,2	1,6	1,7	1,4

Linie	14	15	16	17
Wellenlänge [nm]	548,8	555,0	567,1	284,0
Fehler [nm]	1,2	1,3	1,5	1,4

Tabelle 2: Intensive Linien im Bereich von 300nm bis 580nm

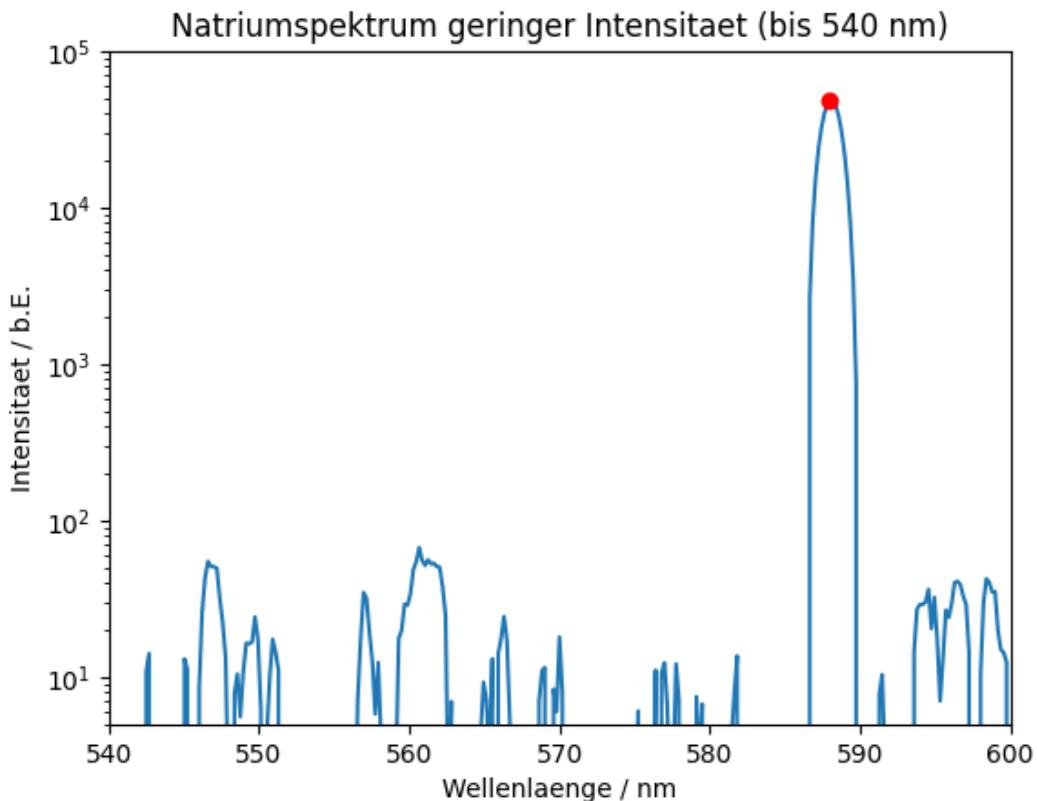


Abbildung 17: NaD Linie zwischen 540nm und 600nm

Position des Peaks: $\lambda = (588,0 \pm 1,4) \text{ nm}$

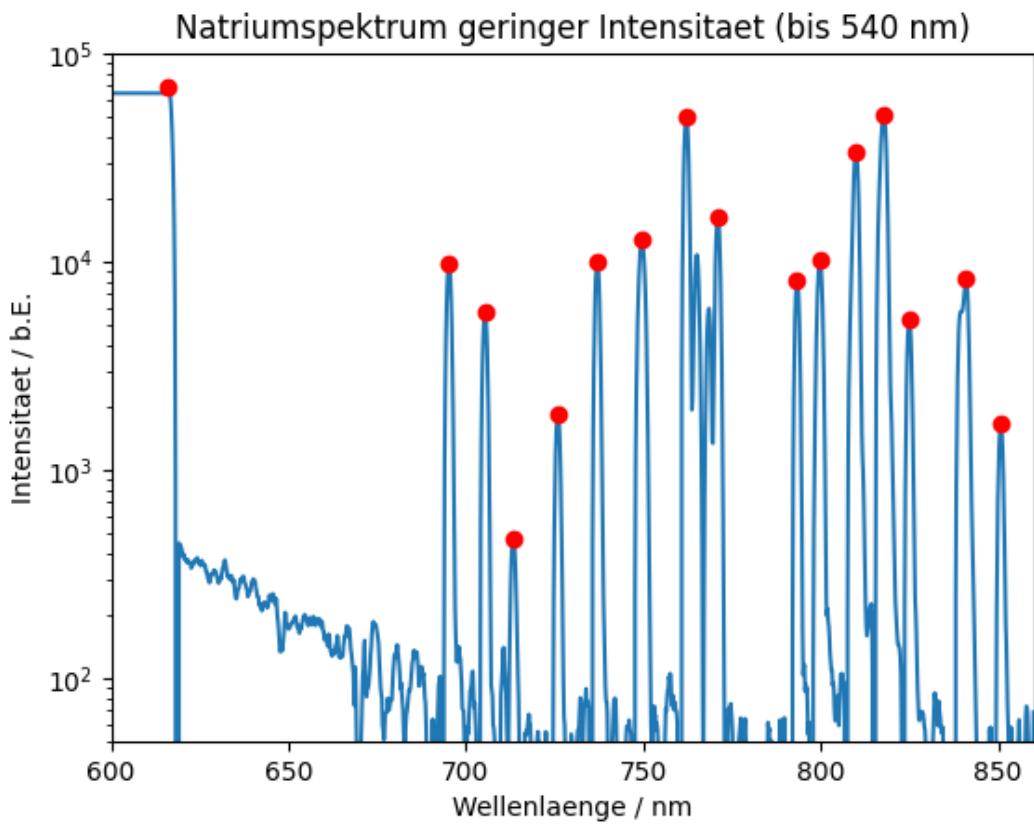


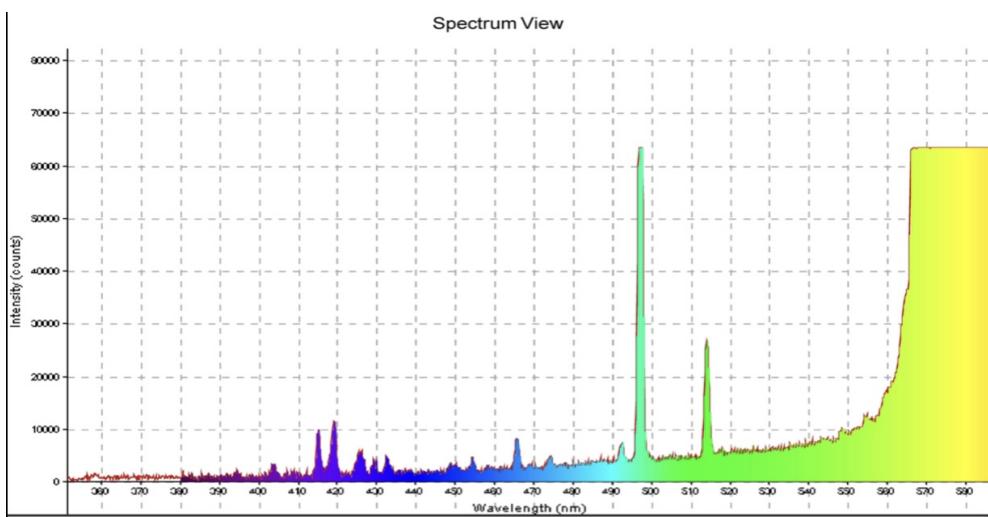
Abbildung 18: Spektrum im Bereich 600 nm bis 850 nm

Linie	1	2	3	4	5	6	7	8	9
Wellenlänge [nm]	705,4	713,4	726,0	737,0	749,5	762,1	771,0	793,3	799,7
Fehler [nm]	1,1	1,2	0,7	1,3	1,7	1,6	1,6	1,3	1,5

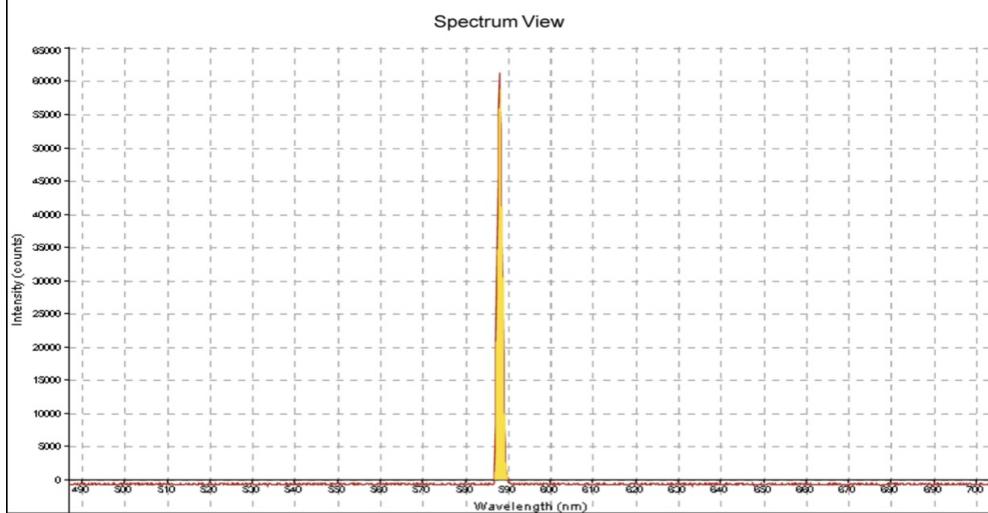
Linie	10	11	12	13	14	15	16
Wellenlänge [nm]	809,9	817,7	825,1	841,0	850,8	695,3	616,1
Fehler [nm]	1,1	1,2	0,7	1,3	1,4	1,6	1,6

Tabelle 3: Intensive Linien im Bereich von 600nm bis 850nm

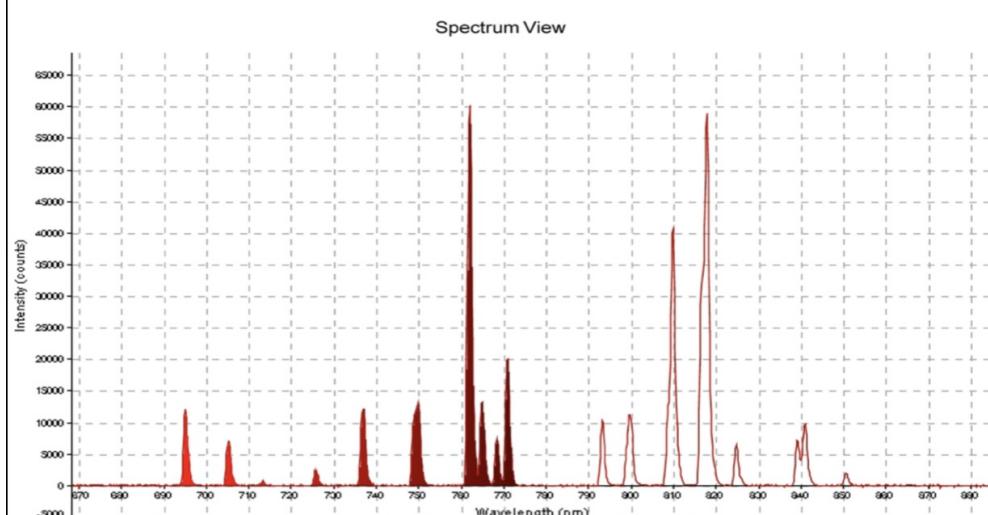
Allerdings ist es wichtig zu beachten, dass Python aufgrund numerischer Fehler in einigen Fällen möglicherweise falsche Informationen über die Peaks liefert. Numerische Ungenauigkeiten können die Genauigkeit der Ergebnisse beeinträchtigen und zu falschen Informationen führen, insbesondere bei der Analyse von Spitzen oder extremen Werten in den Daten.



300 nm bis 540 nm



NaD Linie



600 nm bis 850 nm

3.4 Zuordnung der gefundenen Linien zu Serien

Die Linien, die entdeckt wurden, müssen zunächst den drei Serien des Natriums zugeordnet werden. Dazu ist es erforderlich, die ungefähren Wellenlängen der Linien für jede Serie zu berechnen und sie dann mit den beobachteten Linien zu vergleichen.

3.4.1 1. Nebenserie

Für diese Serie kann ein Korrekturterm für die d-Energieniveaus von Null angenommen werden. Damit gilt:

$$\frac{hc}{\lambda_m} = \frac{E_{Ry}[eV]}{m^2} - E_{3p}[eV] \quad (6)$$

$$\lambda_m[nm] \approx (1.2398 \cdot 10^3 [nm \cdot eV]) / \left(\frac{-13.605eV}{m^2} - E_{3p}[eV] \right) \quad (7)$$

$$\Delta \lambda_m = (1.2398 \cdot 10^3 [nm \cdot eV]) / \left(\left(\frac{-13.605eV}{m^2} - E_{3p}[eV] \right)^2 \right) \cdot \Delta E_{3p} \quad (8)$$

Hier beschreibt m die Hauptquantenzahl des Energieniveaus; E_{3p} die Energie des 3p Zustands und $E_{Ry} = -13,605 \text{ eV}$ die Rydbergenergie. Der Vorfaktor hc berechnet sich zu $hc = 1,2398 \times 10^3 \text{ [nm eV]}$.

Die gemessene Linie im Bereich von 819 nm wird mit dem Wert m = 3 ($\lambda_3 = 819 \text{ nm}$) assoziiert, dadurch wird daher der entsprechende Energieunterschied berechnet als:

$$E_{3p} = (3,026 \pm 0,004) \text{ eV} \quad (9)$$

Anschließend können die Wellenlängen λ_{th} mithilfe von Gl.(7), (8) ausgerechnet⁸ und in Tabelle (2) mit den experimentell bestimmten Werten λ_{exp} verglichen werden:

m	λ_{exp} [nm]	λ_{th} [nm]	$\sigma - \text{Abweichung}$
3	$817,7 \pm 1,2$	$819,0 \pm 2,3$	0,5
4	$567,1 \pm 1,5$	$570,0 \pm 1,1$	1,6
5	$497,3 \pm 1,7$	$499,7 \pm 0,9$	1,3
6	$465,8 \pm 1,1$	$468,3 \pm 0,8$	1,8
7	$449,8 \pm 1,2$	$451,2 \pm 0,7$	1,0
8	$438,3 \pm 1,3$	$440,8 \pm 0,7$	1,7
9	$432,8 \pm 1,0$	$433,9 \pm 0,7$	0,9
10	—	$429,2 \pm 0,6$	—
11	$426,0 \pm 0,8$	$425,6 \pm 0,6$	0,4
12	$419,3 \pm 1,5$	$423,0 \pm 0,6$	2,3

Tabelle 4: Erwartete Linien der 1.Nebenserie

⁸Python-Code 6

Wir haben die Wellenlängen der Peaks aus experimentellen Messungen ausgewählt, die den theoretisch ermittelten Werten am nächsten kommen. Solche Wellenlängen entsprechen auch den größten Peaks in unseren Spektren. Allerdings können wir bei dem Wert $\lambda = (429, 2 \pm 0, 6) \text{ nm}$ keine zugehörige Wellenlänge finden. Alle berechneten Fehlerabweichungen betragen innerhalb von 3σ und sind somit nicht signifikant.

3.4.2 2. Nebenserie

Es wurde angenommen ein gemessener Wert im Experiment von $\lambda = 589 \text{ nm}$ für die D-Linie gefunden (NaD Linie, siehe Abbildung 17), die dem Übergang $3p \rightarrow 3s$ entspricht. Daraus ergibt sich:

$$E_{3s} = E_{3p} - 1.2398 \cdot 10^3 [\text{nm eV}] / \lambda \quad (10)$$

$$\Delta E_{3s} = \sqrt{(\Delta E_{3p})^2 + \left(\frac{hc}{\lambda^2} \Delta \lambda \right)^2} \quad (11)$$

Somit folgt eine Bindungsenergie von $E_{3s} = -5.130 \pm 0.008 \text{ eV}$. Es folgt der Korrekturfaktor:

$$\Delta_s = 3 - \sqrt{\frac{-13.605 \text{ eV}}{E_{3s}}} \quad \Delta(\Delta_s) = \frac{1}{2} \sqrt{\frac{-13.605 \text{ eV}}{E_{3s}^3}} \cdot \Delta E_{3s} \quad (12)$$

$$\Rightarrow \Delta_s = 1.3711 \pm 0.0012 \quad (13)$$

Nun können auch die Wellenlängen für $3 < m < 10$ bestimmt werden⁹:

$$\lambda_m [\text{nm}] \approx (1.2398 \cdot 10^3 [\text{nm} \cdot \text{eV}]) / \left(\frac{-13.605 \text{ eV}}{(m - \Delta_s)^2} - E_{3p} [\text{eV}] \right) \quad (14)$$

$$\Delta \lambda_m = \sqrt{\left(\frac{hc \cdot \Delta E_{3p}}{\left(\frac{E_{Ry}}{(m - \Delta_s)^2} - E_{3p} \right)^2} \right)^2 + \left(\frac{2E_{Ry}hc \cdot \Delta(\Delta_s)}{\left(\frac{E_{Ry}}{(m - \Delta_s)^2} - E_{3p} \right)^2 (m - \Delta_s)^3} \right)^2} \quad (15)$$

Diese Werte für die theoretischen Wellenlängen können wieder mit den experimentell bestimmten Werten verglichen werden:

⁹Python-Code 7

m	λ_{exp}	λ_{th}	Abweichung σ
4	-	$1173,8 \pm 5,2$	-
5	$616,1 \pm 1,6$	$622,4 \pm 1,4$	3,0
6	$514,2 \pm 1,4$	$518,7 \pm 0,9$	2,7
7	$474,4 \pm 1,2$	$477,6 \pm 0,8$	2,2
8	$454,6 \pm 0,8$	$456,5 \pm 0,7$	1,8
9	$438,3 \pm 1,3$	$444,1 \pm 1,6$	2,8

Tabelle 5: Erwartete Linien der 2. Nebenserie

Der gemessene Wert von $\lambda = (1173,8 \pm 5,2) \text{ nm}$ liegt deutlich außerhalb des erwarteten Messbereichs, weshalb keine entsprechende Wellenlänge gefunden werden kann. Alle berechneten Fehlerabweichungen liegen innerhalb von 3σ und sind nicht signifikant.

3.4.3 Hauptserie

Nun soll der benötigte Korrekturterm Δ_p aus E_{sp} berechnet werden:

$$\Delta_p = 3 - \sqrt{(-13.605 \text{ eV})/(E_{3p})} \quad (16)$$

$$\Delta(\Delta_p) = \frac{1}{2} \sqrt{(-13.605 \text{ eV})/(E_{3p}^3)} \cdot \Delta E_{3p} \quad (17)$$

$$\Rightarrow \Delta_p = 0.8794 \pm 0.0015 \quad (18)$$

Nun bestimmen wir wieder die Wellenlängen für $m = 4, 5$ mithilfe der folgenden Gleichungen:

$$\lambda_m [\text{nm}] \approx \frac{1.2398 \cdot 10^3 [\text{nm} \cdot \text{eV}]}{-13.605 \text{ eV}/(m - \Delta_p)^2 - E_{3s} [\text{eV}]} \quad (19)$$

$$\Delta \lambda_m = \sqrt{\left(\frac{hc \cdot \Delta E_{3s}}{\left(\frac{E_{Ry}}{(m - \Delta_p)^2} - E_{3s} \right)^2} \right)^2 + \left(\frac{2E_{Ry}hc \cdot \Delta(\Delta_p)}{\left(\frac{E_{Ry}}{(m - \Delta_p)^2} - E_{3s} \right)^2 (m - \Delta_p)^3} \right)^2} \quad (20)$$

Die ermittelten theoretischen Werte werden auch mit den Messungen verglichen¹⁰:

m	λ_{exp}	λ_{th}	Abweichung σ
4	$330,4 \pm 1,0$	$332,1 \pm 0,7$	1,4
5	$284,0 \pm 1,4$	$286,4 \pm 0,5$	1,6

Tabelle 6: Erwartete Linien der Hauptserie

Alle Fehlerabweichungen liegen innerhalb von 3σ und sind damit nicht signifikant.

¹⁰Python-Code 8

3.5 Bestimmung der Serienenergien der l-abhangigen Korrekturfaktoren

Bevor wir ien die Rydbergenergie E_{Ry} , E_{3p} und die Korrekturterme Δ_d und Δ_s zu bestimmen wollen wir zuerst eine kurze Zusammenfassung fur die berechneten 3 Serien machen:

m	3	4	5	6	7
λ_{exp} [nm]	$817,7 \pm 1,2$	$567,1 \pm 1,5$	$497,3 \pm 1,7$	$465,8 \pm 1,1$	$449,8 \pm 1,2$

m	8	9	10	11	12
λ_{exp} [nm]	$438,3 \pm 1,3$	$432,8 \pm 1,0$	-	$426,0 \pm 0,8$	$419,3 \pm 1,5$

Tabelle 7: 1. Nebenserie

m	4	5	6	7	8	9
λ_{exp} [nm]	-	$616,1 \pm 1,6$	$514,2 \pm 1,4$	$474,4 \pm 1,2$	$454,6 \pm 0,8$	$438,3 \pm 1,3$

Tabelle 8: 2. Nebenserie

m	4	5
λ_{exp} [nm]	$330,4 \pm 1,0$	$284,0 \pm 1,4$

Tabelle 9: Hauptserie

Aus den gemessenen Wellenlangen konnen nun E_{Ry} und E_{3p} sowie die Korrekturterme Δ_s und Δ_d bestimmt werden, indem zuerst die Wellenlangen in Abhangigkeit von den Quantenzahlen mit einer Fit-Funktion ausgewertet werden. Anhand der Gleichung:

$$\lambda_m[nm] \approx 1,2398 \times 10^3 [nm\ eV] / [E_{Ry}/(m - \Delta_d)^2 - E_{3p}] \quad (21)$$

werden E_{Ry} , E_{3p} und Δ_d ermittelt als¹¹:

$$E_{Ry} = (-13.992 \pm 0.617) \text{ eV} \quad (22)$$

$$E_{3p} = (-3.047 \pm 0.008) \text{ eV} \quad (23)$$

$$\Delta_d = -0.023 \pm 0.060 \quad (24)$$

¹¹Python-Code 9

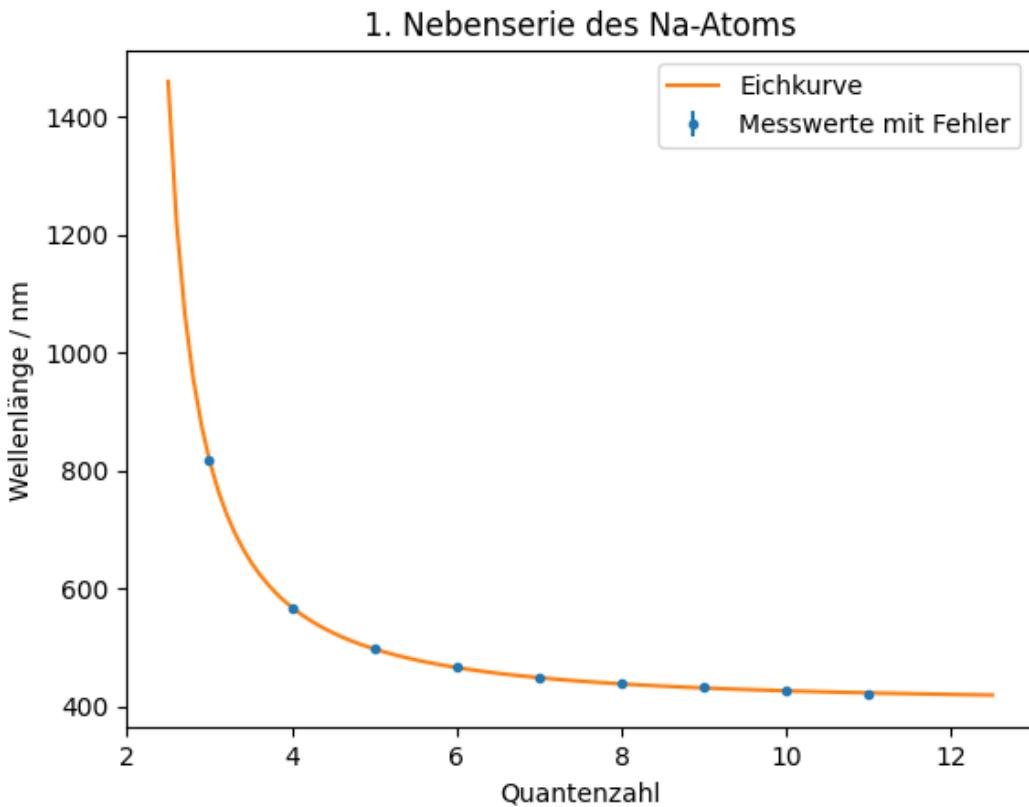


Abbildung 19: Eichkurve der 1. Nebenserie

Wir vergleichen die Rydbergkonstante mit dem Literaturwert $E_{RyL} = 13,606 \text{ eV}$, die Energieniveau in 3p mit dem Wert in (3.4.1) $E'_{3p} = (3,026 \pm 0,004) \text{ eV}$. Die Fehlerabweichungen ergeben sich als:

$$\frac{|E_{Ry} - E_{RyL}|}{\Delta E_{Ry}} \approx 0,63 \quad (25)$$

$$\frac{|E_{3p} - E'_{3p}|}{\sqrt{(\Delta E_{3p})^2 + (\Delta E'_{3p})^2}} \approx 2,35 \quad (26)$$

Beide Fehler befinden sich innerhalb von 3σ und sind also nicht signifikant. Wir verzichteten hier auf die Berechnung des Korrekturterms Δ_d , da der Literaturwert einerseits nicht verfügbar ist und andererseits der Fehler selbst bereits das Doppelte des potenziellen Δ_d beträgt, was später in Diskussion weiter betrachtet werden muss.

Anschließend möchten wir noch auf die Gute des Fits eingehen und die χ^2 -Summe berechnen, um die bestimmten Parameter auf ihre Genauigkeit zu überprüfen:

$$\chi = \sum_i^N \left(\frac{\text{Funktionswert}_i - \text{Messwert}_i}{\text{Fehler}_i} \right)^2 \quad (27)$$

$$\chi_{red}^2 = \frac{\chi^2}{\text{Freiheitsgrad}} \quad (28)$$

Der Freiheitsgrad berechnet sich aus der Anzahl der Messwerte abzüglich der Zahl der Fitparameter¹²:

$$\chi^2 = 9,73 \quad \chi_{red}^2 = 1,62 \quad (29)$$

Die Fitwahrscheinlichkeit, dass bei einer erneuten Messung ein χ^2 -Wert größer oder gleich dem bereits berechneten χ_{exp}^2 ist, hängt von der χ^2 -Verteilung mit den Freiheitsgraden des Experiments ab. Es gilt¹³:

$$P = 1 - F(\chi_{exp}^2, k) \approx 14\% \quad (30)$$

Hierbei steht k für die Freiheitsgrade, die sich aus der Anzahl der Messungen und Parameter ergeben, und χ_{exp}^2 die bereits berechnete Chi-Summe. Zur genauen Berechnung benötigt man den χ^2 -Wert χ_{exp}^2 und die Anzahl der Freiheitsgrade, um die entsprechende Verteilungsfunktion zu nutzen.

Es wird nun die Anpassung für die zweite Nebenserie wiederholt. Hier sollen wir die folgende Gleichung benutzen¹⁴:

$$\lambda_m[nm] \approx 1,2398 \times 10^3 [nm \text{ eV}] / [E_{Ry}/(m - \Delta_s)^2 - E_{3p}] \quad (31)$$

Die Rydbergkonstante, das Energieniveau sowie der Korrekturterm ergeben sich als:

$$E_{Ry} = (-14.793 \pm 3.195) \text{ eV} \quad (32)$$

$$E_{3p} = (-3.058 \pm 0.040) \text{ eV} \quad (33)$$

$$\Delta_s = 0.238 \pm 0.341 \quad (34)$$

¹²Python-Code 10

¹³Python-Code 11

¹⁴Python-Code 12

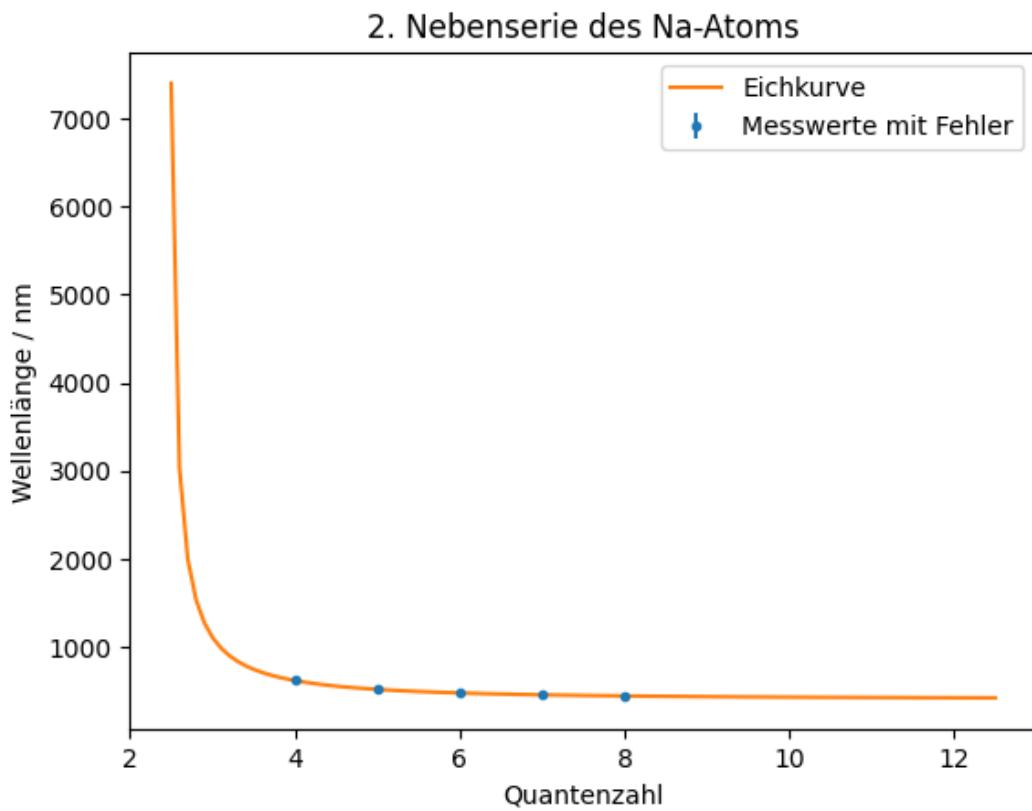


Abbildung 20: Eichkurve der 2. Nebenserie

Wir berechnen wieder die Fehlerabweichungen:

$$\frac{|E_{Ry} - E_{RyL}|}{\Delta E_{Ry}} \approx 0,37 \quad (35)$$

$$\frac{|E_{3p} - E'_{3p}|}{\sqrt{(\Delta E_{3p})^2 + (\Delta E'_{3p})^2}} \approx 0,80 \quad (36)$$

Die χ^2 -Summe, sowie die Fitwahrscheinlichkeit lauten;

$$\chi^2 = 6.74 \quad \chi^2_{red} = 3.37 \quad P = 3.0\% \quad (37)$$

Alle Werte sind in die folgende Tabelle eingetragen:

	1. Nebenserie	2. Nebenserie	$\sigma_{1,Lit}$	$\sigma_{2,Lit}$	$\sigma_{1,2}$
E_{Ry}	$-13,992 \pm 0,617 \text{ eV}$	$-14,793 \pm 3,195 \text{ eV}$	0,63	0,37	0,25
E_{3p}	$-3,047 \pm 0,008 \text{ eV}$	$-3,058 \pm 0,040 \text{ eV}$	2,35	0,80	0,04
$\Delta_{d,s}$	$-0,023 \pm 0,060$	$0,238 \pm 0,341$	-	-	-
χ^2	9,73	6,74	-	-	-
χ^2_{red}	1,62	3,37	-	-	-
P	14%	3%	-	-	-

Tabelle 10: Parameter der 1. und 2. Nebenserie, sowie Abweichungen

4 Zusammenfassung und Diskussion

In diesem Experiment wurden die Spektren verschiedener Lichtquellen erfasst und sowohl qualitativ als auch quantitativ analysiert. Im ersten Abschnitt des Experiments wurde die Absorption von Glas untersucht, indem wir zwei Spektren - eines mit und eines ohne Fenster - verglichen. Aufgrund der ungünstigen Wetterbedingungen könnten die aufgenommenen Spektren jedoch erheblich von dem tatsächlichen Sonnenspektrum abweichen. Trotz dieser Schwierigkeiten wurde eine bemerkenswert hohe UV-Absorption festgestellt. Danach haben wir zusätzlich die Fraunhoferlinien betrachtet und stellten fest, dass die vom Sonnenspektrum abgeleiteten Wellenlängen dieser Linien geringfügig von den Literaturwerten abwichen. Leider trotz der Compensation des Dunkelstroms während des Experiments blieben gewisse Rauschung in der Intensitätsverteilung, insbesondere bei den gemessenen Fraunhoferlinien, die bei der Auswertung mit Python berücksichtigt werden sollte. Insgesamt entsprachen unsere Messwerte jedoch weitgehend den Literaturwerten.

Es ist jedoch erkennbar, dass die Abweichungen zwischen den theoretisch berechneten und experimentell ermittelten Werten im Langwellen- und Kurzwellenbereich deutlich signifikanter sind als im mittleren Bereich. Dies lässt sich darauf zurückführen, dass in den äußeren Bereichen des Spektrums stärkere Störeinflüsse oder Hintergrundrauschen auftreten können, welche die Messgenauigkeit beeinträchtigen. Die atmosphärische Absorption und Streuung von Licht im Hochfrequenzbereich (also im Kurzwellenbereich) spielen hier eine entscheidende Rolle (vgl. Abschnitt 1.3). Im Kurzwellenbereich können atmosphärische Effekte wie Streuung die Messergebnisse beeinflussen, während im Langwellenbereich Interferenzen oder Wechselwirkungen zwischen Licht und anderen Materiepartikeln die Messungen beeinträchtigen können.

Im zweiten Teil des Experiments wurden die Spektren verschiedener Lichtquellen qualitativ analysiert. Ursprünglich sollte mithilfe von SectraSuite ein Diagramm erstellt werden, das die zu vergleichenden Spektren zusammenfasst. Stattdessen benutzen wir separat die aufgenommenen Spektren, was ein bisschen redundant erscheint. Leider mussten wir das Spektrum einer Halogenlampe auslassen, da die dafür zur Verfügung gestellte Lampe außer Betrieb war.

Anschließend wurden die Spektren für schwache und starke Linien wie NaD quantitativ ausgewertet. Um die charakteristischen Peaks in der Nähe der NaD-Linie zu analysieren, erweiterten wir den Wellenlängenbereich von $300 - 540 \text{ nm}$ bis 580 nm , da andernfalls aufgrund der Dominanz der NaD-Linie andere Peaks von Python vernachlässigt wurden und es zu numerischen Fehlern kam (Bei kleinen Werten treten leicht „Auslöschung“ auf). Die Funktion „`find_peaks`“ in Python lieferte nur relative hohe Werte im Vergleich zu ihren Nachbarn (lokale Maxima), weshalb in Abschnitt 3.3 alle von Python ermittelten Werte aufgeführt wurden, um Zuordnungsmöglichkeiten zu ermöglichen. Dies kann jedoch zu Ungenauigkeiten bei der Auswertung führen, insbesondere bei übersättigten Peaks, die manuell angenähert wurden.

Bei der Zuordnung der gefundenen Linien im zweiten Teil haben wir zunächst theoretische Werte ermittelt. Im zweiten Teil der Zuordnung der identifizierten Linien haben wir zunächst die theoretischen Werte bestimmt und dann in unserer Messung nach den experimentell nächstgelegenen Linien zu den theoretischen Werten gesucht. Für diese gefundenen Linien wird ein Abgleich durchgeführt, um sicherzustellen, dass die identifizierten Linien tatsächlich den Peaks in unserem Spektrum entsprechen. Solche Herangehensweise könnte natürlich zu potenziellen Fehlern führen. Dennoch lagen die Abweichungen der Messwerte der ersten Nebenserien alle innerhalb eines insignifikanten Bereichs. Die NaD-Linie, die wir erfasst haben, lag sehr nahe am Literaturwert von $\lambda = 589 \text{ nm}$. Bei der Auswertung der zweiten Nebenserien lagen alle ermittelten Linien ebenfalls innerhalb des 3σ -Bereichs.

Als das letzte Teil haben wir die Rydberg-Konstante, verschiedene Energieniveaus sowie Korrekturterme ausgewertet, wobei die Fehlerabweichungen nicht signifikant waren. Es ist jedoch zu beachten, dass die Fehler der beiden Korrekturterme jeweils über doppelt so groß wie ihre eigenen Werte sind. Dies liegt einerseits daran, dass wir möglicherweise in Abschnitt 3.4 die Fehler der Wellenlängen etwas überschätzt haben. Andererseits kann aufgrund der geringen Beiträge der beiden Werte während der numerischen Berechnung auch eine Auslöschung auftreten, was zu starken Fehlern führen kann.

5 Anhang

Auswertung Python 234

December 9, 2023

0.0.1 Versuch 234 Auswertung

Auswertung 1: Sonnenspektrum

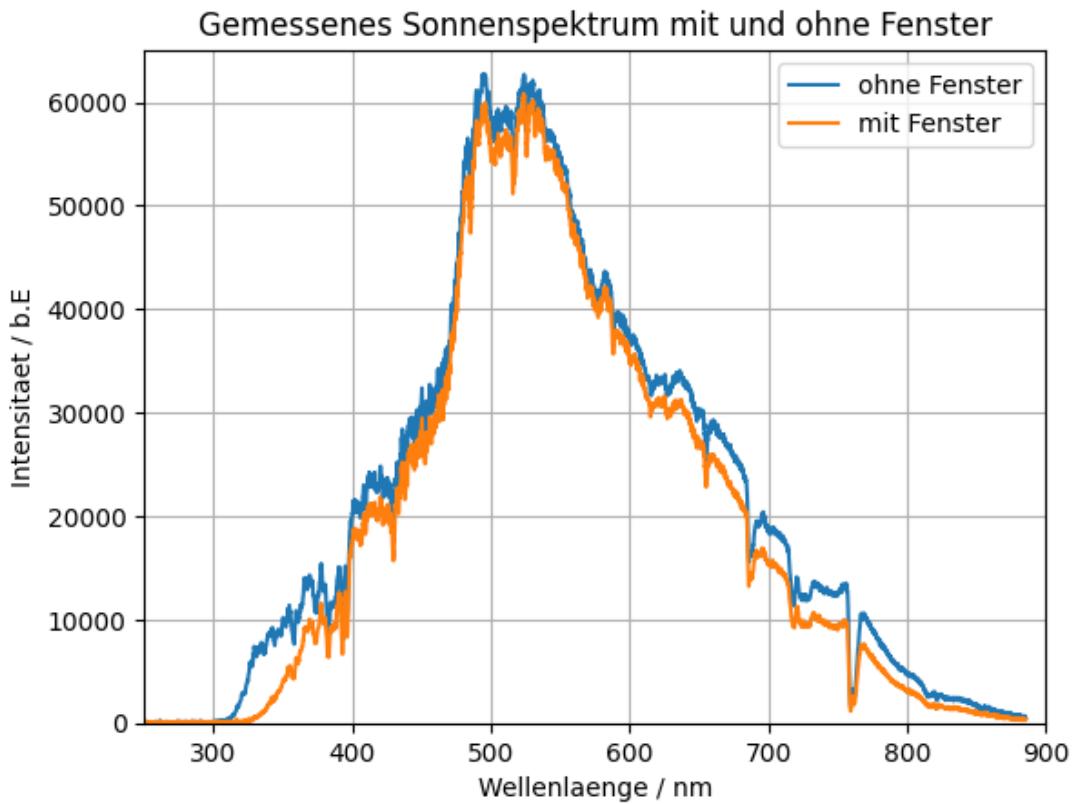
```
[38]: # matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Das Ersetzen der Kommas geht mit folgender Funktion:
def comma_to_float(valstr):
    return float(valstr.decode("utf-8").replace(',', '.'))

#Importieren Spektrum ohne Gas
lamb_og, inten_og=np.loadtxt('Sonnenlicht ohne Fenster.txt', skiprows=17,
                             converters= {0:comma_to_float, 1:comma_to_float}, comments='>', unpack=True)

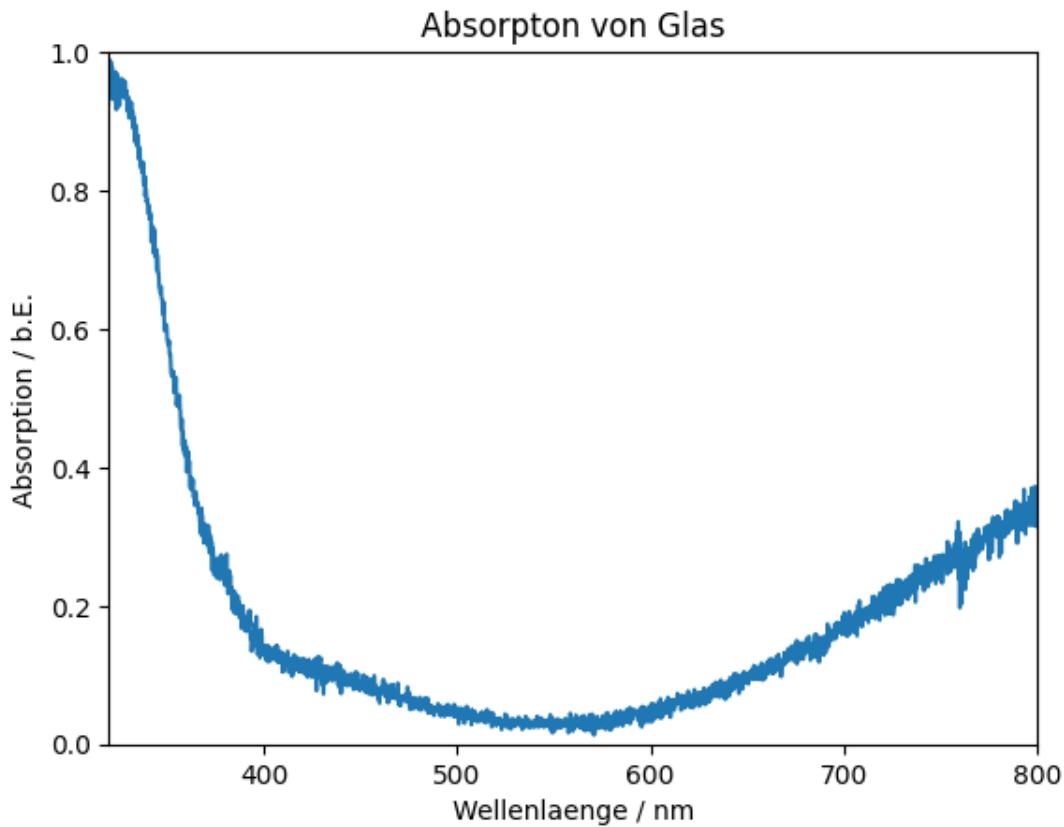
#Importieren Spektrum mit Glas
lamb_mg, inten_mg=np.loadtxt('Sonnenlicht mit Fenster.txt', skiprows=17,
                             converters= {0:comma_to_float, 1:comma_to_float}, comments='>', unpack=True)
```

```
[39]: #Beide Spektren mit und ohne Fensterglas in ein gemeinsames Diagramm tragen:
plt.plot(lamb_og, inten_og, label='ohne Fenster')
plt.plot(lamb_mg, inten_mg, label='mit Fenster')
plt.title('Gemessenes Sonnenspektrum mit und ohne Fenster')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E')
plt.legend()
plt.grid()
plt.ylim((0,65000))
plt.xlim((250,900))
plt.savefig("Himmel_m_o_G.pdf", format="pdf")
```



[40]: `#wie berechnen die Absorption von dem Glas, wobei Img die gemessene Intensität
mit und IoG ohne Fensterglas`

```
A=1-inten_mg/inten_og
plt.plot(lamb_mg, A)
plt.title('Absorptio von Glas')
plt.xlabel('Wellenlänge / nm')
plt.ylabel('Absorption / b.E.')
plt.ylim((0,1))
plt.xlim((320,800))
plt.savefig("Absorption_Glas.pdf", format="pdf")
```



Auswertung 2: Identifikation der Fraunhofer Linien

```
[41]: #wellenlaengen_fraunhoferlinien:
lambda_offen, inten_offen=np.loadtxt('Sonnenlicht ohne Fenster.txt',
                                     skiprows=17, converters={0:comma_to_float, 1:comma_to_float}, comments='>',
                                     unpack=True)
plt.axvline(x=760.9, ymin=0, ymax=65000, label='A', color='darkred')
plt.axvline(x=687.5, ymin=0, ymax=65000, label='B', color='red')
plt.axvline(x=656.5, ymin=0, ymax=65000, label='C', color='orange')
plt.axvline(x=589.75, ymin=0, ymax=65000, label='D-Linien', color='yellow')
plt.axvline(x=527.2, ymin=0, ymax=65000, label='E', color='green')
plt.axvline(x=517.3, ymin=0, ymax=65000, label='$b_1$', color='lime')
plt.axvline(x=486.1, ymin=0, ymax=65000, label='F', color='turquoise')
plt.axvline(x=430.7, ymin=0, ymax=65000, label='G', color='lightblue')
plt.axvline(x=397.2, ymin=0, ymax=65000, label='H', color='darkblue')
plt.axvline(x=393.7, ymin=0, ymax=65000, label='K', color='purple')
#graph plotten
plt.plot(lambda_offen, inten_offen, label='Spektrum', color='black')
plt.title('Frauenhofersche Linien')
plt.xlabel('Wellenlaenge [nm]')
```

```

plt.ylabel('Absorption [b.E.]')
plt.legend(fontsize=8)
plt.grid()
plt.ylim((0,65000))
plt.xlim(250,800)
plt.savefig('fraunhoferlinien.pdf', format='PDF')
plt.savefig('fraunhoferlinien.png', format='PNG')

from scipy.signal import find_peaks
# wellenlaengen fraunhoferlinien
fraunhoferlinien = [759.4, 686.7, 656.3, 589.6, 589.0, 587.6, 527.0, 518.4, 486.
↪1, 430.8, 396.8, 393.4]

# Dein Code für das Plotten der Linien und des Spektrums

# Suche Peaks in der Nähe der Fraunhoferlinien
toleranz = 5 # Toleranz um die bekannten Wellenlängen herum

peaks_near_fraunhofer = []

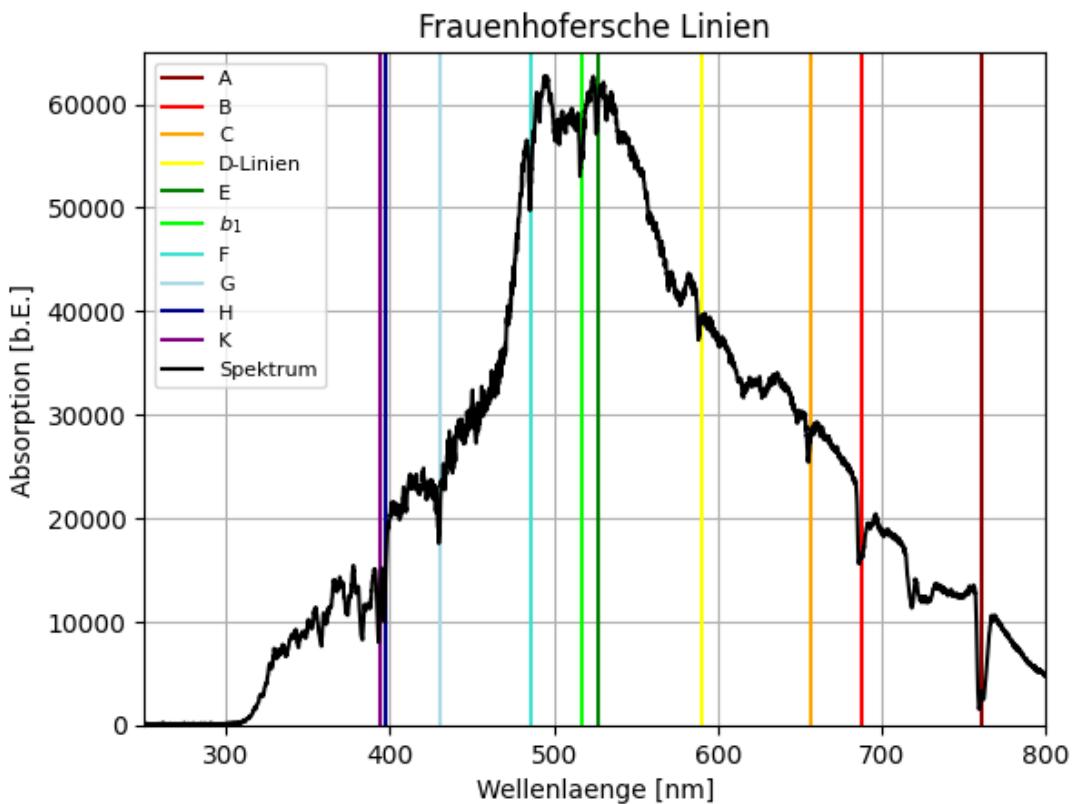
for wl in fraunhoferlinien:
    peaks, _ = find_peaks(inten_offen, height=5000, distance=10)
    nearby_peaks = [lambda_offen[idx] for idx in peaks if abs(lambda_offen[idx] ↪
↪- wl) <= toleranz]
    peaks_near_fraunhofer.extend(nearby_peaks)

# Entferne doppelte Werte und sortiere die Wellenlängen
peaks_near_fraunhofer = sorted(list(set(peaks_near_fraunhofer)))

# Zeige die gefundenen Peaks in der Nähe der Fraunhoferlinien an
print("Wellenlängen der Peaks in der Nähe der Fraunhoferlinien:", ↪
↪peaks_near_fraunhofer)

```

Wellenlängen der Peaks in der Nähe der Fraunhoferlinien: [388.754, 390.796, 395.081, 400.176, 427.784, 432.636, 481.224, 483.417, 485.609, 487.601, 489.591, 514.574, 518.917, 521.284, 524.043, 526.406, 528.374, 530.341, 585.303, 589.935, 591.863, 651.349, 653.234, 655.118, 657.376, 660.384, 683.576, 686.181, 689.154, 756.138]



Auswertung 3: Balmerserie

```
[42]: #wellenlaengen balmerserie
plt.axvline(x=656.55, ymin=0, ymax=65000, label=r'$H_{\alpha}$', color='red')
plt.axvline(x=486.4, ymin=0, ymax=65000, label=r'$H_{\beta}$', color='turquoise')
plt.axvline(x=434.1, ymin=0, ymax=65000, label=r'$H_{\gamma}$', color='blue')
plt.axvline(x=410.5, ymin=0, ymax=65000, label=r'$H_{\delta}$', color='purple')

#graph plotten
plt.plot(lambda_offen, inten_offen, label='Spektrum', color='black')
plt.title('Balmer Serie')
plt.xlabel('Wellenlaenge [nm]')
plt.ylabel('Absorption [b.E.]')
plt.legend(fontsize=10)
plt.grid()
plt.ylim(0,65000)
plt.xlim(320,800)
plt.savefig('balmerserie.pdf', format='PDF')
plt.savefig('balmerserie.png', format='PNG')
```

```

from scipy.signal import find_peaks

# die bekannten Wellenlängen der Frabelinien
wellenlaengen_frabelinien = [656.55, 486.4, 434.1, 410.5]

# Suche Peaks in der Nähe der bekannten Wellenlängen
toleranz = 5 # Toleranz um die bekannten Wellenlängen herum
peaks_near_lines = []

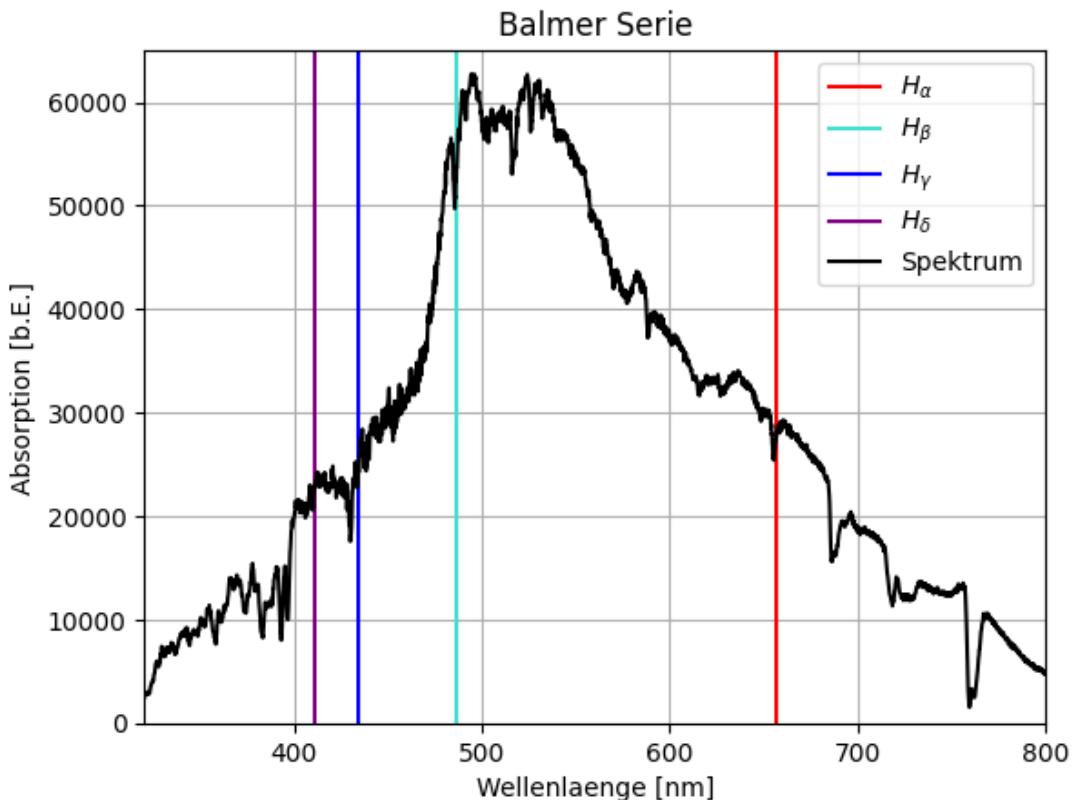
for wl in wellenlaengen_frabelinien:
    peaks, _ = find_peaks(inten_offen, height=5000, distance=10)
    nearby_peaks = [lambda_offen[idx] for idx in peaks if abs(lambda_offen[idx] - wl) <= toleranz]
    peaks_near_lines.extend(nearby_peaks)

# Entferne doppelte Werte und sortiere die Wellenlängen
peaks_near_lines = sorted(list(set(peaks_near_lines)))

# Zeige die gefundenen Peaks in der Nähe der Frabelinien an
print("Wellenlängen der Peaks in der Nähe der Frabelinien:", peaks_near_lines)

```

Wellenlängen der Peaks in der Nähe der Frabelinien: [408.315, 412.379, 432.636, 436.271, 438.491, 483.417, 485.609, 487.601, 489.591, 653.234, 655.118, 657.376, 660.384]



Auswertung 4: Vergleich verschiedener Lichtquellen

[43]: #Es sollte eigentlich während des Versuchs ein gemeinsames Diagramm erstellt werden, nähmlcih in SectraSuite dem letzten gemessenen Spektrum (z.B. dem LED-Spektrum) die anderen Spektren zu überlagern. Allerdings haben wir leider nur die Bild von Spekren, weshalb es nicht möglich ist. Trotzdem haben wir die Code bereit gestellt, um auch Möglichkeit zu bitten, mit txt.-Data zu bearbeiten.

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import scipy.integrate as integrate
from scipy.stats import chi2
from IPython.display import HTML, display
from scipy.special import gamma

def comma_to_float(valstr):
    return float(valstr.decode("utf-8").replace(',', '.'))
```

```

#spekturm gluelampe
lambda_glueh, inten_glueh=np.loadtxt('data/gluehlampe.txt', skiprows=17,
                                     converters={0:comma_to_float,1:comma_to_float}, comments='>', unpack=True)

#graph plotten
plt.plot(lambda_glueh, inten_glueh, label='Gluehlampe', color='turquoise')
plt.title('Gemessenes Gluehlampenspektrum')
plt.xlabel('Wellenlaenge [nm]')
plt.ylabel('Intensitaet [b.E.]')
plt.legend()
plt.grid()
plt.ylim((0,45000))
plt.xlim((250,900))
plt.savefig('balmerserie.png', format='PNG')

```

□

```

FileNotFoundError                         Traceback (most recent call last)
→last)

c:\Users\shiy0\OneDrive\Dokumente\PAP2.1\Versuch 234\Python
→234\Auswertung Python 234.ipynb Cell 10 line 1
    <a href='vscode-notebook-cell:/c%3A/Users/shiy0/OneDrive/Dokumente/PAP2.
    →1/Versuch%20234/Python%20234/Auswertung%20Python%20234.ipynb#X12sZmlsZQ%3D%3D?
    →line=13'>14</a>      return float(valstr.decode("utf-8").replace(',', '.')) 
    <a href='vscode-notebook-cell:/c%3A/Users/shiy0/OneDrive/Dokumente/PAP2.
    →1/Versuch%20234/Python%20234/Auswertung%20Python%20234.ipynb#X12sZmlsZQ%3D%3D?
    →line=15'>16</a> #spekturm gluelampe
    ---> <a href='vscode-notebook-cell:/c%3A/Users/shiy0/OneDrive/Dokumente/PAP2.
    →1/Versuch%20234/Python%20234/Auswertung%20Python%20234.ipynb#X12sZmlsZQ%3D%3D?
    →line=16'>17</a> lambda_glueh, inten_glueh=np.loadtxt('data/gluehlampe.txt',
    →skiprows=17, converters={0:comma_to_float,1:comma_to_float}, comments='>', 
    →unpack=True)
    <a href='vscode-notebook-cell:/c%3A/Users/shiy0/OneDrive/Dokumente/PAP2.
    →1/Versuch%20234/Python%20234/Auswertung%20Python%20234.ipynb#X12sZmlsZQ%3D%3D?
    →line=18'>19</a> #graph plotten
    <a href='vscode-notebook-cell:/c%3A/Users/shiy0/OneDrive/Dokumente/PAP2.
    →1/Versuch%20234/Python%20234/Auswertung%20Python%20234.ipynb#X12sZmlsZQ%3D%3D?
    →line=19'>20</a> plt.plot(lambda_glueh, inten_glueh, label='Gluehlampe',
    →color='turquoise')
```

```

File c:
→\Users\shiy0\AppData\Local\Programs\Python\Python312\Lib\site-packages\numpy\lib\npyio.py:1373, in loadtxt(fname, dtype, comments, delimiter, converters, skiprows, usecols, unpack, ndmin, encoding, max_rows, quotechar, like)
    1370 if isinstance(delimiter, bytes):
    1371     delimiter = delimiter.decode('latin1')
→ 1373 arr = _read(fname, dtype=dtype, comment=comment, delimiter=delimiter,
    1374             converters=converters, skiplines=skiprows, usecols=usecols,
    1375             unpack=unpack, ndmin=ndmin, encoding=encoding,
    1376             max_rows=max_rows, quote=quotechar)
    1378 return arr

File c:
→\Users\shiy0\AppData\Local\Programs\Python\Python312\Lib\site-packages\numpy\lib\npyio.py:992, in _read(fname, delimiter, comment, quote, imaginary_unit, usecols, skiplines, max_rows, converters, ndmin, unpack, dtype, encoding)
    990     fname = os.fspath(fname)
    991 if isinstance(fname, str):
--> 992     fh = np.lib._datasource.open(fname, 'rt', encoding=encoding)
    993     if encoding is None:
    994         encoding = getattr(fh, 'encoding', 'latin1')

File c:
→\Users\shiy0\AppData\Local\Programs\Python\Python312\Lib\site-packages\numpy\lib\_datasource.py:193, in open(path, mode, destpath, encoding, newline)
    156 """
    157 Open `path` with `mode` and return the file object.
    158 ...
    189
    190 """
    192 ds = DataSource(destpath)
--> 193 return ds.open(path, mode, encoding=encoding, newline=newline)

File c:
→\Users\shiy0\AppData\Local\Programs\Python\Python312\Lib\site-packages\numpy\lib\_datasource.py:533, in DataSource.open(self, path, mode, encoding, newline)
    530     return _file_openers[ext](found, mode=mode,
    531                             encoding=encoding, newline=newline)
    532 else:
--> 533     raise FileNotFoundError(f"{{path}} not found.")

```

```
FileNotFoundException: data/gluehlampe.txt not found.
```

```
[ ]: #spektrum Energiesparlampe
lambda_es, inten_es=np.loadtxt('energiespar.txt', skiprows=17, converters={0:
    >comma_to_float, 1:comma_to_float}, comments='>', unpack=True)

#graph plotten
plt.plot(lambda_es, inten_es, label='Gluehlampe', color='turquoise')
plt.title('Gemessenes Energiesparlampenspektrum')
plt.xlabel('Wellenlaenge [nm]')
plt.ylabel('Intensitaet [b.E.]')
plt.legend()
plt.grid()
plt.ylim((0,61000))
plt.xlim((250,900))
plt.savefig('figures/energiesparlampe.pdf', format='PDF')

#spektrum led gluehbirne
lambda_led, inten_led=np.loadtxt('LED1.txt', skiprows=17, converters={0:
    >comma_to_float, 1:comma_to_float}, comments='>', unpack=True)

#graph plotten
plt.plot(lambda_led, inten_led, label='LED Lampe', color='blue')
plt.title('Gemessenes Spektrum der LED Lampe')
plt.xlabel('Wellenlaenge [nm]')
plt.ylabel('Intensitaet [b.E.]')
plt.legend()
plt.grid()
plt.ylim((0,60000))
plt.xlim((250,900))
plt.savefig('figures/ledgluehbirne.pdf', format='PDF')

#alle lampen in einem graph
plt.plot(lambda_led, inten_led, label='LED Lampe', color='blue')
plt.plot(lambda_glueh, inten_glueh, label='Gluehbirne', color='turquoise')
plt.plot(lambda_es, inten_es, label='Energiesparlampe', color='purple')
plt.title('Gemessene Spektren der verschiedenen Lampen')
plt.xlabel('Wellenlaenge [nm]')
plt.ylabel('Intensitaet [b.E.]')
plt.legend()
plt.grid()
plt.ylim((0,68000))
plt.xlim((250,900))
plt.savefig('figures/allelampen.pdf', format='PDF')
```

```

#LED1
lambda_red_led, inten_red_led=np.loadtxt('laser1.txt', skiprows=17,
                                         converters={0:comma_to_float,1:comma_to_float},comments='>', unpack=True)

#graph plotten
plt.plot(lambda_red_led, inten_red_led, label='erste LED', color='red')
plt.title('Gemessenes Spektrum der ersten LED')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.legend()
plt.grid()
plt.ylim((0,60000))
plt.xlim((580,660))
plt.savefig('figures/LED_rot.pdf',format='PDF')

#LED2
lambda_gelb_led, inten_gelb_led=np.loadtxt('laser2.txt', skiprows=17,
                                             converters={0:comma_to_float,1:comma_to_float},comments='>', unpack=True)

#graph plotten
plt.plot(lambda_gelb_led, inten_gelb_led, label='zweite LED', color='yellow')
plt.title('Gemessenes Spektrum der zweite LED')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.legend()
plt.grid()
plt.ylim((0,60000))
plt.xlim((560,630))
plt.savefig('figures/LED2.pdf',format='PDF')

#LED3
lambda_orange_led, inten_orange_led=np.loadtxt('laser3.txt', skiprows=17,
                                                 converters={0:comma_to_float,1:comma_to_float},comments='>', unpack=True)

#graph plotten
plt.plot(lambda_orange_led, inten_orange_led, label='dritte LED', color='orange')
plt.title('Gemessenes Spektrum der dritten LED')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.legend()
plt.grid()
plt.ylim((0,60000))
plt.xlim((560,630))
plt.savefig('figures/LED3.pdf',format='PDF')

#LED4

```

```

lambda_blaue_led, inten_blaue_led=np.loadtxt('laser5.txt', skiprows=17, u
˓→converters={0:comma_to_float,1:comma_to_float},comments='>', unpack=True)

#graph plotten
plt.plot(lambda_blaue_led, inten_blaue_led, label='vierte LED', color='blue')
plt.title('Gemessenes Spektrum der vierten LED')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.legend()
plt.grid()
plt.ylim((0,60000))
plt.xlim((400,550))
plt.savefig('figures/LED5.pdf',format='PDF')

#LED6
lambda_grau_led, inten_grau_led=np.loadtxt('laser6.txt', skiprows=17, u
˓→converters={0:comma_to_float,1:comma_to_float},comments='>', unpack=True)

#graph plotten
plt.plot(lambda_grau_led, inten_grau_led, label='sechste LED', color='gray')
plt.title('Gemessenes Spektrum der sechsten LED')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.legend()
plt.grid()
plt.ylim((0,60000))
plt.xlim((400,680))
plt.savefig('figures/LED6.pdf',format='PDF')

#LED5
lambda_weiss_led, inten_weiss_led=np.loadtxt('laser5_2.txt', skiprows=17, u
˓→converters={0:comma_to_float,1:comma_to_float},comments='>', unpack=True)

#graph plotten
plt.plot(lambda_weiss_led, inten_weiss_led, label='fünfte LED', color='lightblue')
plt.title('Gemessenes Spektrum der fünfte LED')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.legend()
plt.grid()
plt.ylim((0,60000))
plt.xlim((400,680))
plt.savefig('figures/LED5.pdf',format='PDF')

#alle led zusammen

```

```

plt.plot(lambda_weiss_led, inten_weiss_led, label='weisse LED', color='lightblue')
plt.plot(lambda_ww_led, inten_ww_led, label='warmweisse LED', color='lightgray')
plt.plot(lambda_gelb_led, inten_gelb_led, label='gelbe LED', color='yellow')
plt.plot(lambda_blau_led, inten_blau_led, label='blau LED', color='blue')
plt.plot(lambda_orange_led, inten_orange_led, label='orange LED', color='orange')
plt.plot(lambda_red_led, inten_red_led, label='rote LED', color='red')

plt.title('Gemessenes Spektrum der verschiedenen LED')
plt.xlabel('Wellenlaenge [nm]')
plt.ylabel('Intensitaet [b.E]')
plt.legend()
plt.grid()
plt.ylim(0,65000)
plt.xlim(350,850)
plt.savefig('figures/LED_zusammen.pdf',format='PDF')

```

Auswertung 5: Auswertung des Natriumspektrums schwach

```

[97]: from scipy.signal import find_peaks
from scipy.signal import savgol_filter

lamb_og, inten_og = np.loadtxt('Na Schwachen Linien 400-540 nm.txt', skiprows=17, converters={0: comma_to_float, 1: comma_to_float}, comments='>', unpack=True)

# Filtern der Daten für Wellenlängen bis zu 540 nm
lamb_filtered = []
inten_filtered = []
for i in range(len(lamb_og)):
    if lamb_og[i] <= 580:
        lamb_filtered.append(lamb_og[i])
        inten_filtered.append(inten_og[i])

# Anwenden eines Savitzky-Golay-Filters, um das Signal zu glätten
window_length = 15 # Fenstergröße für die Glättung
poly_order = 2 # Ordnung des Polynoms für die Glättung
inten_smoothed = savgol_filter(inten_filtered, window_length, poly_order)

# Finden von Peaks in den geglätteten Daten mit einer relativen Prominenz
prominence_threshold = 400 # Ändere den Schwellenwert nach Bedarf
peaks, _ = find_peaks(inten_smoothed, distance=20, prominence=prominence_threshold) # Nur Peaks über der relativen Prominenz

# Ausgabe der Wellenlängenwerte der identifizierten Peaks
peaks_wavelengths = np.array(lamb_filtered)[peaks]

```

```

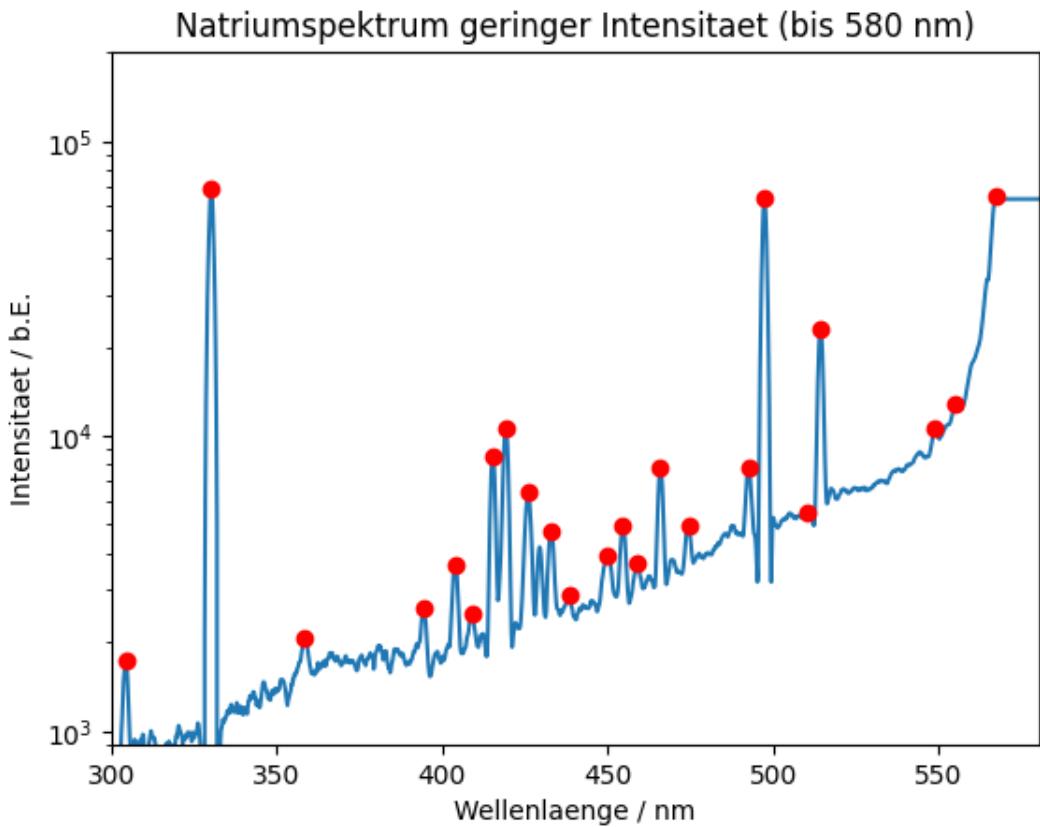
print("Positionen der Peaks (Wellenlängen in nm):")
for peak_wavelength in peaks_wavelengths:
    print(f"{peak_wavelength:.2f}")

# Plot der gefilterten und geglätteten Daten mit markierten signifikanten Peaks
plt.plot(lamb_filtered, inten_smoothed)
plt.plot(peaks_wavelengths, np.array(inten_smoothed)[peaks], "ro") # Markiere die Peaks in Rot
plt.title('Natriumspektrum geringer Intensität (bis 580 nm)')
plt.xlabel('Wellenlänge / nm')
plt.ylabel('Intensität / b.E.')
plt.yscale('log')
plt.ylim((900, 200000))
plt.xlim((300, 580)) # Begrenzung bis 540 nm
plt.show()

```

Positionen der Peaks (Wellenlängen in nm):

195.46
 241.28
 248.00
 284.03
 304.45
 330.37
 358.42
 394.47
 404.04
 409.13
 415.22
 419.28
 425.96
 432.84
 438.29
 449.77
 454.60
 458.81
 465.83
 474.44
 492.57
 497.34
 510.23
 514.18
 548.77
 555.02
 567.09



Auswertung 5: NaD

```
[75]: lamb_og, inten_og = np.loadtxt('NaD.txt', skiprows=17, converters={0: lambda x: float(x), 1: comma_to_float}, comments='>', unpack=True)

# Filtern der Daten für Wellenlängen bis zu 540 nm
lamb_filtered = []
inten_filtered = []
for i in range(len(lamb_og)):
    if lamb_og[i] >= 540:
        lamb_filtered.append(lamb_og[i])
        inten_filtered.append(inten_og[i])

# Anwenden eines Savitzky-Golay-Filters, um das Signal zu glätten
window_length = 15 # Fenstergröße für die Glättung
poly_order = 2 # Ordnung des Polynoms für die Glättung
inten_smoothed = savgol_filter(inten_filtered, window_length, poly_order)

# Finden von Peaks in den geglätteten Daten mit einer relativen Prominenz
prominence_threshold = 200 # Ändere den Schwellenwert nach Bedarf
```

```

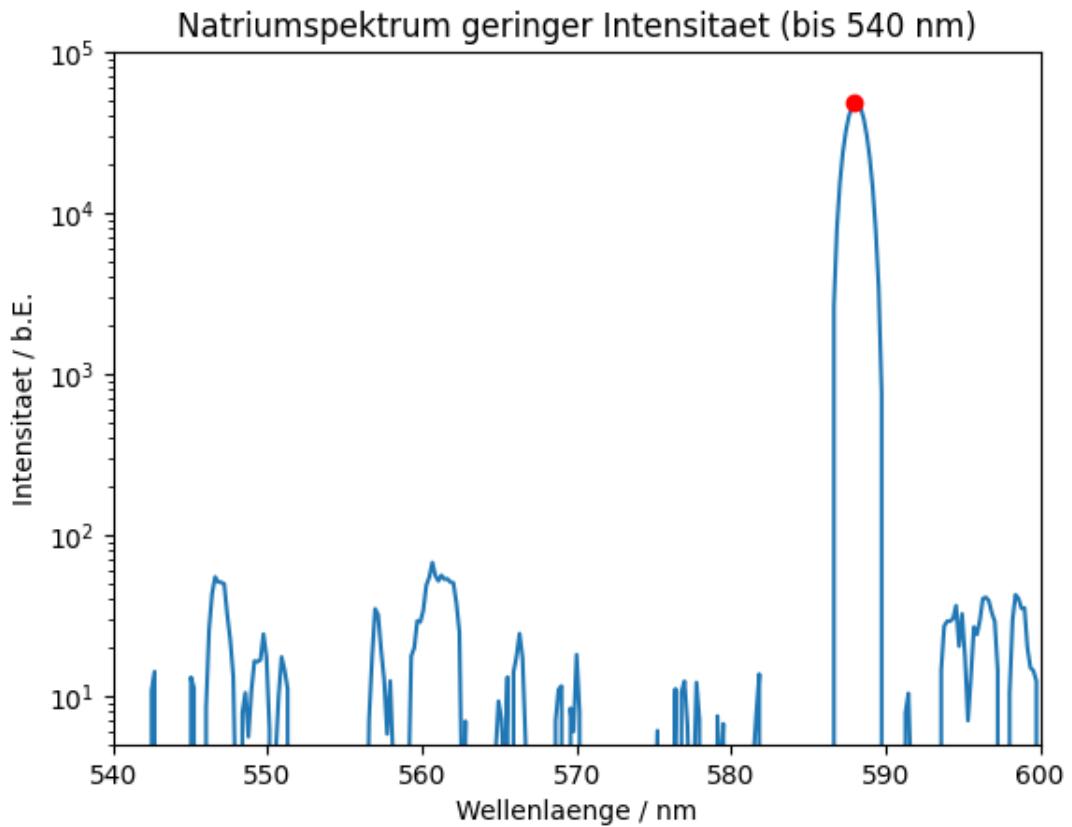
peaks, _ = find_peaks(inten_smoothed, distance=20, ↴
                     prominence=prominence_threshold) # Nur Peaks über der relativen Prominenz

# Ausgabe der Wellenlängenwerte der identifizierten Peaks
peaks_wavelengths = np.array(lamb_filtered)[peaks]
print("Positionen der Peaks (Wellenlängen in nm):")
for peak_wavelength in peaks_wavelengths:
    print(f"{peak_wavelength:.2f}")

# Plot der gefilterten und geglätteten Daten mit markierten signifikanten Peaks
plt.plot(lamb_filtered, inten_smoothed)
plt.plot(peaks_wavelengths, np.array(inten_smoothed)[peaks], "ro") # Markiere ↴ Peaks in Rot
plt.title('Natriumspektrum geringer Intensität (bis 540 nm)')
plt.xlabel('Wellenlänge / nm')
plt.ylabel('Intensität / b.E.')
plt.yscale('log')
plt.ylim(5, 100000)
plt.xlim((540, 600)) # Begrenzung bis 540 nm
plt.show()

```

Positionen der Peaks (Wellenlängen in nm):
588.01



```
[69]: lamb_og, inten_og = np.loadtxt('greater 650 nm.txt', skiprows=17, converters={0:
    ↪ comma_to_float, 1: comma_to_float}, comments='>', unpack=True)

# Filtern der Daten für Wellenlängen bis zu 540 nm
lamb_filtered = []
inten_filtered = []
for i in range(len(lamb_og)):
    if lamb_og[i] >= 540:
        lamb_filtered.append(lamb_og[i])
        inten_filtered.append(inten_og[i])

# Anwenden eines Savitzky-Golay-Filters, um das Signal zu glätten
window_length = 15 # Fenstergröße für die Glättung
poly_order = 2 # Ordnung des Polynoms für die Glättung
inten_smoothed = savgol_filter(inten_filtered, window_length, poly_order)

# Finden von Peaks in den geglätteten Daten mit einer relativen Prominenz
prominence_threshold = 200 # Ändere den Schwellenwert nach Bedarf
peaks, _ = find_peaks(inten_smoothed, distance=20, ↪
    ↪ prominence=prominence_threshold) # Nur Peaks über der relativen Prominenz
```

```

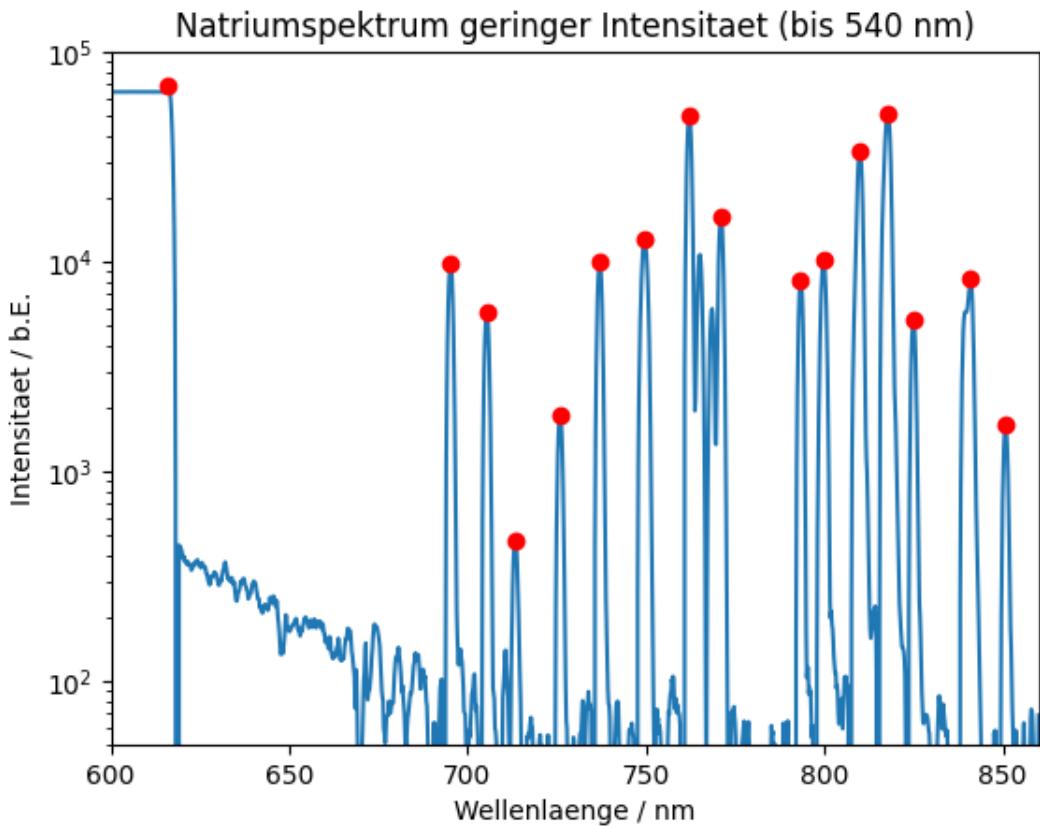
# Ausgabe der Wellenlängenwerte der identifizierten Peaks
peaks_wavelengths = np.array(lamb_filtered)[peaks]
print("Positionen der Peaks (Wellenlängen in nm):")
for peak_wavelength in peaks_wavelengths:
    print(f"{peak_wavelength:.2f}")

# Plot der gefilterten und geglätteten Daten mit markierten signifikanten Peaks
plt.plot(lamb_filtered, inten_smoothed)
plt.plot(peaks_wavelengths, np.array(inten_smoothed)[peaks], "ro") # Markiere die Peaks in Rot
plt.title('Natriumspektrum geringer Intensität (bis 540 nm)')
plt.xlabel('Wellenlänge / nm')
plt.ylabel('Intensität / b.E.')
plt.yscale('log')
plt.ylim((50, 100000))
plt.xlim((600, 860)) # Begrenzung bis 540 nm
plt.show()

```

Positionen der Peaks (Wellenlängen in nm):

567.48
 582.02
 587.23
 616.04
 695.27
 705.44
 713.36
 726.01
 736.95
 749.46
 762.08
 771.04
 793.29
 799.65
 809.85
 817.73
 825.05
 840.99
 850.79



Auswertung 6: Zuordnung der gefundenen Linien

```
[95]: #Zuordnung der gefundene Linien
hc=1.2398*10**3
E_ry=-13.605
#1.Nebenserie
# Bestimmung von E_3p
E_3p=(-13.605/3**2)-(1.2398*10**3/819)
d_E_3p=(1.2398*10**3/((819)**2))*2.3
print(E_3p)
print(d_E_3p)

def berechne_werte(m):
    l_theo_1 = 1.2398 * 1000 / (-13.605 / m**2 - E_3p)
    d_l_theo_1 = (1.2398 * 1000 / (-13.605 / m**2 - E_3p)**2) * d_E_3p
    return l_theo_1, d_l_theo_1

# Nutzereingabe für m
m_wert = float(input("Gib den Wert für m ein: "))
```

```

# Berechnung der Werte basierend auf der Nutzereingabe von m
ergebnis_l_theo_1, ergebnis_d_l_theo_1 = berechne_werte(m_wert)

# Ausgabe der berechneten Werte
print(f"l_theo_1: {ergebnis_l_theo_1}")
print(f"d_l_theo_1: {ergebnis_d_l_theo_1}")

```

-3.0254639804639805
0.004251201247538243
l_theo_1: 420.9902918194865
d_l_theo_1: 0.6077209309342518

Auswertung 7: 2. Nebenserie

```

[108]: #2.Nebenserie
E_3s    = E_3p-(hc/589)
d_E_3s = np.sqrt((d_E_3p)**2+((hc/589**2)*1.8)**2)

print(E_3s)
print(d_E_3s)

#Korrekturfaktor
s=3-np.sqrt(E_ry/E_3s)
d_s=0.5*np.sqrt(E_ry/E_3s**3)*d_E_3s

print(s)
print(d_s)

def berechne_werte(m):
    l_theo_1 = 1.2398E3/(-13.605/(m-s)**2-E_3p)
    d_l_theo_1 = np.sqrt(((1.2398E3/(-13.605/
    ↪(m-s)**2-E_3p)**2)*d_E_3p)**2+(2*E_ry*hc*d_s/(((1.2398E3/(-13.605/
    ↪(m-s)**2-E_3p)**2)*(m-s)**3))**2)
    return l_theo_1, d_l_theo_1

# Nutzereingabe für m
m_wert = float(input("Gib den Wert für m ein: "))

# Berechnung der Werte basierend auf der Nutzereingabe von m
ergebnis_l_theo_1, ergebnis_d_l_theo_1 = berechne_werte(m_wert)

# Ausgabe der berechneten Werte
print(f"l_theo_1: {ergebnis_l_theo_1}")
print(f"d_l_theo_1: {ergebnis_d_l_theo_1}")

```

-5.130387579784863
0.007710537525208829

```

1.3715509119579206
0.0012237104513034744
l_theo_1: 444.10630868657296
d_l_theo_1: 0.6763967373804647

```

Auswertung 8: Hauptserie

```
[8]: import numpy as np

E_ry=-13.605
E_3p=-3.0254639804639805
d_E_3p=0.004251201247538243
E_3s= -5.130387579784863
d_E_3s=0.007710537525208829
hc=1.2398*10**3

#Hauptserie
#Korrekturterm p
p=3-np.sqrt(E_ry/E_3p)
d_p=0.5*np.sqrt(E_ry/E_3p**3)*d_E_3p
print(p)
print(d_p)

def berechne_werte(m):
    l_theo_3=1.2398E3/(-13.605/(m-p)**2-E_3s)
    d_l_theo_3=np.sqrt(((1.2398E3/(-13.605/
    -(m-p)**2-E_3s)**2)*d_E_3s)**2+(2*E_ry*hc*d_p/(((1.2398E3/(-13.605/
    -(m-p)**2-E_3s)**2)*(m-p)**3))**2)
    return l_theo_3, d_l_theo_3

# Nutzereingabe für m
m_wert = float(input("Gib den Wert für m ein: "))

# Berechnung der Werte basierend auf der Nutzereingabe von m
ergebnis_l_theo_3, ergebnis_d_l_theo_3 = berechne_werte(m_wert)

# Ausgabe der berechneten Werte
print(f"l_theo_3: {ergebnis_l_theo_3}")
print(f"d_l_theo_3: {ergebnis_d_l_theo_3}")
```

```

0.8794267453437401
0.0014898514283928777
l_theo_3: 286.3868430267978
d_l_theo_3: 0.5115192199857751

```

Auswertung 9: Bestimmung der Serienenergien und der l-abhängigen Korrekturfaktor

```
[13]: import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

def fit_func(m, E_Ry, E_3p, D_d):
    return 1.2398E3 / (E_Ry / (m - D_d)**2 - E_3p)

# Messwerte und Fehler
wellenl = np.array([817.7, 567.1, 497.3, 465.8, 449.8, 438.3, 432.8, 426.0, 419.
    ↪3])
fehler = np.array([1.2, 1.5, 1.7, 1.1, 1.2, 1.3, 1.0, 0.8, 1.5]) # Annahme: ↪
    ↪Fehlerwerte für jede Wellenlänge

quantenz = np.arange(3, 12) # Geändert auf 12, um 9 Wellenlängenwerte zu haben

# Curve Fitting
para = [-13.6, -3, -0.02]
popt, pcov = curve_fit(fit_func, quantenz, wellenl, sigma=fehler, p0=para)

# Extrahierte Standardabweichungen aus der Kovarianzmatrix (Wurzel der ↪
    ↪Diagonalelemente)
perr = np.sqrt(np.diag(pcov))

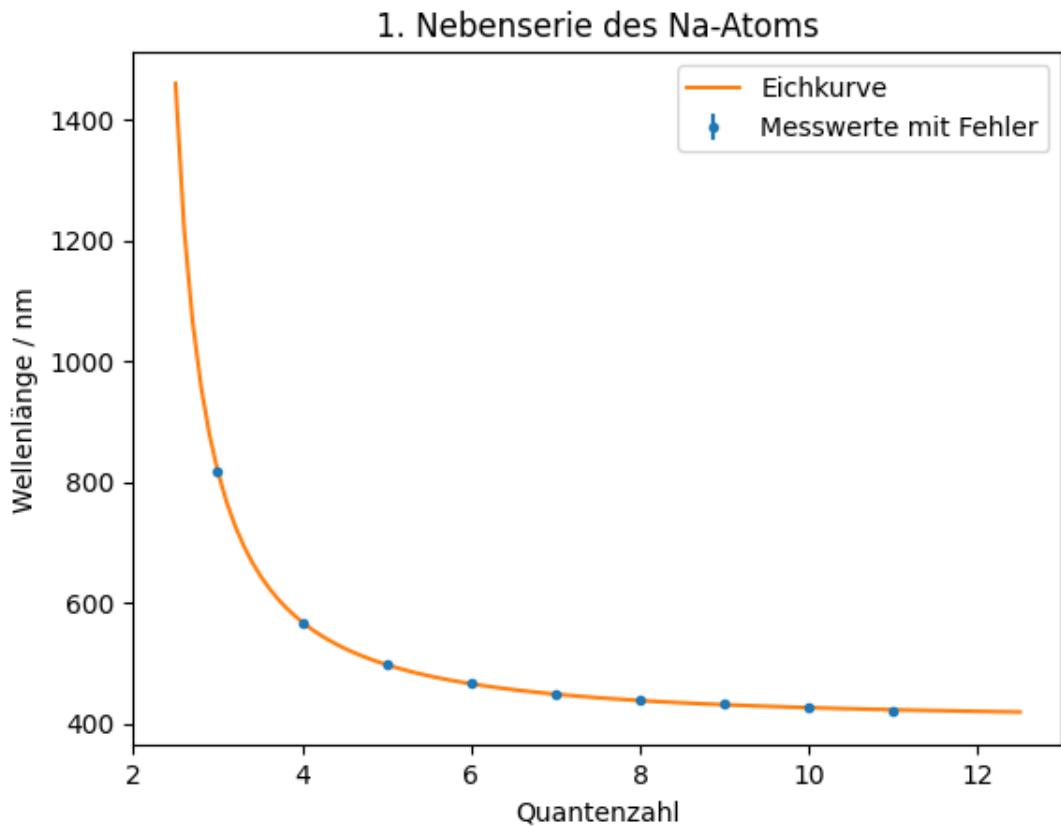
# Eichkurve
x_fit = np.linspace(2.5, 12.5, 100) # Für die Eichkurve
y_fit = fit_func(x_fit, *popt)

# Plotten der Daten und Eichkurve
plt.errorbar(quantenz, wellenl, fehler, fmt=". ", label='Messwerte mit Fehler')
plt.plot(x_fit, y_fit, label='Eichkurve')
plt.xlabel('Quantenzahl')
plt.ylabel('Wellenlänge / nm')
plt.title('1. Nebenserie des Na-Atoms')
plt.legend()

# Fehler für die Parameter ausgeben
print(f"E_Ry = {popt[0]} ± {perr[0]}")
print(f"E_3p = {popt[1]} ± {perr[1]}")
print(f"D_d = {popt[2]} ± {perr[2]}")

plt.show()
```

E_Ry = -13.991502532957456 ± 0.6172136676380785
E_3p = -3.0471626785398054 ± 0.008154668528653415
D_d = -0.023242179072388354 ± 0.06001752725675183



Auswertung 10: Chi-Summe sowie Güte des Fits

```
[14]: chi2_=np.sum((fit_func(quantenz,*popt)-wellenl)**2/fehler**2)
dof=len(quantenz)-3 #dof: degrees of freedom, Freiheitsgrad
chi2_red=chi2_/dof
print("chi2=", chi2_)
print("chi2_red=",chi2_red)
```

```
chi2= 9.725241961129985
chi2_red= 1.620873660188331
```

Auswertung 11: Fitwahrscheinlichkeit

```
[16]: from scipy.stats import chi2
prob=round(1-chi2.cdf(chi2_,dof),2)*100
print("Wahrscheinlichkeit:", prob, "%")

plt.errorbar(quantenz,wellenl,fehler, fmt=".")
plt.xlabel('Quantenzahl')
plt.ylabel('Wellenlaenge / nm')
plt.title('1. Nebenserie des Na-Atoms')
```

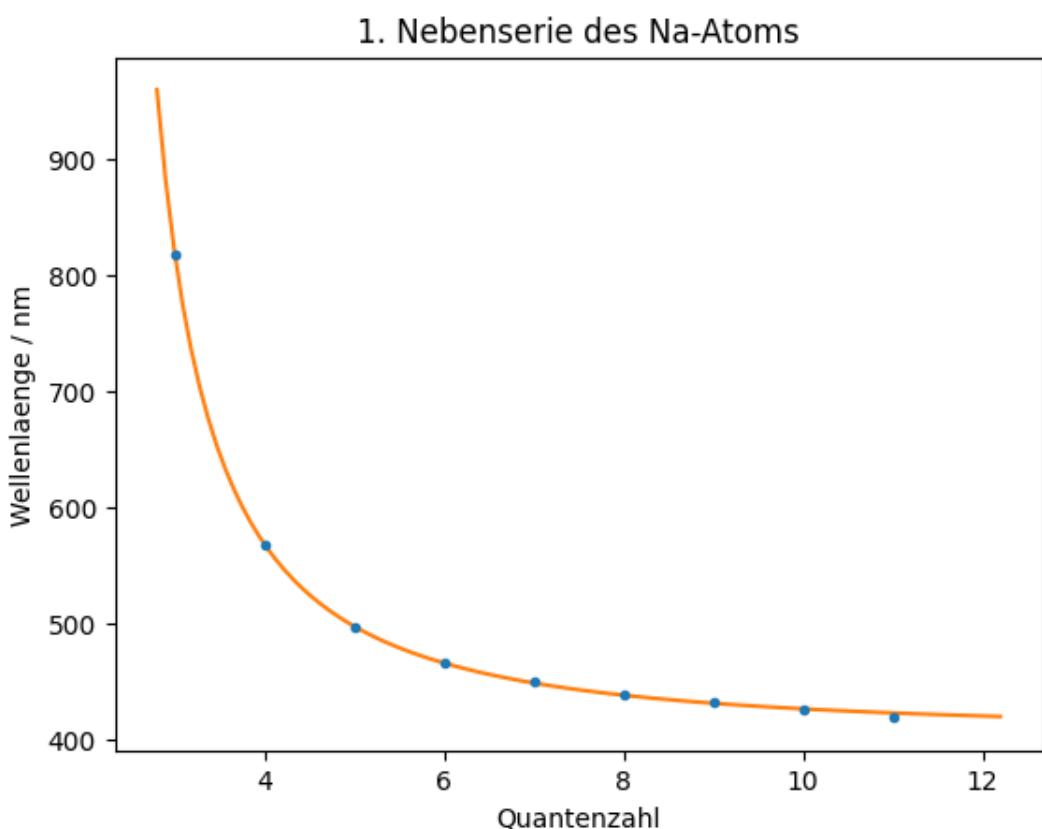
```

x=np.linspace(2.8,12.2, 100)
plt.plot(x, fit_func(x,*popt))

# Fehler für die Parameter ausgeben
print(f"E_Ry = {popt[0]} ± {perr[0]}")
print(f"E_3p = {popt[1]} ± {perr[1]}")
print(f"D_d = {popt[2]} ± {perr[2]}")

```

Wahrscheinlichkeit: 14.000000000000002 %
 E_Ry = -13.991502532957456 ± 0.6172136676380785
 E_3p = -3.0471626785398054 ± 0.008154668528653415
 D_d = -0.023242179072388354 ± 0.06001752725675183



Auswertung 12: Wiederholung für die 2. Nebenserie

```

[17]: import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

def fit_func(m, E_Ry, E_3p, D_s):

```

```

    return 1.2398E3 / (E_Ry / (m - D_s)**2 - E_3p)

# Messwerte und Fehler
wellenl = np.array([616.1, 514.2, 474.4, 454.6, 438.3])
fehler = np.array([1.6, 1.4, 1.2, 0.8, 1.3]) # Annahme: Fehlerwerte für jede
    ↪Wellenlänge

quantenz = np.arange(4, 9) # Geändert auf 9, um 5 Wellenlängenwerte zu haben

# Curve Fitting
para = [-13.6, -3, -0.02]
popt, pcov = curve_fit(fit_func, quantenz, wellenl, sigma=fehler, p0=para)

# Extrahierte Standardabweichungen aus der Kovarianzmatrix (Wurzel der
    ↪Diagonalelemente)
perr = np.sqrt(np.diag(pcov))

# Eichkurve
x_fit = np.linspace(2.5, 12.5, 100) # Für die Eichkurve
y_fit = fit_func(x_fit, *popt)

# Plotten der Daten und Eichkurve
plt.errorbar(quantenz, wellenl, fehler, fmt=". ", label='Messwerte mit Fehler')
plt.plot(x_fit, y_fit, label='Eichkurve')
plt.xlabel('Quantenzahl')
plt.ylabel('Wellenlänge / nm')
plt.title('2. Nebenserie des Na-Atoms')
plt.legend()

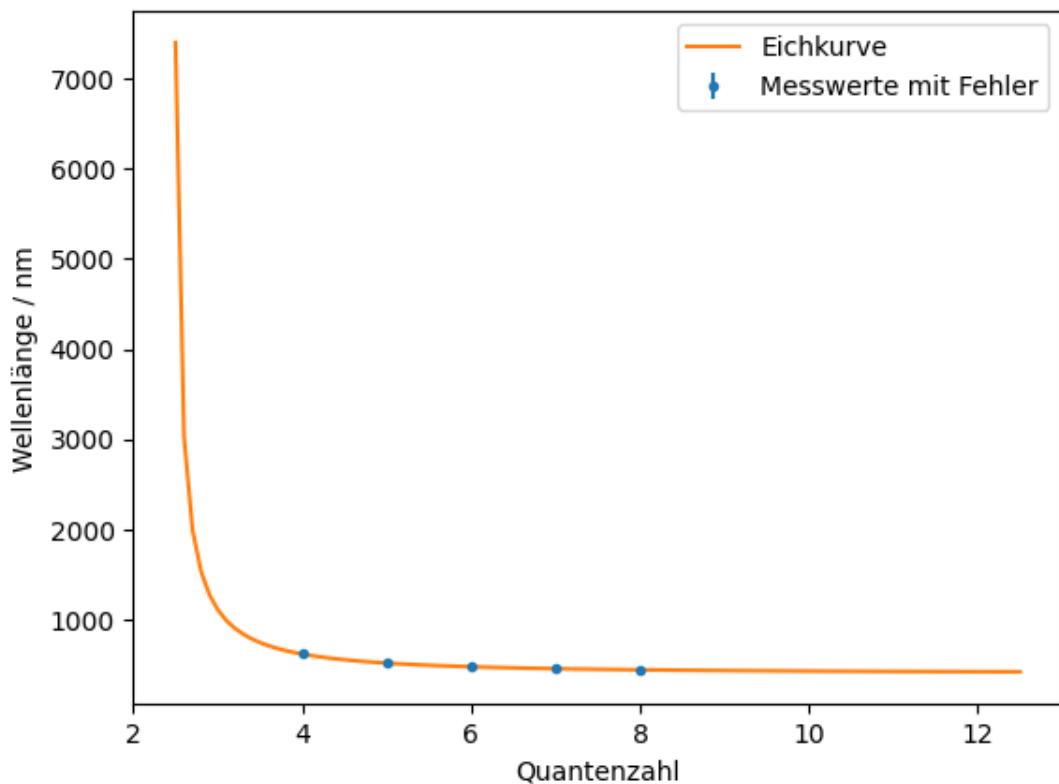
# Fehler für die Parameter ausgeben
print(f"E_Ry = {popt[0]} ± {perr[0]}")
print(f"E_3p = {popt[1]} ± {perr[1]}")
print(f"D_s = {popt[2]} ± {perr[2]}")

plt.show()

```

$E_{Ry} = -14.792991172142969 \pm 3.194538793354411$
 $E_{3p} = -3.0583729131715685 \pm 0.04010344624717103$
 $D_s = 0.237878186931764 \pm 0.3412310359707126$

2. Nebenserie des Na-Atoms



```
[18]: chi2_=np.sum((fit_func(quantenz,*popt)-wellenl)**2/fehler**2)
dof=len(quantenz)-3 #dof: degrees of freedom, Freiheitsgrad
chi2_red=chi2_/dof
print("chi2=", chi2_)
print("chi2_red=",chi2_red)

from scipy.stats import chi2
prob=round(1-chi2.cdf(chi2_,dof),2)*100
print("Wahrscheinlichkeit:", prob, "%")

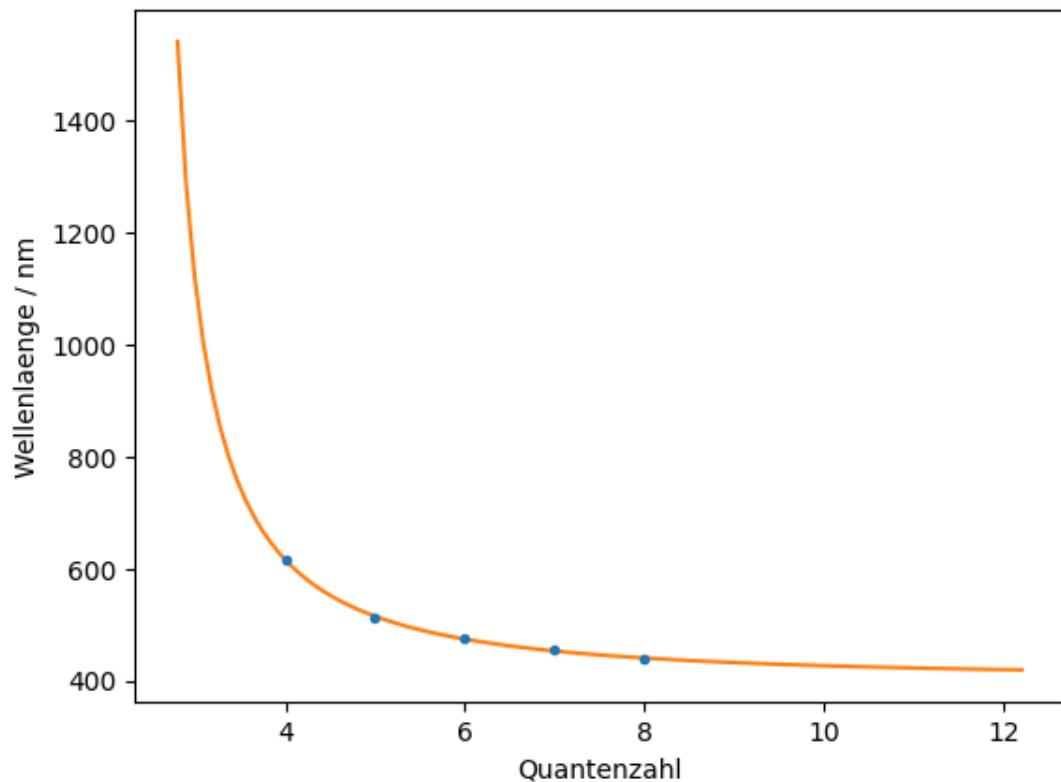
plt.errorbar(quantenz,wellenl,fehler, fmt=".")
plt.xlabel('Quantenzahl')
plt.ylabel('Wellenlaenge / nm')
plt.title('2. Nebenserie des Na-Atoms')
x=np.linspace(2.8,12.2, 100)
plt.plot(x, fit_func(x,*popt))

# Fehler für die Parameter ausgeben
print(f"E_Ry = {popt[0]} ± {perr[0]}")
print(f"E_3p = {popt[1]} ± {perr[1]}")
```

```
print(f"D_d = {popt[2]} ± {perr[2]}")
```

```
chi2= 6.734394970419471
chi2_red= 3.3671974852097355
Wahrscheinlichkeit: 3.0 %
E_Ry = -14.792991172142969 ± 3.194538793354411
E_3p = -3.0583729131715685 ± 0.04010344624717103
D_d = 0.237878186931764 ± 0.3412310359707126
```

2. Nebenserie des Na-Atoms



```
[ ]:
```

6 Quelle

- Wagner, J. (April 2022). Physikalisches Praktikum PAP 2.1 für Studierende der Physik [Praktikumsanleitung]. Ruprecht-Karls-Universität Heidelberg. Abgerufen am 29. Oktober 2023, von https://www.physi.uni-heidelberg.de/Einrichtungen/AP/info/Corona /PAP2_1_2023.pdf