Versuch 234 Lichtquellen

Frauendorfer Andreas

13.06.2022

1 Einleitung

1.1 Modell des schwarzen Strahlers

Ein schwarzer Strahler absorbiert die gesamte auf ihn einfallende Strahlung. Er kann auch wieder die gesamte Strahlung abgeben. Dabei ist die Intensität folgendermaßen verteilt:

$$M(,T) \tag{1}$$

mit der Strahlungsleistung M$proFlächenihaltdAimWellenlängenbereich\lambda$ bis $\lambda + d\lambda$

1.2 Wiensches Verschiebungsgesetz

Bei reinen Temperaturstrahlern tritt das Intensitätsmaximum bei kleineren Wellenlängen auf.

$$\lambda_{max} = \frac{2897,8m \cdot K}{T} \tag{2}$$

So lassen sich Lichtquellen anhand des Lichts charakterisieren, dass sie emittieren. Je höher die Temperatur des Strahlers ist, desto kleiner Wellenlängen werden emittiert. Auch sind ab ca 1000K sichtbare Wellenlängen involviert. So sieht man kein kleineren Temperaturen vermehrt Licht und bei ca 5888K mischen sich alle sichtbaren Farben so, dass die Lichtquelle weiß erscheint. Für höhere Temperaturen erscheint die Lichtquelle dann violetter.

1.3 Streuung

Der Wirkungsquerschnitt bei Sonnenlichtsstreuung ist in der vierten Potenz von der Frequenz abhängig, deshalb wird blaues Licht 16-mal stärker als rotes Licht gestreut und ist deshalb besser am Himmel zu sehen.
Wasserdampf, Sauerstoff und Kohlendioxied streuen das Sonnenlicht sehr stark und sind deshalb die Ursachen für einige der Fraunhofschen Linien.

1.4 Nichttemperaturstrahler

Es können auch Strahler durch Anregung von Elektronen in abstrahlen, statt durch Wärmestrahlung. Diese Strahler wie zum Beispiel Laser oder Leuchtstoffröhren haben ein diskretes Spektrum. Quecksilber, das hauptsächlich im UV-Bereich strahlt, regt hierbei einen fluoreszierenden Stoff an, der im sichtbaren Bereich strahlt.
LED (Light Emitting Diode) funktionieren via Rekombination in einem pn-Übergang. Sie haben einen hohen Wirkungsgrad.
Natrium besitzt ein Außenelektron und kann deshalb für große Radien als Wasserstoff ähnlich angesehen werden.
. Der Grundzustand des Valenzelektrons ist das 3s Orbital. Die bekannteste Linie ist das gelbe Dublett bie 589nm

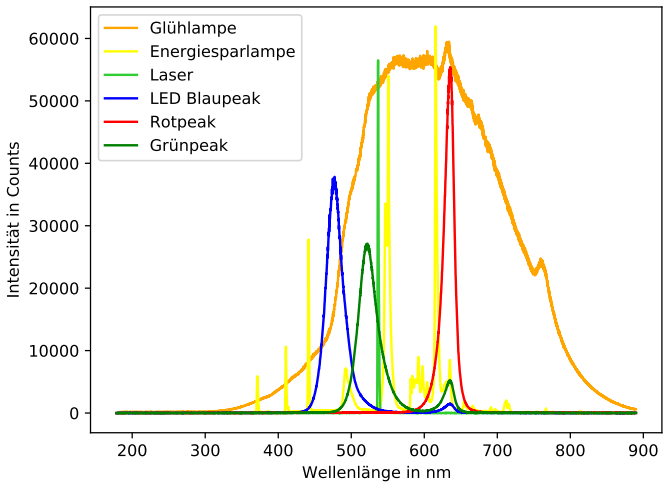
Messprotokoll 234

Unterschrift Tutor: gez von tutor:nikolai bolik

2 Auswertung

2.1 Auswertung der unterschiedlichen Lichtquellen

Zuerst plotten wir die Spektren der verschiedenen Lampen in einem Diagramm und deren Peaks zu vergleichen.



Die unterschiedlichen Lampen, sind gut anhand der unterschiedlichen Peaks zu unterschieden (siehe Diagramm 1). Die mittlere Wellenlänge jeder Lampe wird aus dem Diagramm abgelesen, wobei es sich bei Quellen mit auf eine Wellenlänge konzentrierten Emissionsspektrum direkt um dieses Wellenlänge handelt.
Die Hallogenlampe war leider im Versuchsaufbau nicht vorhanden und wurde deshalb nicht mit betrachtet.

$$\lambda_{Glüh} \approx 610nm \tag{3}$$
$$\lambda_{Energie} \approx 560nm \tag{4}$$
$$\lambda_{Laser} \approx 523nm \tag{5}$$
$$\lambda_{LEDb} \approx 490nm \tag{6}$$
$$\lambda_{LEDr} \approx 625nm \tag{7}$$
$$\lambda_{LEDg} \approx 510nm \tag{8}$$

Auch daran lassen sich die Quellen unterscheiden.

2.2 Auswertung des Sonnenspektrums

Hier tragen wir die Spektren in einem Diagramm ein, die unsere Kamera hinter einer Glasscheibe, ohne Scheibe und direkt in die Sonne aufnimmt.

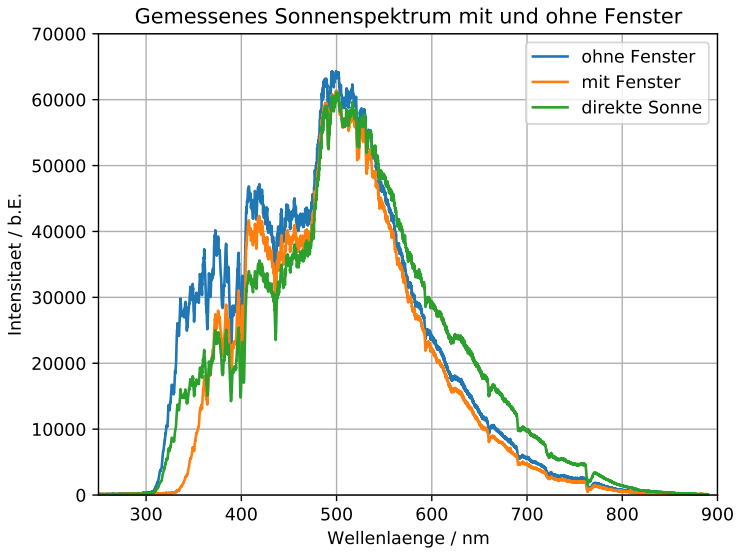


Figure 1: Gemessenes Sonnenspektrum mit und ohne Fenster und in direkte Sonne

Schaut man direkt in die Sonne, bekommt man nochmal mehr kurzwelliges Licht ins Auge, was gefährlich sein kann!
Glas absorbiert stark kurze Wellenlängen, wie man an der Differenz im linken Bereich des Diagramms 2 der blauen und orangen Kurve sieht, damit wir keine Sonnenbrand bekommen. Sichtbares Licht hingegen wird ungehindert durchgelassen.

Hier wird die Absorption des Glases in Abhängigkeit von der Wellenlänge betrachtet. Die Formel zur Berechnung lautet:

$$A_{Glas} = 1 - \frac{I_{mG}(\lambda)}{I_{oG}(\lambda)} \tag{9}$$

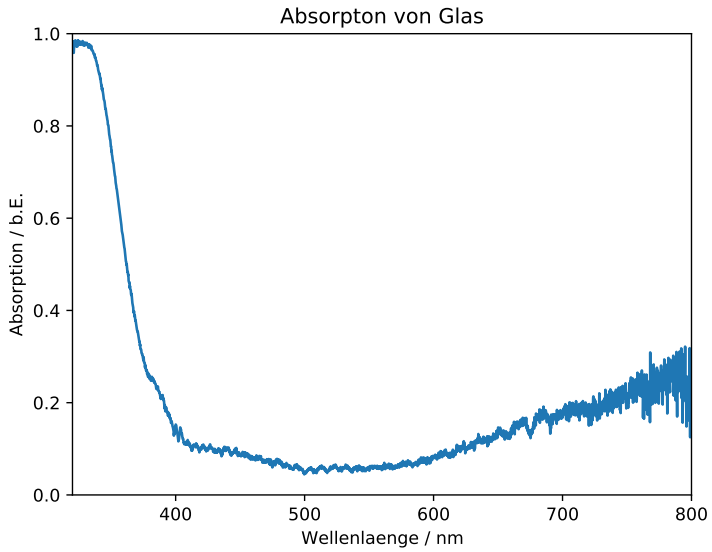


Figure 2: Absorption von Glas

Dabei wird deutlich dass auch ein Anteil der langen Wellenlängen absorbiert wird und sogar nur das Licht im sichtbaren Bereich durchkommt. Da die Absorption hier jedoch schwächer ausfällt, konnten wir das in Diagramm 2 nicht erkennen.

2.3 Frauenhoferlinien

In der Atmosphäre absorbieren die Wasserstoffatom bestimmte Wellenlängen besonders stark. Diese Balmer-Serie Wellenlängen kommen deshalb nicht bei uns in der Kamera an und können aus dem Spektrum der Sonne abgelesen werden. (siehe Diagramm 3)

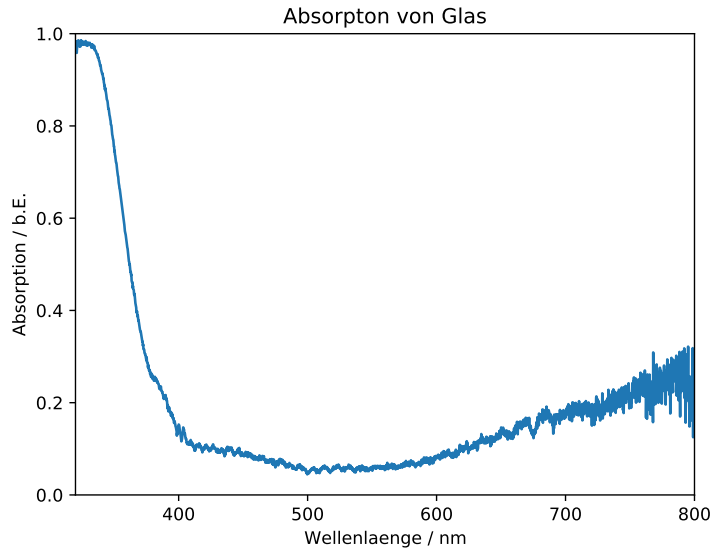


Figure 3: Absorption von Glas

In Diagramm 4 sind die Frauendorfschen Linien als Intensitätsenbrüche zu erkennen. Diese wurden mit der Cursorfunktion bestimmt und in Tabelle 1 mit den Literaturwerten verglichen.

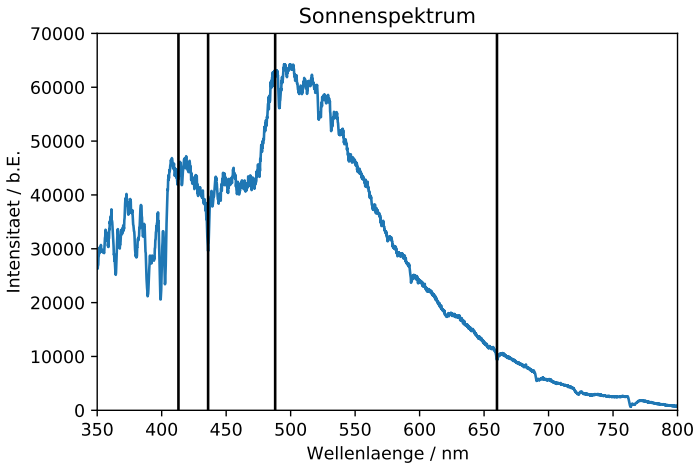


Figure 4: Absorption von Glas

Table 1: Sonnenlichtspektrum: Messung der Frauenhoferlinien und Balmerserie

Bezeichnung	gemessene Wellenlängen [nm]	Literaturwert [nm]	Abweichung
H_δ	413 ± 1	410,1	3,1 σ
H_γ	436 ± 1	431,7	4,3 σ
H_β	488 ± 1	486,1	1,9 σ
H_α	660 ± 1	656,3	3,7 σ

Die Gemessenenn Absorptionsspektren sind im Vergleich zu den Literaturwerten systematische zu hoch. Dies könnte daran liegen, dass andere Effekte der Athmosphäre vernachlässigt wurden.

2.4 Auswertung des Natriumspektrums

Zuordnung der gefunden Linien zu den Natriumserien

Zunächst berechnen wir die zu erwartenden Wellenlängen für die Hauptserie, sowie für die 1. und 2. Nebenserie.

Linien der 1. Nebenserie

Wir nähern zunächst den Korrekturterm des d-Orbitals als 0. Somit gilt für die einzelnen Übergange und ihre Wellenlängen λ_m:

$$\frac{h_c}{\lambda_m} = \frac{E_{Ry}}{m^2} - E_{3p} \Rightarrow E_{3p} = \frac{E_{Ry}}{m^2} - \frac{h_c}{\lambda_m} \tag{10}$$

mit dem Fehler

$$\Delta E_{3p} = \frac{hc}{\lambda_m^2} \Delta\lambda_m \tag{11}$$

Der gemessenen Linie $\lambda = (820,4 \pm 5,5)nm$ ordnen wir den Wert m = 3 zu. Daraus erhält man dann

$$E_{3p} = (-3,032 \pm 0,010)eV \tag{12}$$

Daraus lassen sich die restlichen theoretischen Werte bestimmen.

Linien der 2. Nebenserie

Für die Energie E_{3s} der Nebenserie gilt

$$E_{3s} = E_{3p} - \tfrac{hc}{\lambda} \tag{13}$$

mit den zugehörigen Fehler von

$$\Delta E_{3s} = \sqrt{(\Delta E_{3p})^2 + (\tfrac{hc}{\lambda^2}\Delta\lambda)^2} \tag{14}$$

Außerdem gilt:

$$E_{3s} = \frac{E_{Ry}}{(3-\Delta_s)^2} \tag{15}$$

Stellt man die Gleichung (7) und (12) um so erhält man für Δ_s:

$$\Delta_s = 2 - \sqrt{\frac{E_{Ry}}{E_{3s}}} = 1,372 \pm 0,01 \tag{16}$$

mit dem Fehler

$$\Delta(\Delta_s) = \frac{1}{2}\frac{E_{Ry}}{E_{3s}^2\sqrt{E_{Ry}/E_{3s}}} \tag{17}$$

Nun berechnet man die Wellenlängen der 2. Nebenserie über

$$\lambda_m = \frac{hc}{E_{Ry}/(m-\Delta_s)^2 - E_{3p}} \tag{18}$$

mit dem Fehler

$$\Delta\lambda_m = \sqrt{\Big(\frac{hc\Delta E_{3p}}{E_{Ry}/((m-\Delta_s)^2 - E_{3p}))}\Big)^2 + \Big(\frac{2hc\Delta(\Delta_s)(m-\Delta_s)^3}{(E_{Ry}/(m-\Delta_s)^2 - E_{3p})}\Big)^2} \tag{19}$$

Linien der Hauptserie

Der Korrekturfaktor Δ_p ergibt sich analog zu Δ_s (13) und $\Delta(\Delta_s)$ (14)

$$\Delta_p = 3 - \sqrt{\frac{E_{Ry}}{E_{3p}}} = 0,879 \pm 0,004 \tag{20}$$

Analog die Wellenlängen:

$$\lambda_m = \frac{hc}{E_{Ry}/(m-\Delta_p)^2 - E_{3s}} \tag{21}$$

In den folgenden Tabellen sind die theoretischen Werte, welche durch die obigen Berechnungen bestimmt wurden, mit den experimentell vorgefundenen Werten verglichen, sowohl für die 1. und 2. Nebenserie, als auch für die Hauptserie. Die experimentellen Werte wurden aus den gemessenen Daten, die in den Diagrammen (5), (6), (7) und (8) dargestellt sind bestimmt. Die Berechnungen wurden mit Python durchgeführt und sind im Anhang vorzufinden.

berechnete Wellenlänge [nm]	gemessene Wellenlängen [nm]	Abweichung
-	820,4 ± 5,5	-
570,7 ± 2,7	573 ± 2	0,68 σ
500,2 ± 2,0	503 ± 2	0,71 σ
468,7 ± 1,8	471 ± 2	0,98 σ
451,6 ± 1,7	nicht gefunden	-
441,2 ± 1,6	439 ± 1	1.16 σ
434,3 ± 1,5	432 ± 2	0,89 σ
429,5 ± 1,5	nicht gefunden	-
426,0 ± 1,5	425 ± 2	0.39 σ
423,4 ± 1,5	421 ± 2	0,96 σ

Table 2: Natriumspektrum: 1. Nebenserie

berechnete Wellenlänge [nm]	gemessene Wellenlängen [nm]	Abweichung
1174,5	nicht messbar	-
622,9 ± 3,2	621 ± 2	2,4 σ
519,1 ± 2,2	520 ± 1	0,35 σ
478,0 ± 1,9	472 ± 2	2,2 σ
456,9 ± 1,7	459 ± 1	1,04 σ
444,5 ± 1,6	444 ± 1	0.27

Table 3: Natriumspektrum: 2. Nebenserie

berechnete Wellenlänge [nm]	gemessene Wellenlängen [nm]	Abweichung
332,8 ± 0,9	336 ± 3,1	0,4 σ

Table 4: Natriumspektrum: Hauptserie

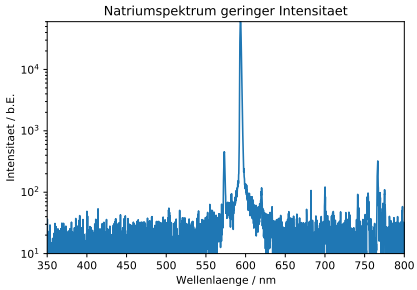


Figure 5: Gesamtes Natriumspektrum

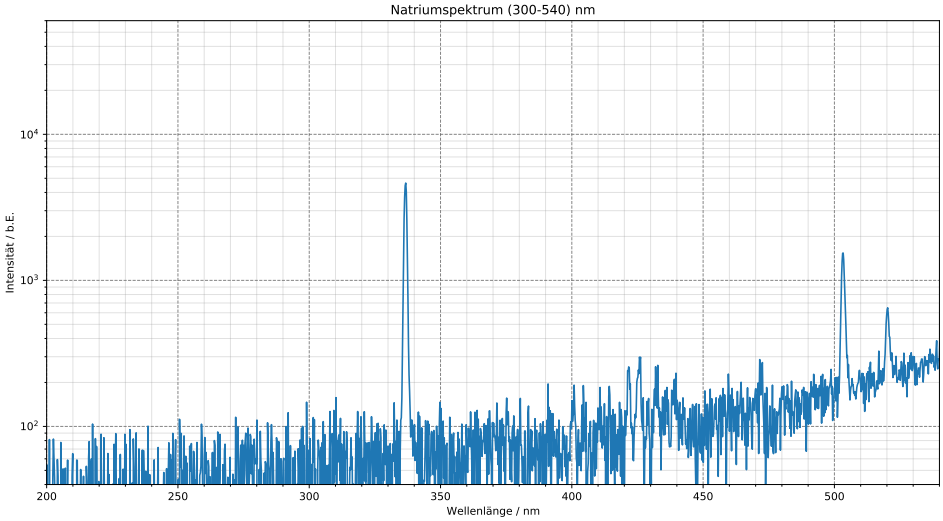


Figure 6: Natriumspektrum geringer Intensität (300-540)nm

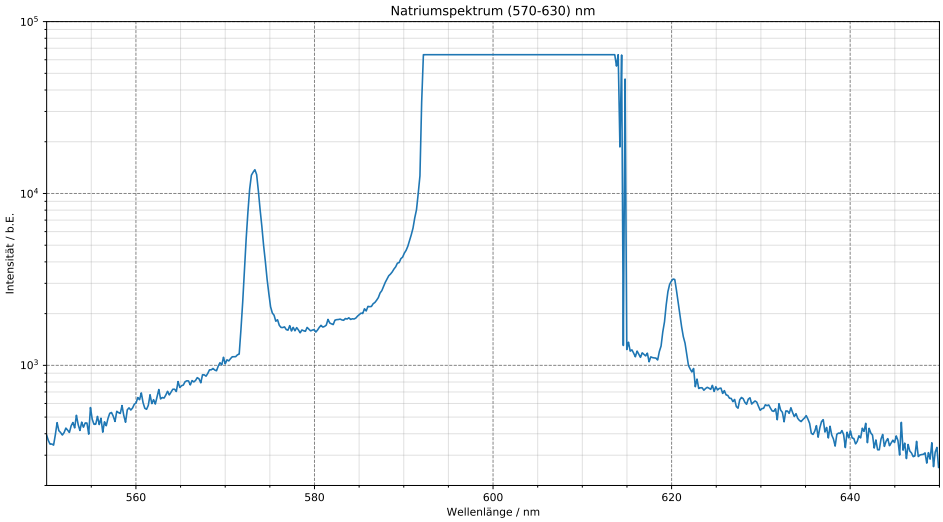


Figure 7: Natriumspektrum geringer Intensität (570-630)nm

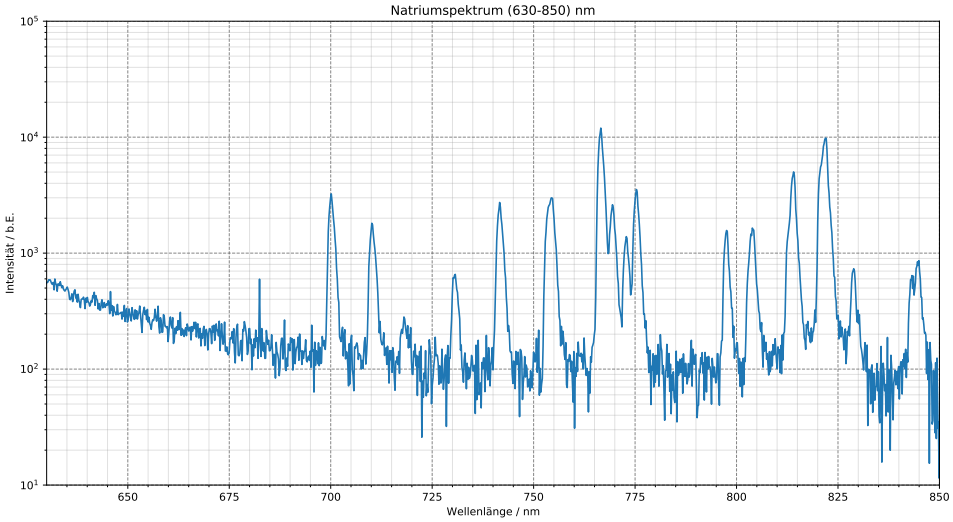


Figure 8: Natriumspektrum geringer Intensität (630-850)nm

Bestimmung der Serienenergien und der l-abhängigen Korrekturfaktoren

Wir bestimmen mit Python die Rydbergenergie E_{Ry}, E_{3p} und die Korrekturterme.
Die Wellenlängen der 1. Nebenserie werden über die Quantenzahl aufgetragen

und es wird ein Fit erstellt.

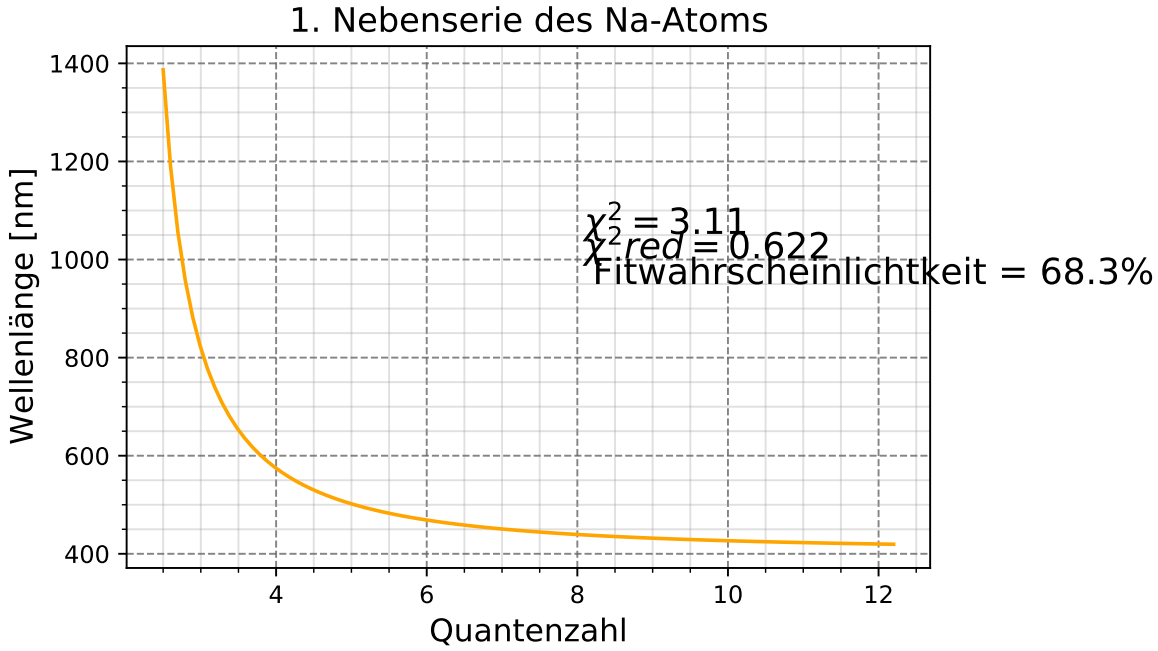


Figure 9: 1. Nebenserie des Na-Atoms

Die bestimmten Werte sind:

- $E_{Ry} = (-16,12 \pm 0,68)$ eV
- $E_{3p} = (-3,06 \pm 0,008)$ eV
- $\Delta_{d,s} = (-0,2281 \pm 0,064)$
- $\chi^2 = 0,3,11$
- $\chi^2_{red} = 0,622$
- Fitwahrscheinlichkeit $68,3\%$

Analog bei der 2. Nebenserie:

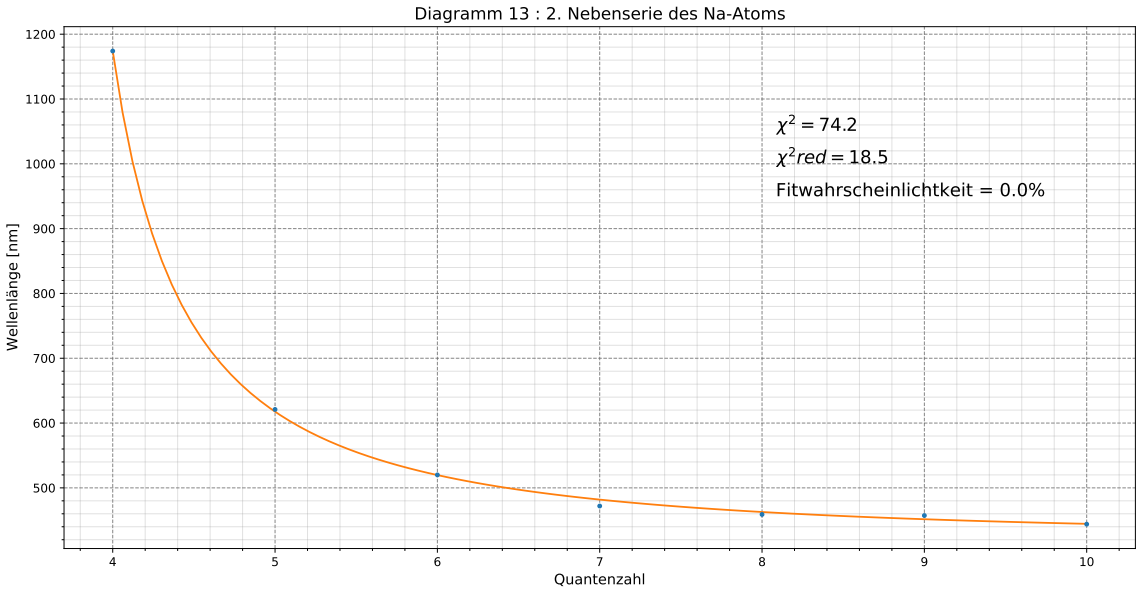


Figure 10: 2. Nebenserie des Na-Atoms

Die bestimmten Werte sind:

- $E_{Ry} = (-10,6 \pm 0,1)$ eV
- $E_{3p} = (-2,941 \pm 0,0029)$ eV
- $\Delta_{d,s} = 1,6$
- $\chi^2 = 74,15$
- $\chi^2_{red} = 18,54$
- Fitwahrscheinlichkeit $0,0\%$

3 Diskussion

In diesem Versuch haben wir unterschiedlichen Lichtquellen analysiert und untersucht.
Beim Sonnenspektrum bzw. Himmelsspektrum konnten wir die Fraunhoferlinien klar erkennen. Interessant war der Unterschied zwischen den diskreten und den kontinuierlichen Lichtquellen, obwohl wir im Spektrum eine noch diskretere Verteilung bei den Nicht-Temepraturstrahlern erwartet haben, jedoch kann dies zumindest teilweise durch Umgebungslicht erklärt werden. Nur der grüne Laser wies eine wirklich diskrete Verteilung auf.

Bei der Analyse des Sonnenspektrums war auffällig, dass das Glas eine sehr hohe Absortionsrate von kurzen Wellenlängen. Dies kann zum Einen an der durchaus starken Verschmutzung der Scheibe liegen, jedoch ist es auch möglich, dass das Material gewollt Licht absorbiert, um dieser Tatsache auf den Grund zu gehen müsste man mehr Informationen über das verwendete Material haben. Die Erkennung der Frauenhoferlinine und der Balmerserie war adäquat möglich mit Abweichungen bis zu 4,3 σ

Wir konnten mehrere Linien nicht zuordnen, da sie entweder nicht erkennbar waren oder nicht in unserem Messspektrum lagen. Die restlichen Messungen ergaben jedoch sehr gute Ergebnisse mit einer maximalen Abweichung zum theoretischen Wert von 1,5σ, wobei 12 der 13 Messungen unter 1σ liegen. Allerdings muss gesagt werden, das es teilweise schwer war, die Linien richtig zuzuordnen wenn sie nahe beieinander liegen oder stark abweichen vom erwarteten Wert. Es ist des Weiteren nicht ausgeschlossen, dass noch andere Absorptionsbanden unsere Messungen verfälschen, bzw. es ist nicht immer eindeutig, ob die Bande den Serien abgehört.

Im letzten Teil berechneten wir die Güte unserer Fits, welche mit 68,3% für die 1. Nebenserie, nah an den angestrebten 50% liegen. Für die zweite Nebenserie war die Fitwahrscheinlichkeit hingegen 0,0%. Das liegt vermutlich an den sämtlichen Vereinfachungen die wir rechenrisch vorgenommen haben, wie bspw. die Unabhängigkeit des Korrekturterms Δ_d von der Hauptquantenzahl n. Außerdem mussten wir ein paar theoretische Werte in unser Diagramm inkludieren, um die nicht gefundenen Linien zu kompensieren. Um hier ein besseres Ergebis erzielen zu können, müsste man die Messungen in einem abgedunkelten Raum und einem hochauflösenden Sensor machen, sowie die Anzahl der Versuchsdurchläufe erhöhen.

Alles in allem sind wir jedoch sehr zufrieden mit unseren Ergebnissen, die diese zumeist mit unseren Erwartungen übereinstimmen.

4 Anhang

Auf den folgenden Seiten ist das Jupyter Notebook, welches für die Datenberechnung und die Plots benutzt wurde, als PDF.

Messprotokoll 234

Unterschrift Tutor: gez von tutor:nikolai bolik

Aufgabe 1

In [1]:

```
%matplotlib inline
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.optimize import curve_fit
from scipy.stats import chi2
plt.xkcd()
plt.rcdefaults()

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

In [2]:

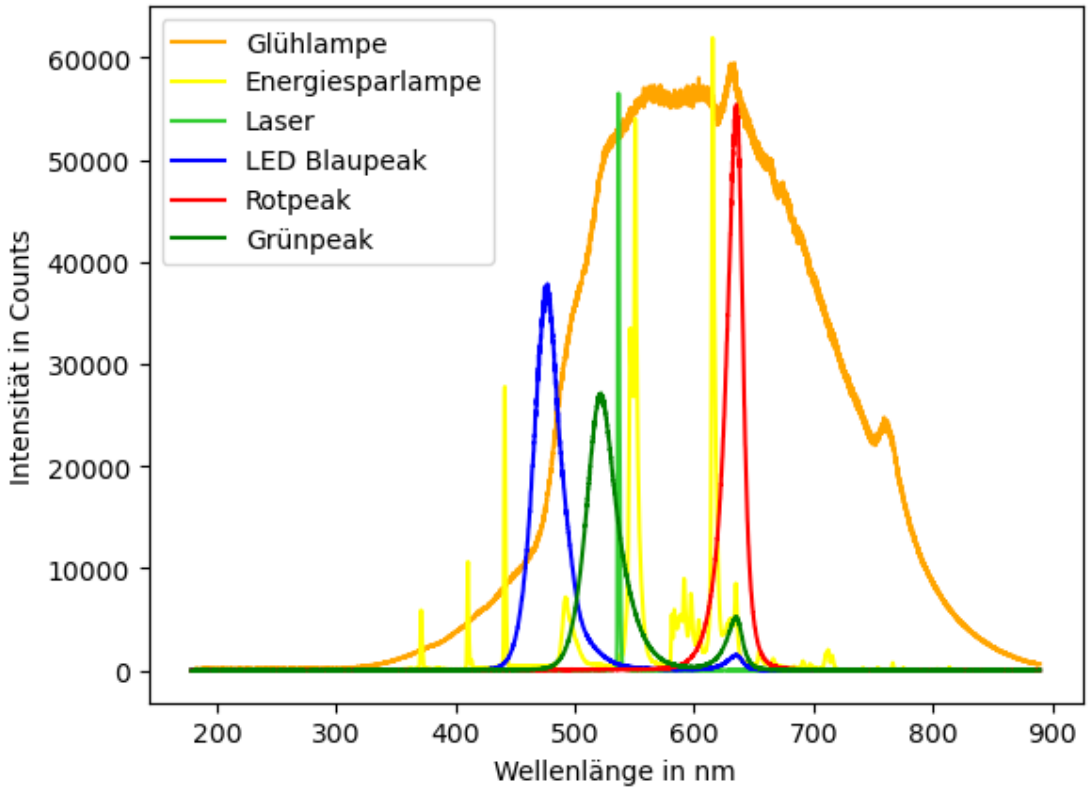
```
#hallogen lampe gabs nicht
```

In [3]:

```
x_glüh, I_glüh = np.loadtxt('Glühlampe.txt',skiprows = 14,converters ={0:comma_to_float
x_energie, I_energie = np.loadtxt('Energiesparlampe.txt',skiprows = 14,converters ={0:c

x_laser, I_laser = np.loadtxt('Laser.txt',skiprows = 14,converters ={0:comma_to_float,
x_LEDblau, I_LEDblau = np.loadtxt('LED Blaupeak.txt',skiprows = 14,converters ={0:comma
x_LEDrot, I_LEDrot = np.loadtxt('LED sehr harter rotpeak.txt',skiprows = 14,converters
x_LEDgrün, I_LEDgrün = np.loadtxt('LED mittelheftiger grünpeak.txt',skiprows = 14,conve
```

In [4]:

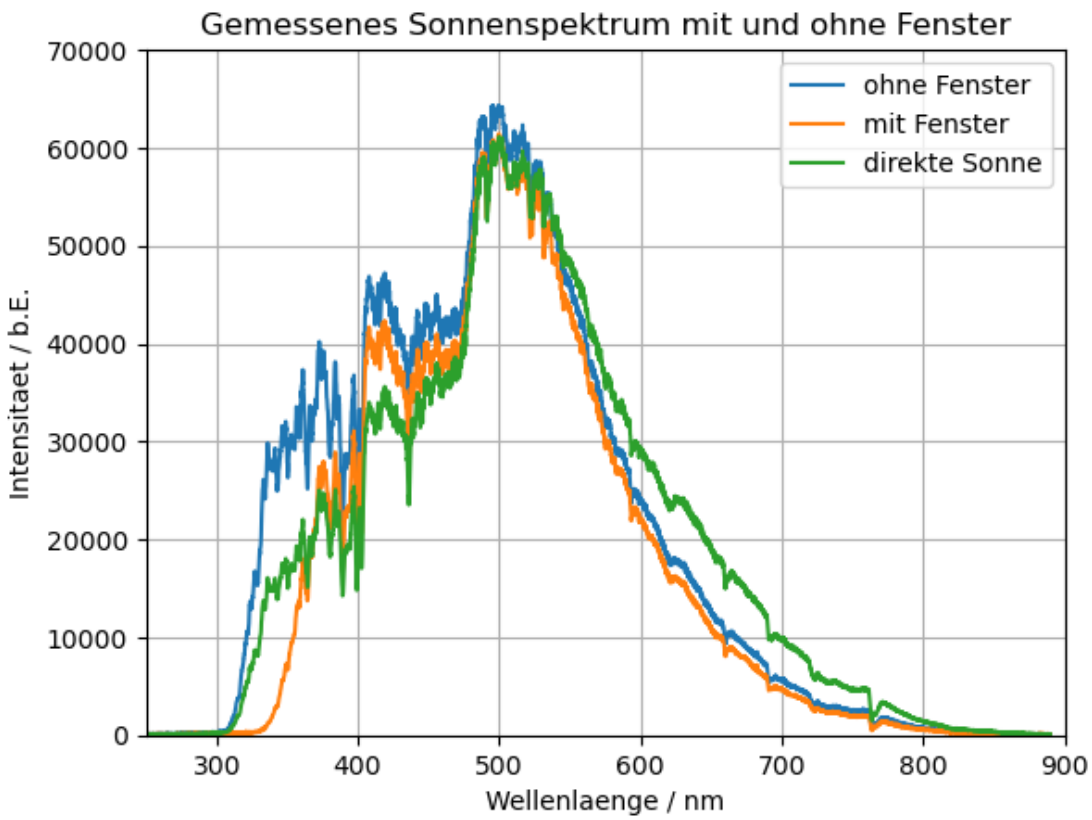
```
x_laser2 = x_laser +500

plt.plot(x_glüh, I_glüh, color = 'orange', label = 'Glühlampe')
plt.plot(x_glüh, I_energie, color = 'yellow', label = 'Energiesparlampe')
plt.plot(x_laser, I_laser, color = 'limegreen', label = 'Laser')
plt.plot(x_LEDblau, I_LEDblau, color = 'blue', label = 'LED Blaupeak' )
plt.plot(x_LEDrot, I_LEDrot, color = 'red', label = 'Rotpeak' )
plt.plot(x_LEDgrün, I_LEDgrün, color = 'green',label = 'Grünpeak'  )
plt.legend()
plt.xlabel('Wellenlänge in nm')
plt.ylabel('Intensität in Counts')

plt.savefig('Lampen.pdf', format = 'pdf')
```



Sonnenspektrum

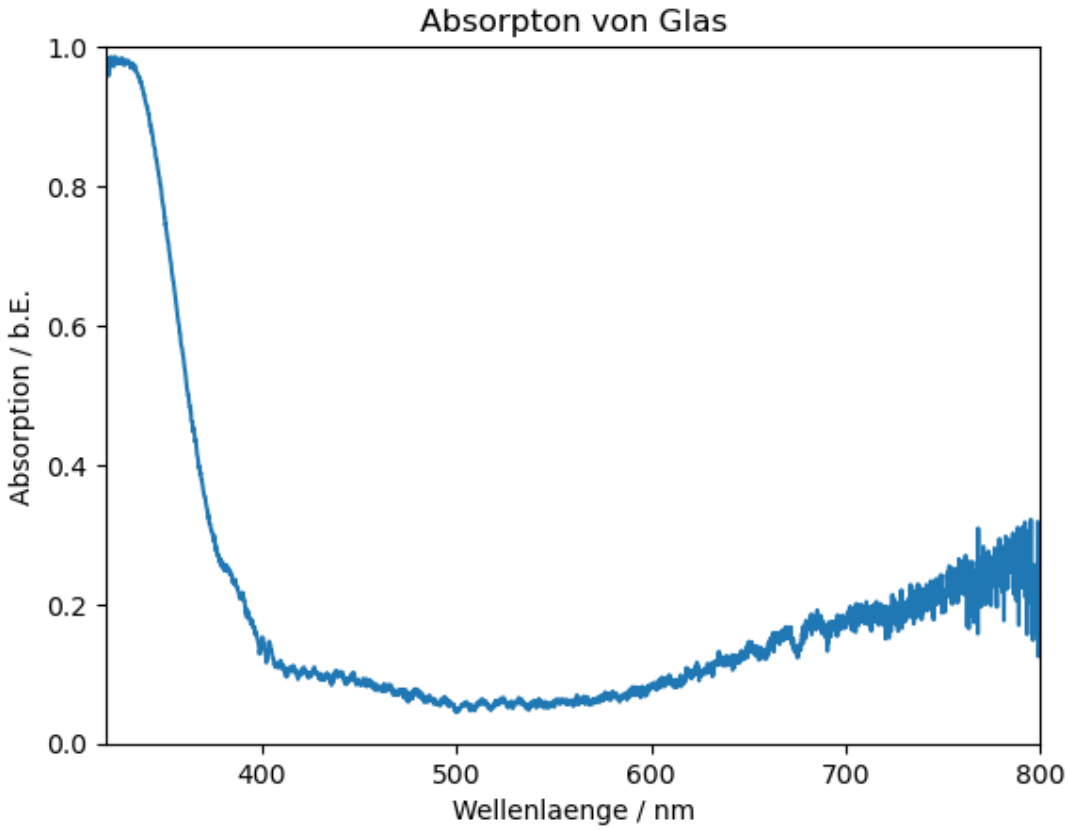
In [5]:

```
lamb_og, inten_og=np.loadtxt('Himmel.txt', skiprows=17, converters= {0:comma_to_float,
#jetzt himmel mit glas
lamb_mg, inten_mg=np.loadtxt('himmel hinterfenster gscheid.txt', skiprows=17, converter
lamb_sonne, inten_sonne=np.loadtxt('Sonne.txt', skiprows=17, converters= {0:comma_to_fl


plt.plot(lamb_og, inten_og, label='ohne Fenster')
plt.plot(lamb_mg, inten_mg, label='mit Fenster')
plt.plot(lamb_sonne, inten_sonne, label='direkte Sonne')
plt.title('Gemessenes Sonnenspektrum mit und ohne Fenster')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.legend()
plt.grid()
plt.ylim((0,70000))
plt.xlim((250,900))
plt.savefig("Himmel_m_o_G_Sonne.pdf", format="pdf")

```



In [6]:

```
1
2 A=1-inten_mg/inten_og
3 plt.plot(lamb_mg, A)
4 plt.title('Absorpton von Glas')
5 plt.xlabel('Wellenlaenge / nm')
6 plt.ylabel('Absorption / b.E.')
7 plt.ylim((0,1))
8 plt.xlim((320,800))
9 plt.savefig("Absorption_Glas.pdf", format = 'pdf')
```



In [7]:

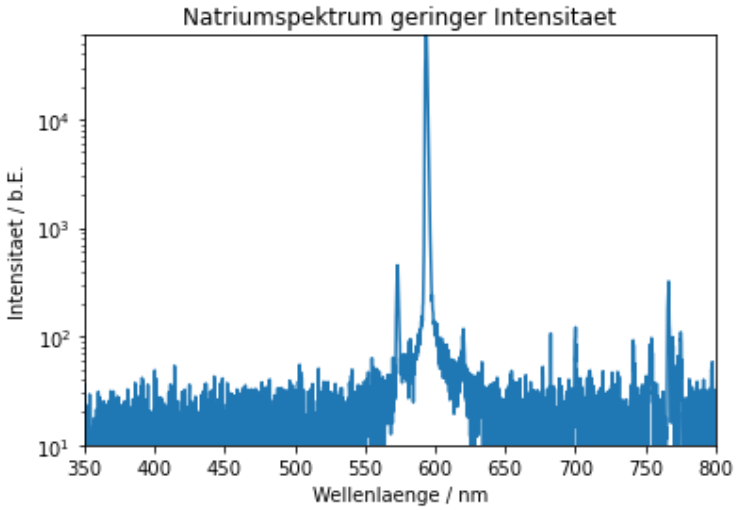
```
%matplotlib notebook
plt.plot(lamb_og, inten_og)
plt.axvline(660, color = 'black', label = 'Balmer alpha')
plt.axvline(488, color = 'black', label = 'Balmer beta')
plt.axvline(436, color = 'black', label = 'Balmer gamma')
plt.axvline(413, color = 'black', label = 'Balmer delta')
plt.title('Sonnenspektrum')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.ylim((0,70000))
plt.xlim((350,800))
plt.savefig("Fraunhofer.pdf", format="pdf")
```

```
<IPython.core.display.Javascript object>
```

Natriumlampen spektrum

In [26]:

```
%matplotlib inline
#um die hauptlinine aufzulösen war keine hohe intensitäy nötig
lamb1, inten1 =np.loadtxt('Natrium Hauptlinie.txt', skiprows=17,
converters= {0:comma_to_float, 1:comma_to_float}, comments='>', unpack=True)
plt.plot(lamb1, inten1)
plt.title('Natriumspektrum geringer Intensitaet')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.yscale('log')
plt.ylim((10,60000))  #y achse ein bisschen verschoben
plt.xlim((350,800))
plt.savefig("Natrium Hauptlinie.pdf", format = "pdf")
```

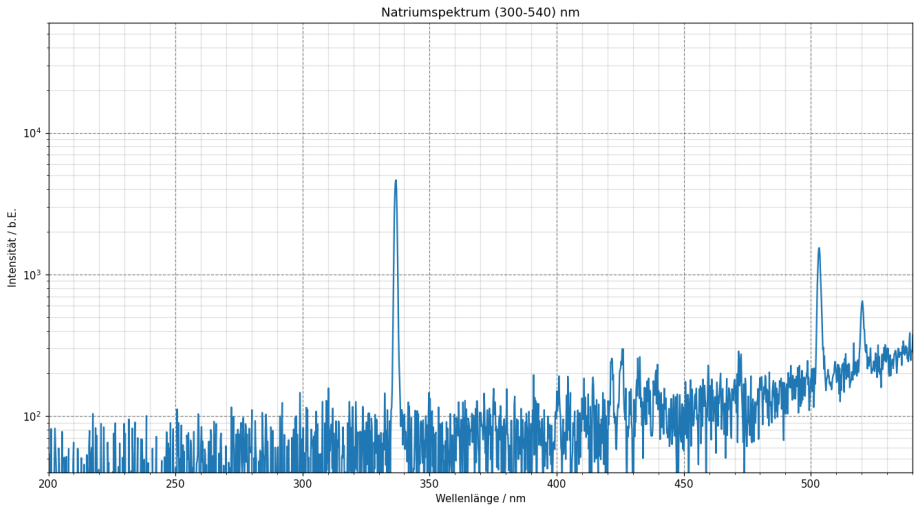


hauptlinie 1 bei 593nm
hauptlinie 2 bei 573nm
fehler 2nm

In [9]:

```
%matplotlib notebook
lamb2, inten2=np.loadtxt('natrium nebenlinien.txt', skiprows=17,
converters= {0:comma_to_float, 1:comma_to_float},
comments='>', unpack=True)
plt.figure(figsize=(15,8))
plt.plot(lamb2, inten2)
plt.title('Natriumspektrum (300-540) nm')
plt.xlabel('Wellenlänge / nm')
plt.ylabel('Intensität / b.E.')
plt.yscale('log')
plt.grid(b=True, which='major', color='#666666', linestyle='--', alpha=0.8)
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.3)
plt.ylim((40,60000))
plt.xlim((200,540))
plt.savefig("Nat2.pdf", format="pdf", bbox_inches='tight')
```

<IPython.core.display.Javascript object>



Die linien die ich hier Rauslesen sind: in nm: pm 1nm 520 \pm 2 503 \pm 3 472 471

414 410 404 400 390 383 380 375 371

439 432 425 \pm 2 421 \pm 2 349

Haupserie 1: 336! andere nicht drauf hm nicht gefunden

In [11]:

```
#kommt da noch was dazwischen?=
lamb3, inten3=np.loadtxt('natrium nebenlinien.txt', skiprows=17,
converters= {0:comma_to_float, 1:comma_to_float},
comments='>', unpack=True)
plt.figure(figsize=(15,8))
plt.plot(lamb3, inten3)
plt.title('Natriumspektrum (570-630) nm')
plt.xlabel('Wellenlänge / nm')
plt.ylabel('Intensität / b.E.')
plt.yscale('log')
plt.grid(b=True, which='major', color='#666666', linestyle='--', alpha=0.8)
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.3)
plt.ylim((2*10**2,10**5))
plt.xlim((550,650))
plt.savefig("Nat3.pdf", format="pdf", bbox_inches='tight')
```

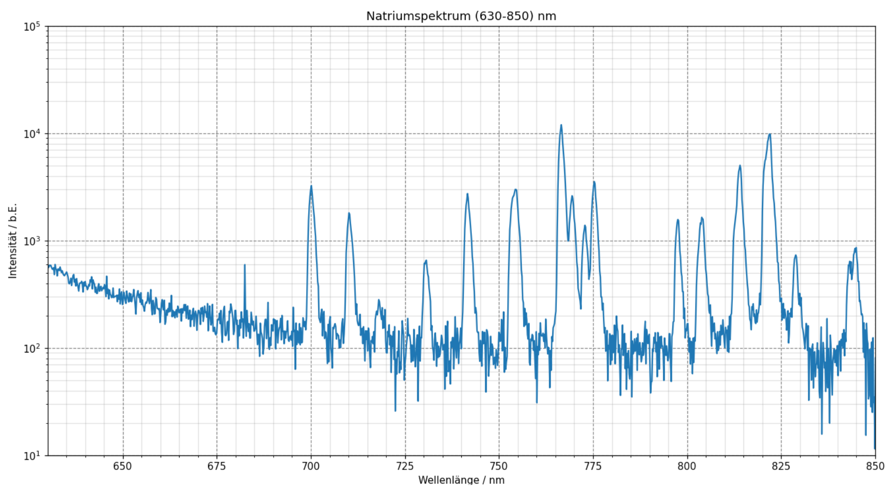
```
<IPython.core.display.Javascript object>
```

621 \pm 2
573 \pm 2
eingetragen

In [13]:

```
%matplotlib notebook
lamb4, inten4=np.loadtxt('natrium nebenlinien.txt', skiprows=17,
converters= {0:comma_to_float, 1:comma_to_float},
comments='>', unpack=True)
plt.figure(figsize=(15,8))
plt.plot(lamb4, inten4)
plt.title('Natriumspektrum (630-850) nm')
plt.xlabel('Wellenlänge / nm')
plt.ylabel('Intensität / b.E.')
plt.yscale('log')
plt.grid(b=True, which='major', color='#666666', linestyle='--', alpha=0.8)
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.3)
plt.ylim((1*10,10**5))
plt.xlim((630,850))
plt.savefig("Nat4.pdf", format="pdf", bbox_inches='tight')
```

<IPython.core.display.Javascript object>



gefundene zuordenbare linien in nm pm 1nm: 844
828
821 813 803 797 775 766 754 741 730 717 710 690 682

über 700 ist wahrschl krypton

521 503 472 471 439 432 425 421 414 410 404 400 390 383 380 375 371 349

Theoretische Berechnungen:

In [14]:

```
hc = 1.2398e3 # nm eV
lam_31 = 820.4
del_lam_31 = 5.5
m_lam_31 = 3 #Ordnung
E_Ry_31 = -13.605 #eV
E_3p_31 = E_Ry_31/(m_lam_31**2)-hc/lam_31
del_E_3p_31 = hc/(lam_31**2)*del_lam_31
m_31 = np.arange(3,13,1) #Ordnung der Linie der ersten Nebenserie
lam_ex_31 = hc/(E_Ry_31/m_31**2-E_3p_31) #nm
del_lam_ex_31 = hc/(E_Ry_31/m_31**2-E_3p_31)**2*del_E_3p_31
```

In [15]:

```
for i in range(len(m_31)):
    print("m = ", m_31[i])
    print("lam_ex_31 [nm]]: ", np.round(lam_ex_31[i],1))
    print("del_lam_ex_31 [nm] : ", np.round(del_lam_ex_31[i],1))


```

```
m =  3
lam_ex_31 [nm]]:  820.4
del_lam_ex_31 [nm] :  5.5
m =  4
lam_ex_31 [nm]]:  570.7
del_lam_ex_31 [nm] :  2.7
m =  5
lam_ex_31 [nm]]:  500.2
del_lam_ex_31 [nm] :  2.0
m =  6
lam_ex_31 [nm]]:  468.7
del_lam_ex_31 [nm] :  1.8
m =  7
lam_ex_31 [nm]]:  451.6
del_lam_ex_31 [nm] :  1.7
m =  8
lam_ex_31 [nm]]:  441.2
del_lam_ex_31 [nm] :  1.6
m =  9
lam_ex_31 [nm]]:  434.3
del_lam_ex_31 [nm] :  1.5
m =  10
lam_ex_31 [nm]]:  429.5
del_lam_ex_31 [nm] :  1.5
m =  11
lam_ex_31 [nm]]:  426.0
del_lam_ex_31 [nm] :  1.5
m =  12
lam_ex_31 [nm]]:  423.4
del_lam_ex_31 [nm] :  1.5
```

In [16]:

```
#Berechnung von E_3s anhand der Linie bei 589nm:
lam_32 = 590.6
del_lam_32 = 3.0
E_3s_32 = E_3p_31 - hc/lam_32 #eV
del_E_3s_32 = np.sqrt((del_E_3p_31)**2+(1239.8*del_lam_32/(lam_32**2))**2)
print("E_3s_32 = ", np.round(E_3s_32,3), '+/-', np.round(del_E_3s_32,3), "eV")
```

```
E_3s_32 =  -5.122 +/- 0.015 eV
```

In [17]:

```
#Berechnen des Korrekturfaktors:
delta_s = 3 - np.sqrt(E_Ry_31/E_3s_32) #eV
del_delta_s = 0.5*13.605*del_E_3s_32/(((E_3s_32)**2)*np.sqrt(E_Ry_31/E_3s_32))
print("delta_s = ", np.round(delta_s,3), "+/-", np.round(del_delta_s,3))
```

```
delta_s =  1.37 +/- 0.002
```

In [18]:

```
#Berechnen der Wellenlängen der zweiten Nebenserie:
m_32 = np.arange(4, 10, 1) #Ordnungen der zweiten Nebenserie
lam_ex_32 = hc/(E_Ry_31/(m_32-delta_s)**2 - E_3p_31) #nm
temp1 = hc/(E_Ry_31/(m_32-delta_s)**2-E_3p_31)**2*del_E_3p_31
temp2 = (hc/(E_Ry_31/(m_32-(delta_s+del_delta_s))**2-E_3p_31))-(hc/(E_Ry_31/(m_32-delta
del_lam_ex_32 = np.sqrt(temp1**2 + temp2**2)
for i in range(len(m_32)):
    print("m = ", m_32[i])
    print("lam_ex_32 [nm]: ", np.round(lam_ex_32[i],1))
    print("del_lam_ex_32 [nm]: ", np.round(del_lam_ex_32[i],1))
```

```
m =  4
lam_ex_32 [nm]:  1174.5
del_lam_ex_32 [nm]:  11.9
m =  5
lam_ex_32 [nm]:  622.9
del_lam_ex_32 [nm]:  3.2
m =  6
lam_ex_32 [nm]:  519.1
del_lam_ex_32 [nm]:  2.2
m =  7
lam_ex_32 [nm]:  478.0
del_lam_ex_32 [nm]:  1.9
m =  8
lam_ex_32 [nm]:  456.9
del_lam_ex_32 [nm]:  1.7
m =  9
lam_ex_32 [nm]:  444.5
del_lam_ex_32 [nm]:  1.6
```

hauptserie

In [29]:

```
#Berechnung des Korrekturfaktors delta_p aus E_3p:
delta_p = 3 - np.sqrt(E_Ry_31/E_3p_31)
del_delta_p = -np.sqrt(E_Ry_31/E_3p_31)/(2*E_3p_31)*del_E_3p_31
print("delta_p = ", np.round(delta_p,3), "+/-", np.round(del_delta_p,3))
```

```
delta_p =  0.879 +/- 0.004
```

In [30]:

```
#Berechnen der Wellenlängen der Hauptserie:
m_33 = np.arange(4, 6, 1)
lam_ex_33 = hc/(E_Ry_31/(m_33-delta_p)**2-E_3s_32)
temp1 = hc/(E_Ry_31/(m_33-delta_p)**2-E_3s_32)**2*del_E_3p_31
temp2 = hc/(E_Ry_31/(m_33-(delta_p+del_delta_p))**2-E_3s_32)-hc/(E_Ry_31/
(m_33-delta_p)**2-E_3s_32)
del_lam_ex_33 = np.sqrt(temp1**2 + temp2**2)
for i in range(len(m_33)):
    print("m = ", m_33[i])
    print("lam_ex_33 = ", np.round(lam_ex_33[i],1))
    print("del_lam_ex_33 = ", np.round(del_lam_ex_33[i],1))
```

```
m =  4
lam_ex_33 =  332.8
del_lam_ex_33 =  0.9
m =  5
lam_ex_33 =  286.9
del_lam_ex_33 =  0.7
```

In [31]:

```
wellen2 = np.array([820.4, 573, 503, 471,0, 439, 432,0,  425, 421])
del_wellen2 = np.array([5.5, 2, 2, 2, 0, 1, 2,0, 2, 2])
sigma =   (lam_ex_31-wellen2)/np.sqrt(del_lam_ex_31**2 + del_wellen2**2)
print(lam_ex_31)
print(del_lam_ex_31)
print(sigma)

#nuller wurden für nicht fundene hinzugefügt!
```

```
[820.4        570.66102463 500.18543966 468.73983175 451.6201056
 441.1624596  434.26820331 429.46751131 425.98330604 423.37090124]
[5.5        2.66114053 2.0444358  1.7954574  1.66670187 1.59040776
 1.54108804 1.50720393 1.48284769 1.4647159 ]
[ 1.46161334e-14 -7.02624133e-01 -9.84104926e-01 -8.40933982e-01
  2.70966340e+02  1.15105946e+00  8.98345732e-01  2.84943201e+02
  3.94942242e-01  9.56397968e-01]
```

In [32]:

```
wellen3 = np.array([0, 632, 520, 472, 459, 444])
delwellen3 = np.array([0, 2,1,2,1,1])

sigma2 =   (lam_ex_32-wellen3)/np.sqrt(del_lam_ex_32**2 + delwellen3**2)
print(sigma2)
```

```
[98.42985549 -2.40335413 -0.35341081  2.19899418 -1.04870874  0.26577004]
```

Bestimmung der Serienenergien und der l-abhängigen Korrekturfaktoren

In [36]:

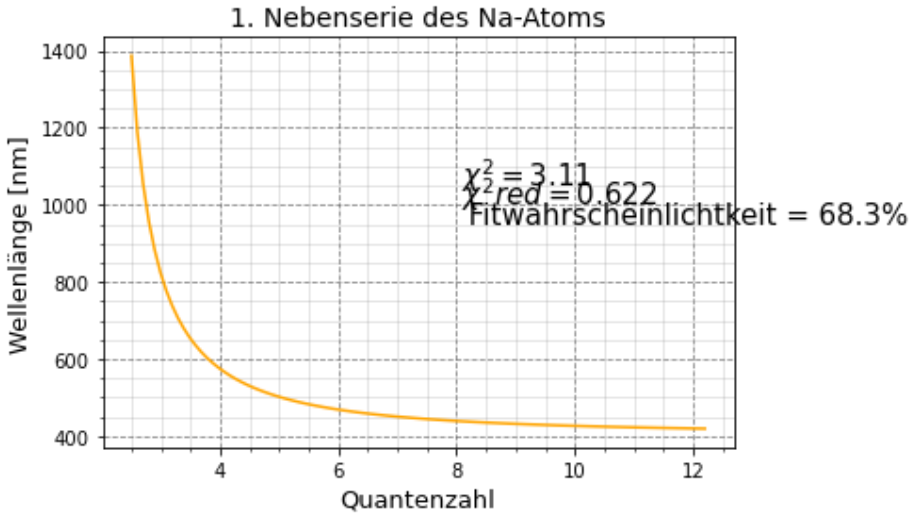
```
wellen1 = np.array([820.4, 573, 503, 471, 439, 432, 425, 421])
del_wellen1 = np.array([5, 2, 2, 2, 1, 2, 2, 2])
quantenz = np.array([3,4,5,6,8,9,10,12])



#die gemessenen Werte werden geplottet und eine Funktion gefittet:
def fit_func(m,E_ry,E_3p,D_d):
    return 1.2398E3/(E_ry/(m-D_d)**2-E_3p)
para = [-13.6,-3,-0.02]
popt, pcov = curve_fit(fit_func, quantenz, wellen1, sigma=del_wellen1 ,p0=para)

#Berechnen der Güte des Fits:
chi1_=np.sum((fit_func(quantenz,*popt)-wellen1)**2/del_wellen1**2)
dof=len(quantenz)-3 #dof:degrees of freedom, Freiheitsgrad
chi1_red=chi1_/dof
prob1=round(1-chi2.cdf(chi1_,dof),3)*100

plt.figure(figsize=(16,8))

plt.errorbar(quantenz, wellen1, yerr = del_wellen1, fmt=".", markersize = 5, capsize =
```

Out[36]:

```
<ErrorbarContainer object of 3 artists>
```

In [34]:

```
plt.xlabel('Quantenzahl',fontsize=13)
plt.ylabel('Wellenlänge [nm]',fontsize=13)
plt.title('1. Nebenserie des Na-Atoms',fontsize=14)
x = np.linspace(2.5, 12.2, 100)
plt.plot(x, fit_func(x, *popt), color = "orange")
plt.grid(b=True, which='major', color='#666666', linestyle='--', alpha=0.8)
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.3)
plt.text(8.09, 1050, f'$\chi^2 = {chi1_:.03}$', size = 15)
plt.text(8.09, 1000, f'$\chi^2red = {chi1_red:.03}$', size = 15)
plt.text(8.05, 950, f' Fitwahrscheinlichtkeit = {prob1:.03}%', size = 15)
plt.savefig("chi.pdf", format='PDF', bbox_inches='tight')
```



In [35]:

```
#Ausgabe der Parameter der Fitkurve:
print("E_Ry [eV]:",np.round(popt[0],2), ", Standardfehler =", np.round(np.sqrt(pcov[0]|
print("E_3p [eV]:",np.round(popt[1],3), ", Standardfehler =", np.round(np.sqrt(pcov[1]|
print("D_d =",np.round(popt[2],4), ", Standardfehler =", np.round(np.sqrt(pcov[2][2]),3
```

```
E_Ry [eV]: -16.12 , Standardfehler = 0.68
E_3p [eV]: -3.06 , Standardfehler = 0.008
D_d = -0.2281 , Standardfehler = 0.064
```

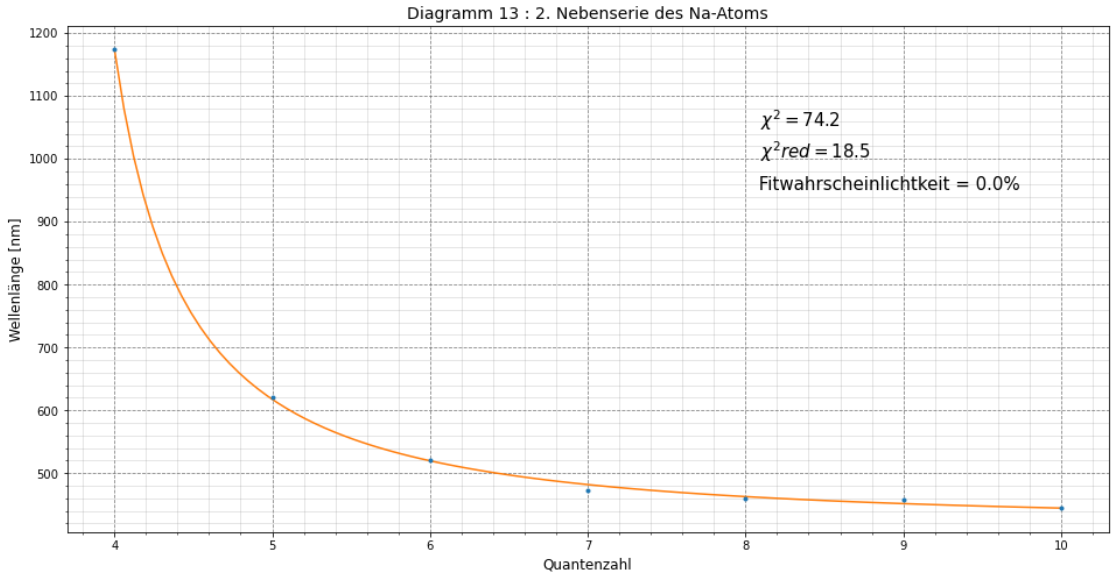
In [61]:

```
print("chi1 =", np.round(chi1_,4))
print("chi1_red =",np.round(chi1_red,4))
print("Fitwahrscheinlichkeit:", prob1,"%")
```

```
chi1 = 3.1105
chi1_red = 0.6221
Fitwahrscheinlichkeit: 68.30000000000001 %
```

In [62]:

```
#analog zu oben:
wellen2 = np.array([ 1174, 621, 520, 472, 459, 457.3, 444])
del_wellen2 = np.array([1,  2.0, 1, 2, 1,1, 1])
quantenz2 = np.arange(4, 11)
def fit_func2(m,E_ry2,E_3p2,delta_s):
    return 1.2398E3/(E_ry2/(m-delta_s)**2-E_3p2)
para = [-13.6 ,-3,-0.02]
popt2, pcov2 = curve_fit(fit_func2, quantenz2, wellen2, sigma=del_wellen2,p0=para)
#Berechnen der Güte des Fits:
chi2_=np.sum((fit_func2(quantenz2,*popt2)-wellen2)**2/del_wellen2**2)
dof=len(quantenz2)-3 #dof:degrees of freedom, Freiheitsgrad
chi2_red=chi2_/dof
prob2=round(1-chi2.cdf(chi2_,dof),2)*100

plt.figure(figsize=(16,8))
plt.errorbar(quantenz2, wellen2, del_wellen2, fmt=".")
plt.xlabel('Quantenzahl',fontsize=12)
plt.ylabel('Wellenlänge [nm]',fontsize=12)
plt.title('Diagramm 13 : 2. Nebenserie des Na-Atoms',fontsize=14)
x2 = np.linspace(4, 10, 100)
plt.plot(x2, fit_func2(x2, *popt2))
plt.grid(b=True, which='major', color='#666666', linestyle='--', alpha=0.8)
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.3)
plt.text(8.09, 1050, f'$\chi^2 = {chi2_:.03}$', size = 15)
plt.text(8.09, 1000, f'$\chi^2red = {chi2_red:.03}$', size = 15)
plt.text(8.05, 950, f' Fitwahrscheinlichtkeit = {prob2:.03}%', size = 15)
plt.savefig("chi1.pdf", format='PDF', bbox_inches='tight')
```



In [63]:

```
print("E_Ry2 [eV]:",np.round(popt2[0],1), ", Standardfehler=", np.round(np.sqrt(pcov2[0
print("E_3p2 [eV]:",np.round(popt2[1],3), ", Standardfehler=", np.round(np.sqrt(pcov2[1
print("delta_s=",np.round(popt2[2],1), ", Standardfehler=", np.round(np.sqrt(pcov2[2][2
```

```
E_Ry2 [eV]: -10.6 , Standardfehler= 1.3
E_3p2 [eV]: -2.941 , Standardfehler= 0.029
delta_s= 1.6 , Standardfehler= 0.1
```

In [64]:

```
print("chi2=", np.round(chi2_,2))
print("chi2_red=",np.round(chi2_red,2))
```

```
chi2= 74.15
chi2_red= 18.54
```

In [ ]:

```

```