Versuchsprotokoll Experiment 234 PAP2: Lichtquellen

Leonard Scheuer

Motivation

Hier sollen Lichtspektren verschiedener Lichtquellen und das Sonnenlicht untersucht werden. Daraus können über Spektrallinien Aussagen über den Aufbau der Lichtquellen sowie Zusammensetzung der Athmosphäre und Sonne im letzteren Fall gewonnen werden.

Grundlagen und Einleitung

Terminologie

Für die weitere Verwendung gelte folgendes:

Bezeichnung	Formelzeichen	Wert	(Quelle)
Plank'sches WQ	h	$6.62607015 \cdot 10^{-34}\mathrm{Js}$	NIST
Lichtgeschwindigkeit	c	$2.99792458 \cdot 10^{8} m/s$	NIST
Boltzmannkonstante	k	$1.380649 \cdot 10^{-23} J/kg$	NIST
Elektrische Feldkonst.	ε_0	$8.8541878128 \cdot 10^{-12} F/m$	NIST
Rydbergenergie	E_{Rd}	$13.605693122994 eV$	NIST

Wir wollen zunächst betrachten, auf welcher Grundlage elektromagnetische Strahlung im gemessenen Wellenlängenbereich entsteht. Dazu betrachten wir zunächst Temperaturstrahler und Streueffekte.

Temperaturstrahler

Jeder Objekt mit von Null verschiedener Temperatur T sendet Strahlung aus, betrachten wir einen Schwarzkörper (der alle einfallende Strahlung zunächst absorbiert), wird diese durch das Planck'sche Strahlungsgesetz beschrieben. Sei dA ein Flächenelement des Schwarzkörpers und $[\lambda, \lambda + d\lambda]$ das betrachtete Wellenlängenintervall, so besagt es für die Strahlungsleistung M_λ:

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

Ist $\lambda_{\max}$ die Wellenlänge, an welcher M_λ Maximal wird, dann besagt das Wiensche Verschiebungsgesetz:

$$\lambda_{\max} = \frac{2.8978 K \cdot nm}{T} \tag{2}$$

Nichttemperaturstrahler

Durch direkte Anregung eines Elektrons lässt sich ebenfalls eine Lichtemission hervorrufen, wenn dieses wieder in einen Zustand mit niedriger Energie zurück fällt. Da die Energiezustände im allgemeinen quantisiert sind, ist das daraus entstehende Spektrum wieder diskret. Auf ähnliche Weise kann ein Photon bei einer Elketron-Loch-Rekombination abgegeben werden. Bei Leuchtmitteln, die auf einem dieser Prinzipien basieren, wie LEDs, wird gelegntlich noch eine foureszierende Überzug eingesetzt, der die diskret verteilten Photonen aufnimmt und über einen weiteren Bereich verteilt, ggf. in den sichtbaren Bereich verschiebt und wieder abgibt.

Rayleigh-Streuung

Die charakteristische Tageszeitabhängige Farbgebung unseres Himmels ist der Rayleigh-Streuung zu verdanken. Die Streuung von Elektomagnetischer Strahlung an, im Verhältnis zur Wellenlänge, kleinen Teilchen folgt mit der vierten Potenz der Frequenz der Strahlung. Tagsüber wird dadurch das blaue Licht stärker gestreut, daher ist es nahezu von allen Richtungen erkennbar. Das Rote Licht hingegen wird weniger gestreut und ist daher auch noch am Abend sichtbar, wenn der Weg des Lichts durch die Athmosphäre länger ist.

Natriumspektrum

In diesem Versuch wir das Spektrum der Natriumdampflampe näher untersucht werden. Hier soll daher das Natriumatom inklusive seiner Energieniveaus

modellhaft beschrieben werden. Wir betrachten das Potential $V(r)$ des Valenzelektrons im Abstand r vom Kern. Nun machen wir die Näherung, dass sich die inneren Elektronen mit der korresponierenden Kernladung etwa ausgleicht und sich daher der Kern für unsere Zwecke wie eine einfach geladene Punkladung verhält. Wir erhalten:

$$V(r) = -\frac{e^2}{4\pi\varepsilon_0 r} \tag{3}$$

Wir können nun die Ernergieniveaus in Abhängigkeit von der Hauptquantenzahl n und der Nebenquantenzahl l angeben:

$$E_{n,l} = -\frac{13.605eV}{(n-\Delta_{n,l})^2} \tag{4}$$

Wobei $\Delta_{n,l}$ ein Korrekturtherm ist. Wir bestimmen die im Spektrum erwarteten Peaks über die Energiedifferenzen, die in der folgenden Abbildung dargestellt sind, mittels:

$$\Delta E = h \cdot f = h \cdot \frac{c}{\lambda} \tag{5}$$

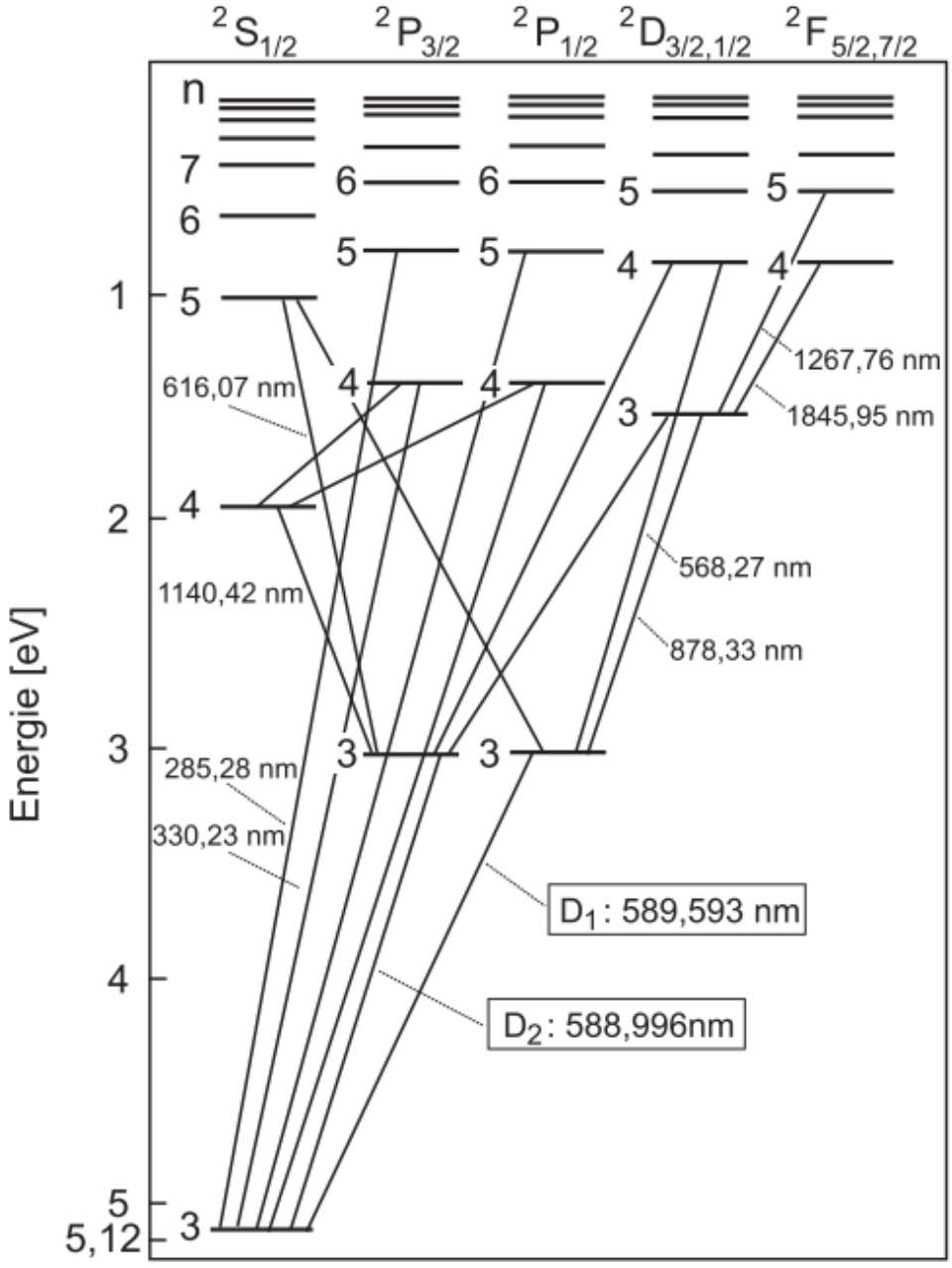


Abbildung 1: Energieniveaus von Natrium (Quelle: Script)

Wir werden nutzen, für E_{3p} der Korrekturtherm vernachlässigbar ist. Wir bestimmen diese Energie via:

$$h \cdot \frac{c}{\lambda_m} = E_{\mathrm{Ry}} - E_{3p} \tag{6}$$

die genaue Rechnung und Fehlerrechnung wird in der Auswertung dargelegt.

Gitterspektrometer

Das hier verwendete Spektometer besteht aus einem das Licht (in Abhängigkeit der Wellenlänge) unterscheidlich stark brechendem Gitter und einem Ortsaufgelösten Photosensor. Das zu Untersuchende Licht wird über einen Lichtwellenleiter eingeleitet und über Spiegel auf ein Gitter und anschließend auf einen CCD-Sensor gelenkt. Dabei wird das Licht (abhängig von der Gitterkonstante) nach Wellenlänge in unterschiedlichem Winkel reflektiert. Der Ort, wo das Licht auf den CCD-Sensor trifft, gibt daher nun aufschluss über die Wellenlänge (Vgl. Abb. 2).

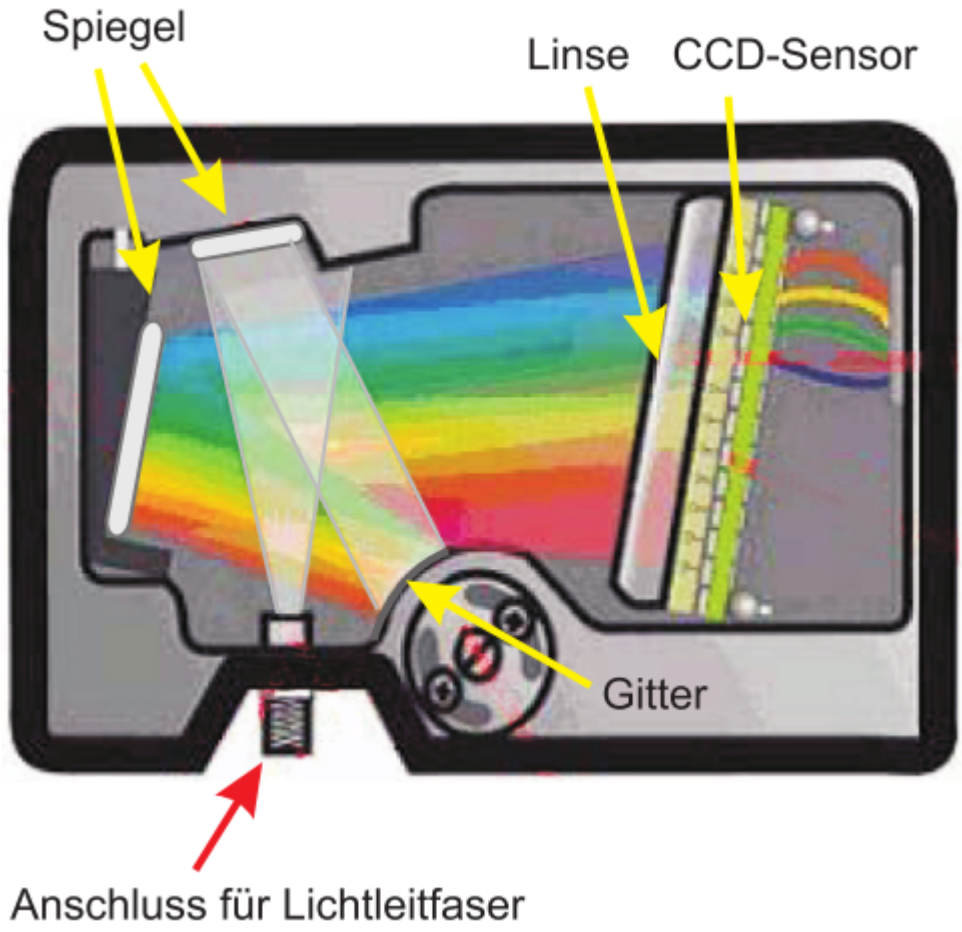


Abbildung 2: Gitterspektrometer (Quelle: Script)

Durchführung/Messungen

Material

- Gitterspektrometer
- Einkoppler mit Lichtwellenleiter zum Gitterspektrometer
- Lichtquellen: LED, LASER, Energiesparlampe, Halogenlampe, Glühhbirne, Natriumdampflampe

Betrachtung des Tageslichts-/Sonnenspektrum

Hier werden nach entsprechender Dunkelmessung folgende Spektren aufgenommen:

- Himmelslicht bei geöffneten Fenster.
- Himmelslicht durch Fensterglas.
- Betrachtung des direkten Sonnenlichts durch das Fenster, wenn witterungsbedingt möglich. (Bei uns war dies nicht der Fall)

Lichtquellenspektren

Wir nehmen hier qualitativ (d.h. ohne vorherige Dunkelmessung) die Spektren der im Material angegbenen Lichtquellen auf.

Natriumspektrum

Hier sollen die Emissionslinien von Natrium bestimmt werden. Dazu wird zuerst die Natriumdampflampe eingeschaltet und gewartet bis ein stabiles Leuchten eintritt. Um Emissionslinien verschiedener Ausprägung sichtbar Aufnehmen zu können, werden hier zwei Messungen mit verschiedener Integrationszeit vorgenommen, einmal sodass die gelbe Hauptlinie in Sättigung geht und einmal so, dass dies gerade nicht geschieht. Tatsächlich ließ sich letzteres im Experiment nicht realisieren, es wurde stattdessen eine andere Messung mit geringerer Integrationszeit vorgenommen.

In [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import pandas as pd
from IPython.display import Markdown, display

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

lamb_og, inten_og = np.loadtxt('himmel_o_g.txt', skiprows=17, converter
comments = '>', unpack = True)
```

```
lamb_mg, inten_mg = np.loadtxt('himmel_m_g.txt', skiprows=17, converter
comments = '>', unpack = True)
```

Auswertung

Lichtquellespektren

In diesem Abschnitt sollen verschiedene Lichtquellen qualitativ untersucht werden und dadurch auf Rückschlüsse auf ihre Funktionsweise gemacht werden. Wir haben die in Tab. 1. aufgeführten Lichtquellen untersucht und ihre Spektren mit geschätztem Mittelwert in Abbildung 3 dargestellt.

Tab. 1: verschiedene Lichtquellen

Leuchtmittel	Farbe im Diagramm	Mittlere Wellenlänge [nm]	Warm / Kalt
Glühlampe	Blau	640	Warm
Energiesparlampe	Hellgrün	565	Warm
LED-Birne	Grau	595	Warm
LED1 (Weiß)	Rosa	565	Warm
LED2 (Weiß)	Dunkelgrün	565	Warm
LED3 (Blau)	Dunkelblau	460	Kalt
LED4 (Kaltweiß)	Gelb	510	Kalt
LED5 (Orange)	Violett	610	Warm
LED6 (Gelb)	Türkis	595	Warm
LED7 (Rot)	Schwarz	630	Warm
LASER	Rot	530	Kalt
Halogenlampe	Hellgrau	640	Warm

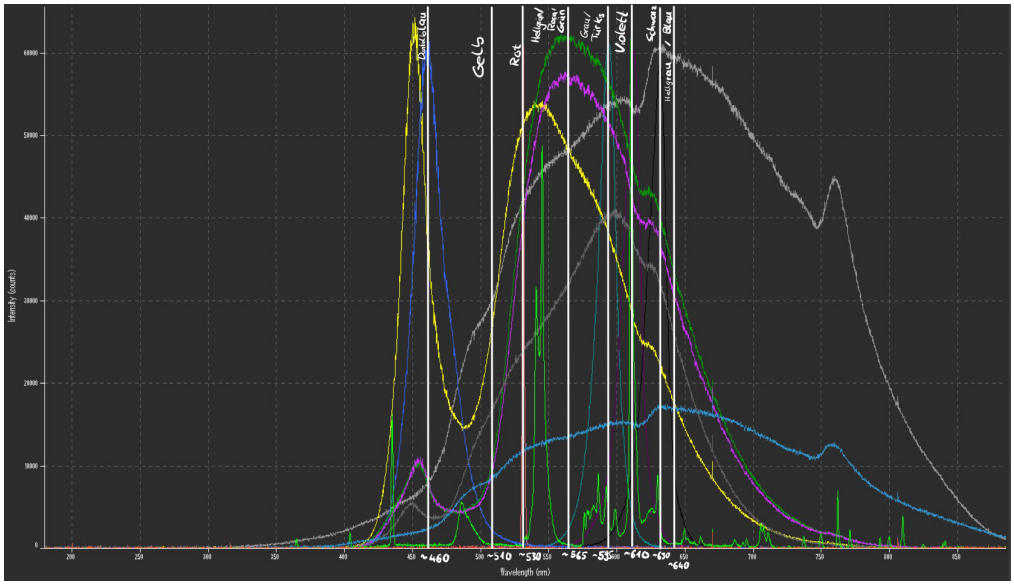


Abbildung 3: Spektren der Lichtquellen in Tab. 0

Erklärung der Spektren

Wir können nun die betrachteten Leuchtmittel einteilen in die in der Einleitung behandelten Strahlerklassen:

Klasse	Spektrenform	Leuchtmittel
Temperaturstrahler	Kontinuierlich	Glühlampe und Halogenlampe
Nichttemperaturstrahler mit genau einem Energieübergang	genau ein Peak	LED1-7 und LASER
Nichttemperaturstrahler mit mehreren Energieübergängen	mehrere diskrete Peaks	Energiesparlampe
Nichttemperaturstrahler mit floureszierendem Überzug	Kontinuierlich	LED-Birne

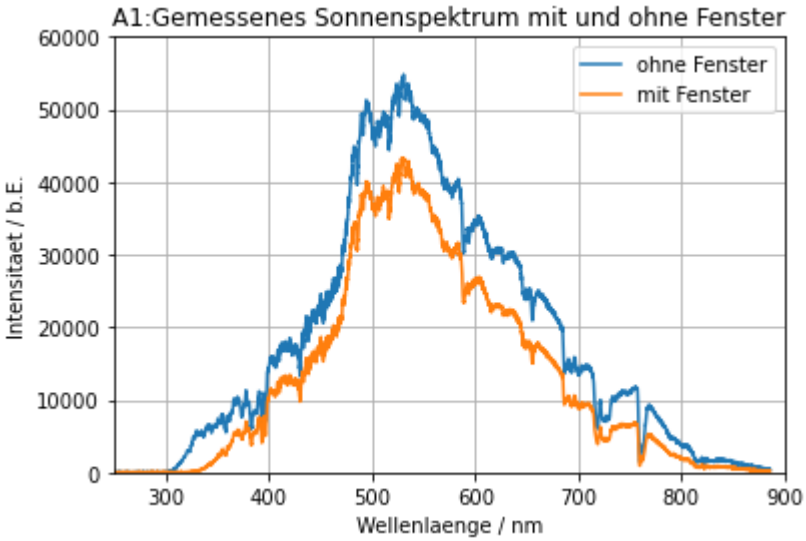
Sonnenlichtsprektrum

Wir betrachten das gemessenen Sonnenspektrum einmal direkt und einmal durch ein Fenster im nachfolgenden Diagramm A1.

In [2]:
```
plt.plot(lamb_og, inten_og, label='ohne Fenster')
plt.plot(lamb_mg, inten_mg, label='mit Fenster')
plt.title('A1:Gemessenes Sonnenspektrum mit und ohne Fenster')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.legend()
```

```
plt.grid()
plt.ylim((0, 60000))
plt.xlim((250, 900))
#plt.savefig("figures/Himmel_m_o_G.pdf", format="pdf")
```

Out[2]: (250.0, 900.0)



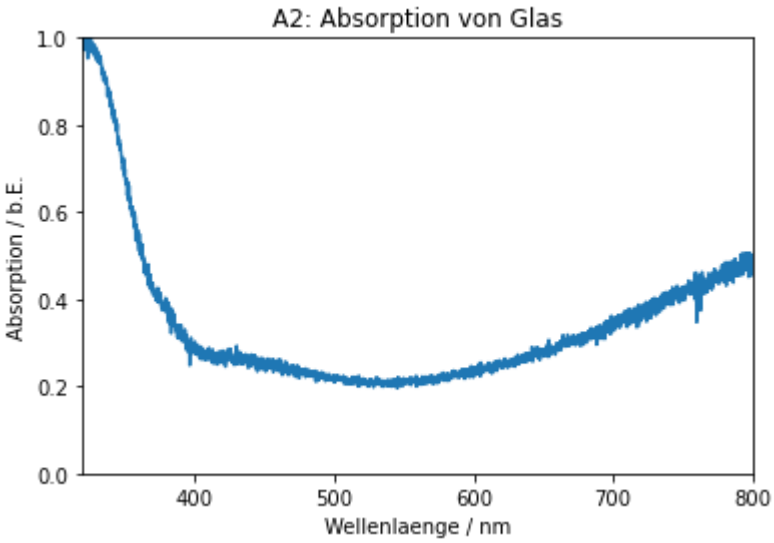
Absorptionsspektrum von Fensterglas

Wir errechen durch den vergleich der beiden Spektren ein Absorptionssprektrum des Fensterglases (Diag. A2). Die Absorption berechenet sich gemäß:

$$A = \frac{\text{Intensität mit Fenster}}{\text{Intensität ohne Fenster}} \tag{7}$$

In [3]:
```
A = 1 - inten_mg / inten_og
plt.plot(lamb_mg, A)
plt.title('A2: Absorption von Glas')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Absorption / b.E.')
plt.ylim((0, 1))
plt.xlim((320, 800))
#plt.savefig("figures/Absorption_Glas.pdf", format="pdf")
```

Out[3]: (320.0, 800.0)



Fraunhoferlinien

Wir nehmen nun das unverfälschte Sonnelichtspektrum auf und stellen es Interaktiv dar, um darin auffällige Dips zu bestimmen. Diese lassen wir anschließend in das Diagramm A3 als Linien zeichen. Zudem nehmen wir diese Tabellarisch mit Fehler auf.

In [4]:
```
%%html
<style>
div.jupyter-widgets.widget-label {display: none;}
</style>
```

In [24]:
```
%matplotlib notebook
plt.plot(lamb_og, inten_og)
plt.title('A3: Sonnenspektrum')
plt.xlabel('Wellenlänge / nm')
plt.ylabel('Intensität / b.E.')
plt.ylim((0, 60000))
plt.xlim((350, 800))
plt.grid()


#Importiere abgelesene Peaks zur Darstellung
fraunhofer_exp = pd.read_csv("fraunhofer_exp.csv")

err=(fraunhofer_exp["wavel"]-fraunhofer_exp["Half-life"]).abs()
fraunhofer_exp = fraunhofer_exp.drop('Half-life', 1)
fraunhofer_exp["wavel"]=fraunhofer_exp["wavel"]
fraunhofer_exp["error"]=err
#print(fraunhofer_exp)
```

```
#fraunhofer = np.loadtxt('fraunhofer_linien.txt') #Ermittelte Wellenlän
plt.plot([fraunhofer_exp["wavel"][0], fraunhofer_exp["wavel"][0]], [0,
for l in fraunhofer_exp["wavel"]:
    plt.plot([l, l], [0, 60000], color = "black", label = '_gemessene Di
#plt.savefig("figures/Fraunhofer.pdf", format="pdf")
plt.legend()

display(Markdown("***Tabelle 2: gefundene Fraunhoferlinien***"))
display(Markdown(fraunhofer_exp.round(2).rename(columns={"wavel":"Welle
```

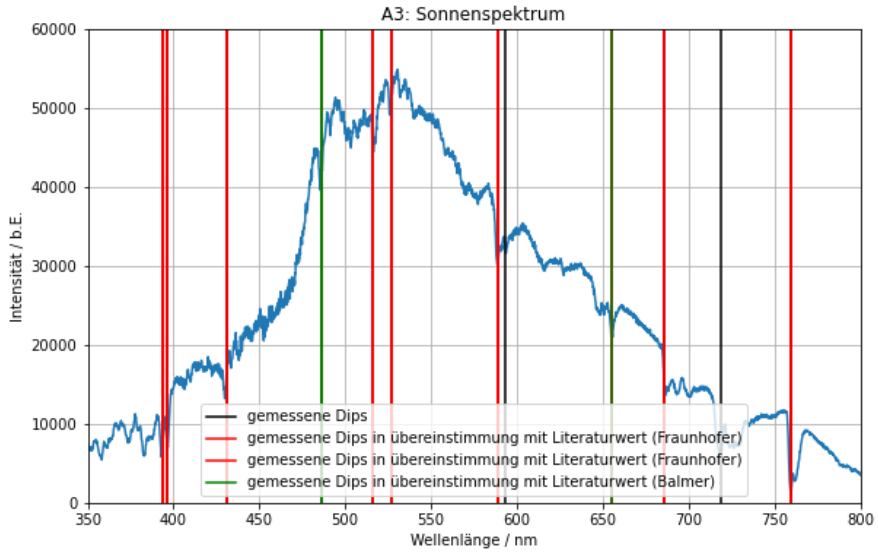


Tabelle 2: gefundene Fraunhoferlinien

	Wellenlänge[nm]	Fehler der Wellenlänge [nm]
0	393.3	1.3
1	396.5	1
2	430.7	2.1
3	486	1.5
4	515.9	3
5	526.7	1.6
6	588.6	1.2
7	593.6	0.81
8	655	2
9	685.5	7.3
10	718.5	4.2

	Wellenlänge[nm]	Fehler der Wellenlänge [nm]
11	759.2	6.3

Wir vergleichen die Werte mit den Literaturwerten aus dem Script, soweit zuordbar:

In [26]:
```
#Literaturvergleich
fraunhofer_lit = pd.read_csv("fraunhofer_lit.csv")

exp_l=pd.Series([],dtype="float64")
exp_l_error=pd.Series([],dtype="float64")
for n in range(0,len(fraunhofer_lit["element"])):
    for k in range(0,len(fraunhofer_exp["wavel"])):
        if abs(fraunhofer_lit.at[n,"lit_l"]-fraunhofer_exp.loc[k,"wavel
            exp_l[n]=fraunhofer_exp.loc[k,"wavel"]
            exp_l_error[n]=fraunhofer_exp.loc[k,"error"]
fraunhofer_lit["exp_l"]=exp_l
fraunhofer_lit["exp_l_error"]=exp_l_error


sigma=pd.Series([],dtype="float64")
for i in range(0,len(fraunhofer_lit["element"])):
    sigma[i]=abs((fraunhofer_lit.loc[i,"lit_l"]-fraunhofer_lit.loc[i,"e

fraunhofer_lit["sigma"]=sigma


#Zeichne gefundenen Linien in rot ein
plt.plot([fraunhofer_lit["exp_l"][0], fraunhofer_lit["exp_l"][0]], [0,
for l in fraunhofer_lit["exp_l"]:
   plt.plot([l, l], [0, 60000], color = "red", label = '_gemessene Dips
plt.legend()

#print(fraunhofer_lit.round(2))
#fraunhofer_lit.to_csv("out_fraunhofer_lit.csv")

display(Markdown("***Tabelle 3: Literaturwerte der Fraunhoferlinien im
display(Markdown(fraunhofer_lit.round(2).rename(columns={"lit_l":"Liter
```

Tabelle 3: Literaturwerte der Fraunhoferlinien im Vergleich mit Messwerten

	Literaturwert der Wellenlänge [nm]	Element	Experimentell ermittelte Wellenlänge [nm]	Fehler der Wellenlänge [nm]	Sigma-Abweichung
0	393.4	Ca	393.3	1.3	0.08
1	396.9	Ca	396.5	1	0.4
2	422.7	Ca	nan	nan	nan

	Literaturwert der Wellenlänge [nm]	Element	Experimentell ermittelte Wellenlänge [nm]	Fehler der Wellenlänge [nm]	Sigma-Abweichung
3	430.8	Ca/Fe	430.7	2.1	0.05
4	486.1	H	486	1.5	0.07
5	518.4	Ma	515.9	3	0.83
6	527	Ca/Fe	526.7	1.6	0.19
7	587.6	He	588.6	1.2	0.83
8	589	Na	588.6	1.2	0.33
9	589.6	Na	588.6	1.2	0.83
10	656.3	H	655	2	0.65
11	686.7	O	685.5	7.3	0.16
12	759.4	O	759.2	6.3	0.03

Wir sehen, dass der Messwert um 588.6nm mehrfach zugeordnet wurde, da es zu den in der Umgebung liegenden Fraunhoferlinien keine besseren Entsprechungen gibt, ist unklar woher sich dieser Messwert tatsächlich ergibt, da alle drei Werte weniger als ein Sigma verschoben sind, handelt es sich vermutlich um eine Überlagerung der drei Linien. In A3 wurden nun alle zugeordneten Werte rot eingefärbt. Werte 10 und 7 aus Tabelle 2 konnnten nicht zugeordet werden.

Balmer-Serie

Wir vergleichen die Balmer-Serie mit den gemessenen Werten, soweit zuordbar, die Literaturwerte wurden dem Script entnommen:

In [27]:
```
balmer_lit = pd.read_csv("balmer_lit.csv")

exp_l=pd.Series([],dtype="float64")
exp_l_error=pd.Series([],dtype="float64")
for n in range(0,len(balmer_lit["lit_l"])):
    for k in range(0,len(fraunhofer_exp["wavel"])):
        if abs(balmer_lit.at[n,"lit_l"]-fraunhofer_exp.loc[k,"wavel"])<
            exp_l[n]=fraunhofer_exp.loc[k,"wavel"]
            exp_l_error[n]=fraunhofer_exp.loc[k,"error"]
balmer_lit["exp_l"]=exp_l
balmer_lit["exp_l_error"]=exp_l_error


sigma=pd.Series([],dtype="float64")
for i in range(0,len(balmer_lit["lit_l"])):
```

```
    sigma[i]=abs((balmer_lit.loc[i,"lit_l"]-balmer_lit.loc[i,"exp_l"])/

balmer_lit["sigma"]=sigma


#Zeichne gefundenen Linien in green ein
plt.plot([ balmer_lit["exp_l"][1],  balmer_lit["exp_l"][1]], [0, 60000]
for l in balmer_lit["exp_l"]:
   plt.plot([l, l], [0, 60000], color = "green", label = '_gemessene Di
plt.legend()
#print(balmer_lit.round(2))
#balmer_lit.to_csv("out_balmer_lit.csv")
display(Markdown("***Tabelle 4: Literaturwerte der Balmerserie im Vergl
display(Markdown(balmer_lit.round(2).rename(columns={"lit_l":"Literatur
```

Tabelle 4: Literaturwerte der Balmerserie im Vergleich mit Messwerten

	Literaturwert der Wellenlänge [nm]	Experimentell ermittelte Wellenlänge [nm]	Fehler der Wellenlänge [nm]	Sigma-Abweichung
0	656.3	655	2	0.65
1	486.1	486	1.5	0.07
2	434	nan	nan	nan
3	410.1	nan	nan	nan

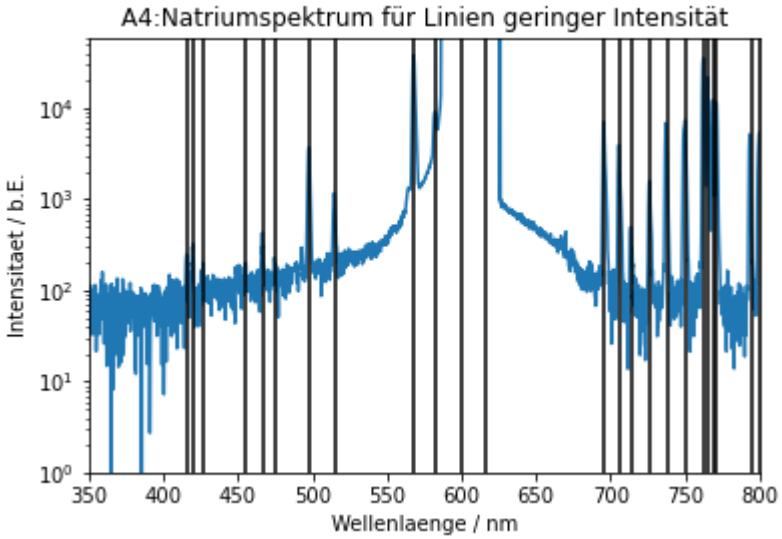
Natrium-Dampflampe

Wir lesen zunächst aus Messungen bei unterschiedlicher Intensität und passenden Wellenlängenbereichen peaks aus den Diagrammen A4-6 ab und stellen anschließend eben diese in diesen dar:

In [28]:
```
#Importiere abgelesene Peaks zur Darstellung
na_lines = pd.read_csv("na1_lines.csv")
err=(na_lines["wavel"]-na_lines["Half-life"]).abs()
na_lines = na_lines.drop('Half-life', 1)
na_lines["wavel"]=na_lines["wavel"]
na_lines["error"]=err
```

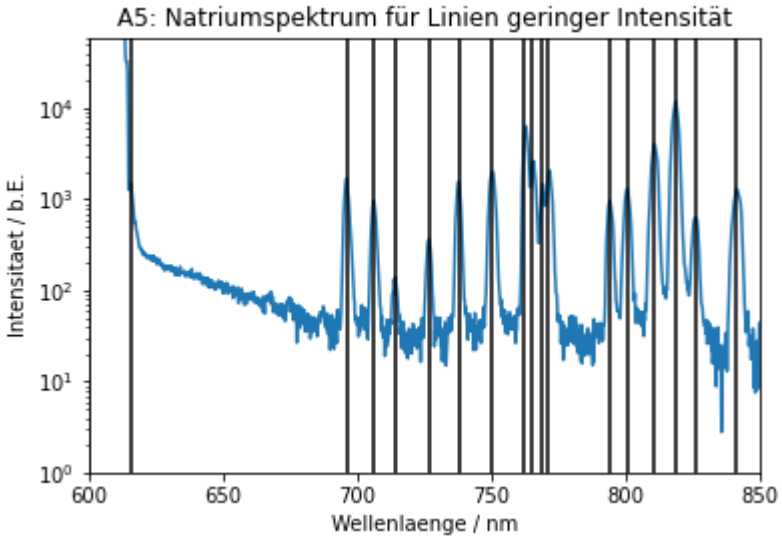
In [29]:
```
%matplotlib notebook
lamb_na1, inten_na1 = np.loadtxt('na1.txt', skiprows=17, converters= {0
plt.plot(lamb_na1, inten_na1)
plt.title('A4:Natriumspektrum für Linien geringer Intensität')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.yscale('log')
plt.ylim((1,60000))
plt.xlim((350, 800))
```

```
for l in na_lines["wavel"]:
    plt.plot([l, l], [1, 60000], color = "black", label = '{0}nm'.forma
```

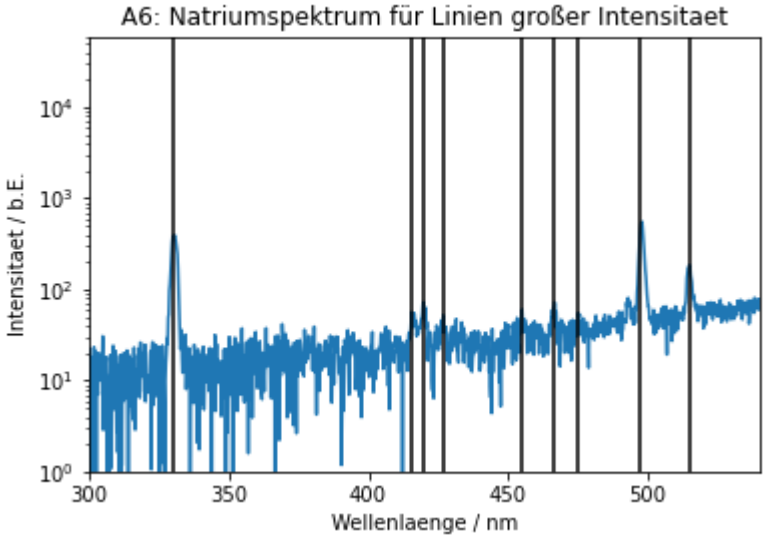


Hier sind auch die bei geringerer Aufnahmeintensität abgelesenen Linien eingezeichnet, daher sind hier noch nicht zwangsläufig alle Peaks gut sichtbar.

In [30]:
```
%matplotlib notebook
lamb_na1, inten_na1 = np.loadtxt('na2.txt', skiprows=17, converters= {0
plt.plot(lamb_na1, inten_na1)
plt.title('A5: Natriumspektrum für Linien geringer Intensität')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.yscale('log')
plt.ylim((1,60000))
plt.xlim((600, 850))
for l in na_lines["wavel"]:
    plt.plot([l, l], [1, 60000], color = "black", label = '{0}nm'.forma
```



In [31]:
```
%matplotlib notebook
lamb_na2, inten_na2 = np.loadtxt('na2.txt', skiprows=17, converters= {0
plt.plot(lamb_na2, inten_na2)
plt.title('A6: Natriumspektrum für Linien großer Intensitaet')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.yscale('log')
plt.ylim((1,60000))
plt.xlim((300,540))
for l in na_lines["wavel"]:
    plt.plot([l, l], [1, 60000], color = "black", label = '{0}nm'.forma
```



Wir lesen die Peaks aus den Diagrammen ab, wobei wir den Fehler über die Halbwertsbreite abgelesen haben. Wir erhalten:

In [32]:
```
na_lines = pd.read_csv("na1_lines.csv")
err=(na_lines["wavel"]-na_lines["Half-life"]).abs()
na_lines = na_lines.drop('Half-life', 1)
na_lines["wavel"]=na_lines["wavel"]
na_lines["error"]=err

#print(na_lines)
#na_lines.to_csv("out_na_lines.csv")
display(Markdown("***Tabelle 5: gefundene Emissionslinien der Na-Dampfl
display(Markdown(na_lines.rename(columns={"wavel":"Wellenlänge[nm]","er
```

Tabelle 5: gefundene Emissionslinien der Na-Dampflame

	Wellenlänge[nm]	Fehler der Wellenlänge [nm]
0	330.1	1.8
1	415.5	1
2	419.6	2.2
3	426.7	1.1
4	454.5	2
5	466.3	1.3
6	475.1	0.7
7	497.4	1.8
8	514.9	1.1
9	567.8	1.8
10	582	1.1
11	600	8
12	615.5	1.8
13	696	2.2
14	706.2	1.7
15	714	1.3
16	726.7	1.1
17	737.8	1.7
18	749.9	1.9
19	762.2	1.5
20	765	1.7
21	768.7	0.8

	Wellenlänge[nm]	Fehler der Wellenlänge [nm]
22	771.3	1.1
23	794.3	0.7
24	800.8	1.9
25	810.8	2.5
26	818.9	2.1
27	826.1	2.1
28	841	3

Linien der 1.Nebenserie

Wir berechnen E_{3p} aus dem Peak bei 818.9nm mit $m = 3$, mit Fehler per Gauß, aus:

$$h \cdot \frac{c}{\lambda_m} = E_{\mathrm{Ry}}[\mathrm{eV}]/m^2 - E_{3p}[\mathrm{eV}] \tag{8}$$

In [33]:
```
E_Ry=-13.605
hc=1.2398E3
E_3p=(E_Ry/(3**2))-hc/818.9
Delta_E_3p=(hc/818.9**2)*2.1
print("E_3p= " + str(round(E_3p, 3)))
print("Delta_E_3p= "+ str(round(Delta_E_3p, 3)))
```

```
E_3p= -3.026
Delta_E_3p= 0.004
```

Wir können nun gemäß der Formel

$$\lambda_m[\mathrm{nm}] \approx \frac{1,2398 \times 10^3[\mathrm{nm\ eV}]}{-13,605\mathrm{eV/m^2} - E_{3p}[\mathrm{eV}]} \tag{9}$$

eine Vorhersage treffen. Wir erstellen eine Tabelle über die verscheidenen Wellenlängen und ordnen, wenn möglich, experimentell ermittelte Wellenlängen zu:

In [34]:
```
ms=pd.Series(range(3,13))
na_serie1 = pd.DataFrame(
    {
        "serie":pd.Series(["1.nS","1.nS","1.nS","1.nS","1.nS","1.nS","1
        "m": ms,
        "theoretical_l":ms.apply(lambda x:1.2398E3/(-13.605/x**2-E_3p))
        "theo_l_error":ms.apply(lambda x:1.2398E3/((-13.605/x**2-E_3p)*
    }
```

```
)

exp_l=pd.Series([],dtype="float64")
exp_l_error=pd.Series([],dtype="float64")
for n in range(0,len(na_serie1["m"])):
    for k in range(0,len(na_lines["wavel"])):
        if abs(na_serie1.at[n,"theoretical_l"]-na_lines.loc[k,"wavel"])
            exp_l[n]=na_lines.loc[k,"wavel"]
            exp_l_error[n]=na_lines.loc[k,"error"]
na_serie1["exp_l"]=exp_l
na_serie1["exp_l_error"]=exp_l_error
na_serie1.to_csv("out_na_serie1.csv")

#print(na_serie1.round(1))
display(Markdown("***Tabelle 6: Theoretische Werte der Na-Emissionslini
display(Markdown(na_serie1.round(1).rename(columns={"theoretical_l":"Th
```

Tabelle 6: Theoretische Werte der Na-Emissionslinien der 1. Nebenserie im Vergleich mit Messwerten

	serie	m	Theoriewert der Wellenlänge [nm]	Fehler des Theoriewertes [nm]	Experimentell ermittelte Wellenlänge [nm]	Fehler der experimentell ermittelten Wellenlänge [nm]
0	1.nS	3	818.9	2.1	818.9	2.1
1	1.nS	4	569.9	1	567.8	1.8
2	1.nS	5	499.6	0.8	497.4	1.8
3	1.nS	6	468.2	0.7	466.3	1.3
4	1.nS	7	451.2	0.6	nan	nan
5	1.nS	8	440.7	0.6	nan	nan
6	1.nS	9	433.8	0.6	nan	nan
7	1.nS	10	429.1	0.6	nan	nan
8	1.nS	11	425.6	0.6	426.7	1.1
9	1.nS	12	423	0.6	nan	nan

2. Nebenserie

Wir bestimmen E_{3s} aus dem Peak bei 582nm per:

$$E_{3s} = E_{3p} - 1,2398 \cdot 10^3 [\text{nm eV}]/\lambda \tag{10}$$

und den Fehler per Gauß:

$$\Delta E_{3s} = \sqrt{\Delta E_{3p}^2 + (1,2398 \cdot 10^3 [\text{nm eV}]/\lambda^2 \Delta\lambda)^2} \tag{11}$$

In [35]:
```
E_3s=E_3p-1.2398E3/582
Delta_E_3s=np.sqrt(Delta_E_3p**2+1.2398E3*1.1/(582)**2)
print("E_3s= " + str(round(E_3s,2)))
print("Delta_E_3s= "+ str(round(Delta_E_3s,2)))
```

```
E_3s= -5.16
Delta_E_3s= 0.06
```

Wir bestimmen den Korrekturfaktor Δ_s:

$$E_3 s = -13.605eV/(3-\Delta_s)^2 \tag{12}$$

$$\therefore \quad \Delta_s = 3 - \sqrt{-13.605eV/E_{3s}} \tag{13}$$

$$\therefore \quad \Delta\Delta_s = \sqrt{-13.605eV/E_{3s}^{3/2}\Delta E_{3s}} \tag{14}$$

In [36]:
```
korr_s=3-np.sqrt(-13.605/E_3s)
Delta_korr_s= np.sqrt(13.605)/(-E_3s)**(3/2)*Delta_E_3s
print("korr_s= " + str(round(korr_s,2)))
print("Delta_korr_s= "+ str(round(Delta_korr_s,2)))
```

```
korr_s= 1.38
Delta_korr_s= 0.02
```

Wir berechnen wieder die Serie mit der folgenden Formel (und entsprechendem Fehler nach Gauß) und vergleichen wie oben mit den experimentellen Werten:

$$\lambda_m = \frac{1,2398 \cdot 10^3 [\text{nm eV}]}{-13.605eV/(m-\Delta_s)^2 - E_{3p}}$$

$$\Delta\lambda_m = \sqrt{(\frac{1,2398 \cdot 10^3 [\text{nm eV}]}{(-13.605eV/(m-\Delta_s)^2 - E_{3p})^2} \frac{2}{(m-\Delta_s)^3}\Delta\Delta_s)^2 + (\frac{}{(-13.}$$

In [37]:
```
ms=pd.Series(range(4,10))
na_serie2 = pd.DataFrame(
    {
        "serie": pd.Series(["2.nS","2.nS","2.nS","2.nS","2.nS","2.nS"])
        "m": ms,
        "theoretical_l":ms.apply(lambda x:1.2398E3/(-13.605/(x-korr_s)*
        "theo_l_error":ms.apply(lambda x:1.2398E3/((-13.605/(x-korr_s)*
    }
)
exp_l=pd.Series([],dtype="float64")
exp_l_error=pd.Series([],dtype="float64")

for n in range(0,len(na_serie2["m"])):
    for k in range(0,len(na_lines["wavel"])):
```

```
        if abs(na_serie2.at[n,"theoretical_l"]-na_lines.loc[k,"wavel"])
            exp_l[n]=na_lines.loc[k,"wavel"]
            exp_l_error[n]=na_lines.loc[k,"error"]

na_serie2["exp_l"]=exp_l
na_serie2["exp_l_error"]=exp_l_error


#print(na_serie2.round(1))
#na_serie2.to_csv("out_na_serie2.csv")

display(Markdown("***Tabelle 7: Theoretische Werte der Na-Emissionslini
display(Markdown(na_serie2.round(1).rename(columns={"theoretical_l":"Th
```

Tabelle 7: Theoretische Werte der Na-Emissionslinien der 2. Nebenserie im Vergleich mit Messwerten

	serie	m	Theoriewert der Wellenlänge [nm]	Fehler des Theoriewertes [nm]	Experimentell ermittelte Wellenlänge [nm]	Fehler der experimentell ermittelten Wellenlänge [nm]
0	2.nS	4	1180.4	5	nan	nan
1	2.nS	5	623	1.2	615.5	1.8
2	2.nS	6	518.9	0.8	514.9	1.1
3	2.nS	7	477.7	0.7	475.1	0.7
4	2.nS	8	456.5	0.7	454.5	2
5	2.nS	9	444.1	0.6	nan	nan

Hauptserie

Wir bestimmen Δ_p mit Fehler nach Gauß:

$$\Delta_p = 3 - \sqrt{-13.605eV/E_{3p}} \tag{17}$$

$$\Delta\Delta_p = \sqrt{-13.605eV/E_{3p}^{3/2}\Delta E_{3p}} \tag{18}$$

In [38]:
```
korr_p=3-np.sqrt(-13.605/E_3p)
Delta_korr_p= np.sqrt(13.605)/(-E_3s)**(3/2)*Delta_E_3p
print("korr_p= " + str(korr_p.round(3)))
print("Delta_korr_p= "+ str(Delta_korr_p.round(4)))
```

```
korr_p= 0.879
Delta_korr_p= 0.0012
```

Wir berechnen wieder (vollkommen analog zu oben) die Serie mit der folgenden

Formel (und entsprechendem Fehler nach Gauß) und vergleichen wie oben mit den experimentellen Werten:

$$\lambda_m = \frac{1,2398 \cdot 10^3 [\text{nm eV}]}{-13.605eV/(m-\Delta_p)^2 - E_{3s}}$$

$$\Delta\lambda_m = \sqrt{(\frac{1,2398 \cdot 10^3 [\text{nm eV}]}{(-13.605eV/(m-\Delta_p)^2 - E_{3s})^2} \frac{2}{(m-\Delta_p)^3} \Delta\Delta_p)^2 + (\frac{}{(-13.}$$

In [39]:
```
ms=pd.Series(range(4,6))
na_serie3 = pd.DataFrame(
    {
        "serie":pd.Series(["Hauptserie","Hauptserie"]).repeat(1),
        "m": ms,
        "theoretical_l":ms.apply(lambda x:1.2398E3/(-13.605/(x-korr_p)*
        "theo_l_error":ms.apply(lambda x:1.2398E3/((-13.605/(x-korr_p)*
    }
)

exp_l=pd.Series([],dtype="float64")
exp_l_error=pd.Series([],dtype="float64")
for n in range(0,len(na_serie3["m"])):
    for k in range(0,len(na_lines["wavel"])):
        if abs(na_serie3.at[n,"theoretical_l"]-na_lines.loc[k,"wavel"])
            exp_l[n]=na_lines.loc[k,"wavel"]
            exp_l_error[n]=na_lines.loc[k,"error"]
na_serie3["exp_l"]=exp_l
na_serie3["exp_l_error"]=exp_l_error

#print(na_serie3.round(1))
#na_serie3.to_csv("out_na_serie3.csv")
display(Markdown("***Tabelle 8: Theoretische Werte der Na-Emissionslini
display(Markdown(na_serie3.round(1).rename(columns={"theoretical_l":"Th
```

Tabelle 8: Theoretische Werte der Na-Emissionslinien der Haupserie im Vergleich mit Messwerten

	serie	m	Theoriewert der Wellenlänge [nm]	Fehler des Theoriewertes [nm]	Experimentell ermittelte Wellenlänge [nm]	Fehler der experimentell ermittelten Wellenlänge [nm]
0	Hauptserie	4	329.8	5.6	330.1	1.8
1	Hauptserie	5	284.7	4.2	nan	nan

Wir fassen die zuordbaren Linien zusammen und betrachten die Abweichungen:

In [40]:
```
na_series=pd.concat([na_serie1,na_serie2,na_serie3],ignore_index=True,
```

```
sigma=pd.Series([],dtype="float64")
for i in range(0,len(na_series["m"])):
    sigma[i]=abs(na_series.loc[i,"theoretical_l"]-na_series.loc[i,"exp_

na_series["sigma"]=sigma

def getRidOfNaNs(serie):
    counts=serie.count(1)
    for i in range(0,len(serie["m"])):
        if counts.loc[i]<5:
                serie=serie.drop(labels=i, axis=0)
    return serie
na_series_mo=getRidOfNaNs(na_series)
#print(na_series.round(2))
#na_series.to_csv("out_na_series_overview.csv")
display(Markdown("***Tabelle 9: Gemessene Werte der Na-Emissionslinien
display(Markdown(na_series_mo.round(1).rename(columns={"theoretical_l":
```

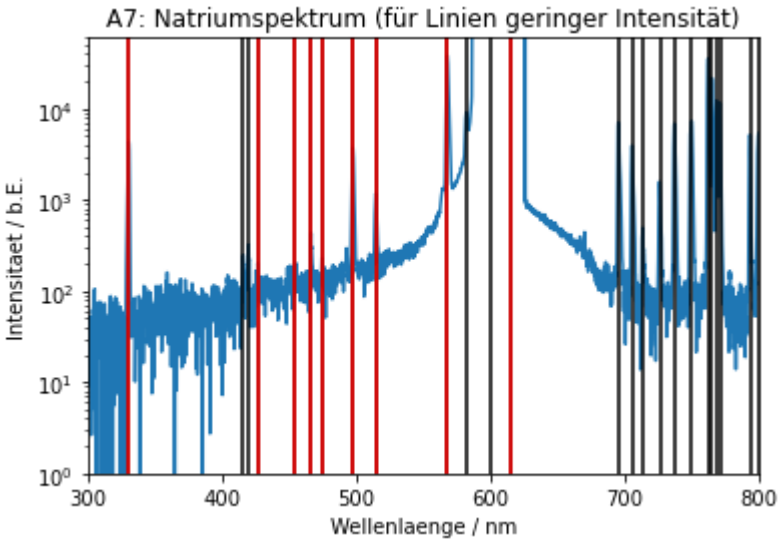
Tabelle 9: Gemessene Werte der Na-Emissionslinien im Vergleich zu vorhergesagten Werten

	serie	m	Theoriewert der Wellenlänge [nm]	Fehler des Theoriewertes [nm]	Experimentell ermittelte Wellenlänge [nm]	Fehler der experimentell ermittelten Wellenlänge [nm]	sigma
0	1.nS	3	818.9	2.1	818.9	2.1	0
1	1.nS	4	569.9	1	567.8	1.8	1
2	1.nS	5	499.6	0.8	497.4	1.8	1.1
3	1.nS	6	468.2	0.7	466.3	1.3	1.3
8	1.nS	11	425.6	0.6	426.7	1.1	0.9
11	2.nS	5	623	1.2	615.5	1.8	3.4
12	2.nS	6	518.9	0.8	514.9	1.1	2.9
13	2.nS	7	477.7	0.7	475.1	0.7	2.6
14	2.nS	8	456.5	0.7	454.5	2	1
16	Hauptserie	4	329.8	5.6	330.1	1.8	0

In Diag. A7 sind alle gefundenen Peaks dargestellt, diejenigen, welche sich zuordnen ließen, sind rot eingefärbt.

In [41]:
```
%matplotlib inline
lamb_na1, inten_na1 = np.loadtxt('na1.txt', skiprows=17, converters= {0
```

```
plt.plot(lamb_na1, inten_na1)
plt.title('A7: Natriumspektrum (für Linien geringer Intensität)')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.yscale('log')
plt.ylim((1,60000))
plt.xlim((300, 800))
for l in na_lines["wavel"]:
    plt.plot([l, l], [1, 60000], color = "black", label = '{0}nm'.forma
for l in na_series["exp_l"]:
    plt.plot([l, l], [1, 60000], color = "red", label = '{0}nm'.format(
```



Bestimmung der Serienenergien und der Korrekturfaktoren

Wir wollen nun umgekehrt die gemessenen Wellelängen der Serien (vgl. Tab. 7) nutzen um die Rydbergenergie E_{Ry},R_{3p} und die Korrekturtherme Δ_d und Δ_p zu bestimmen.

1. Nebenserie

Wir tragen die gefundenen Werte der 1. Nebenserie in ein Diagram ein und fitten eine Funktion der Form

$$\lambda_m = \frac{1,2398 \cdot 10^3 [\text{nm eV}]}{E_{Ry}/(m-\Delta_d)^2 - E_{3p}} \tag{21}$$

wobei, wir E_{Ry},Δ_d und E_{3p} als freie Parameter wählen, nach welchen wir den Fit optimieren. Wir errechenen dann die Abweichungen zu den im vorherigem Abschnitt bestimmten Werten. Außerdem berechnen wir die χ^2-Summe:

$$\chi^2 = \sum_i^N \left(\frac{\text{Funktionswert }_i - \text{ Messwert }_i)}{\text{Fehler }_i} \right)^2 \tag{22}$$

und

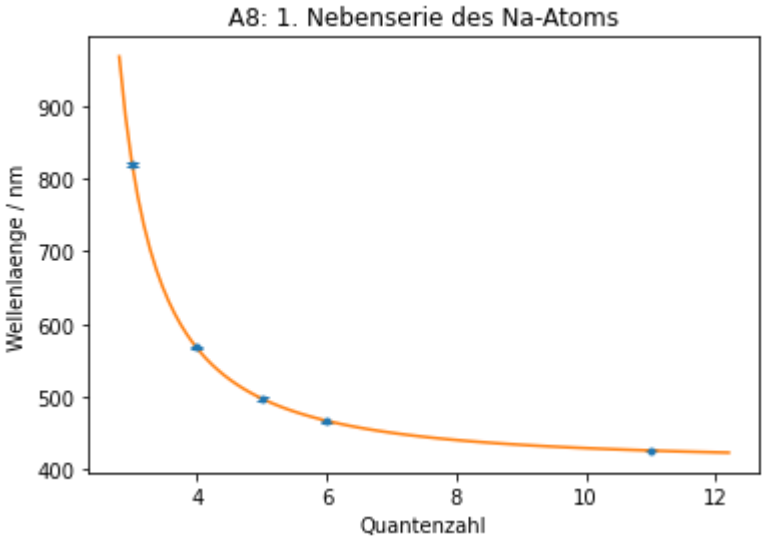
$$\chi^2_{red} = \chi^2/\text{Freiheitsgrade des Fits}$$

sowie die Fitwahrscheinlichkiet, also, dass bei einer wiederholten Messung die χ^2 -Summe größer würde oder gleich bliebe.

In [44]:
```
%matplotlib inline

from scipy.optimize import curve_fit
from scipy.stats import chi2

def fit_func(m,E_Ry,E_3p,D):
    return hc/(E_Ry/((m-D)**2)-E_3p)

def makeFitDiagram(na_serie,title,ds):
    ns =np.array([na_serie["m"],na_serie["exp_l"],na_serie["exp_l_error

    popt, pcov = curve_fit(fit_func, ns[0], ns[1], sigma = ns[2] ,p0 =

    print("E_Ry = {0:4.1f} eV".format(popt[0]), ", Standardfehler = {0:
    print("E_3p = {0:4.3f} eV".format(popt[1]), ", Standardfehler = {0:
    print("Delta_"+ ds +" = {0:4.2f}".format(popt[2]), ", Standardfehle

    chi2_=np.sum((fit_func(ns[0],*popt)-ns[1])**2/ns[2]**2)
    dof=len(ns[0])-3
    chi2_red=chi2_/dof
    print("chi2=", chi2_)
    print("chi2_red=",chi2_red)

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

    plt.errorbar(ns[0],ns[1],ns[2], fmt=".", capsize = 3)
    plt.xlabel('Quantenzahl')
    plt.ylabel('Wellenlaenge / nm')
    plt.title(title)
    x=np.linspace(2.8,12.2, 100)
    _=plt.plot(x, fit_func(x,*popt))

na_serie1_mo=getRidOfNaNs(na_serie1)
makeFitDiagram(na_serie1_mo,'A8: 1. Nebenserie des Na-Atoms',"d")
```

```
E_Ry = -12.4 eV , Standardfehler =  0.4 sigma = 3.0
E_3p = -3.013 eV , Standardfehler = 0.007 sigma = 1.6
Delta_d = 0.12 , Standardfehler = 0.04
chi2= 0.9286701146082879
```

```
chi2_red= 0.46433505730414393
Wahrscheinlichkeit: 63.0 %
```

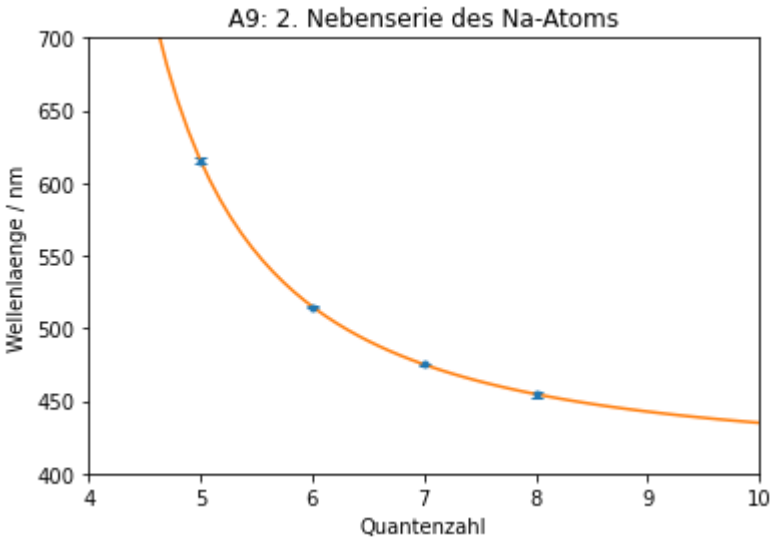


2. Nebenserie

Wir gehen hier analog zur 1. Nebenserie vor, unsere anzufittene Funktion ändert sich nur in der Benennung des Korrekturfaktors $\Delta_d \rightarrow \Delta_s$.

In [45]:
```
%matplotlib inline
na_serie2_mo=getRidOfNaNs(na_serie2)
makeFitDiagram(na_serie2_mo,'A9: 2. Nebenserie des Na-Atoms',"s",)
plt.xlim((4,10))
_=plt.ylim((400,700))
```

```
E_Ry = -13.1 eV , Standardfehler =  0.2 sigma = 2.6
E_3p = -3.028 eV , Standardfehler = 0.003 sigma = 0.5
Delta_s = 1.41 , Standardfehler = 0.02
chi2= 0.007209163988172628
chi2_red= 0.007209163988172628
Wahrscheinlichkeit: 93.0 %
```



Diskussion

In diesem Versuch haben wir zunächst eine qualtitative Berachtung einiger Lichtquellen durchgeführt, welche uns erlaubt hat diese (teilweise mithilfe von Vorwissen) zu klassifizieren. Wir haben anschließend ein Sonnenspektrum untersucht, wobei wir alle, bis auf eine, Frauenhoferlinien beobachten konnten. Die erwarteten Linien von Helium und Natrium sind allerdings zu einem Peak "verschmolzen", sodass wir hier keine genaue zuordnung treffen konnten. Wir konnten zwei Glieder der Balmerserie finden. Alle gemessenen Fraunhoferlinien und Glieder der Balmerserie befinden sich in einer 1-σ-Umgebung zum Literaturwert, stimmen also sehr gut überein. Die anderen Glider waren nicht sichtbar. Eventuell wären diese im Verhältnis zum Rauschen stärker sichtbar, würde über mehr Messungen gemittelt. Im weiteren haben wir das durch ein Fenster transmittierte Licht im Vergleich zum direkten Sonnensprektrum betrachtet und gefunden, dass sichtbares Licht kaum, UV-Licht aber sehr stark absorbiert wird (Diag. A2). Schließlich wurden die Emissionslinien einer Natriumdampflampe vermessen und einem teilweise aus den Messwerten berechneten Modell verglichen, sowie anschließend die zugehörigen Korrekturtherme und Energien für 1. und 2. Nebenserie mittles einer gefitteten Funktion bestimmt. Dabei konnten wir viele von der Vorhersage gemachten Emissionslinien tatsächlich finden, aber nicht alle. Eventuell ließe sich durch mehr Messungen das Rauschen des Signals reduzieren, sodass wir auch kleiner Peaks besser sehen könnten. Diejenigen Peaks, welche wir zuordnen konnten befinden sich bei der 1. Nebenserie alle in einer 2-σ-Umgebung zur Vorhersage, unterscheiden sich also nicht relevant. Bei der zweiten Nebenserie befinden sich

die Messwerte in einer 3,5-σ-Umgebung. Dies ist eine signifikante Abweichung und nicht zufriedenstellend, jedoch wurden mindestens 4 Messwerte für den späteren Fit benötigt, daher wurde die zuordnung trotzdem getroffen, aber die Vorhersage stimmt hier nicht gut mit den Messwerten überein. Optimalerweise würde man hier nocheinmal genauer (mehr Messungen, reduzierung des Rauschens) nachmessen. Wir finden beim fitten der Werte zur 1. Nebenserie nicht signifikant andere (3-σ-Umgebung) Werte für E_{Ry},R_{3p} und Δ_d. Die Fitwahrscheinlichkeit ist mit 63% gut, es wurde ohnehin keine ganz genaue Übereinstimmung erwartet, da Δ_d auch noch eine kleine m-Abhängigkeit besitzt, welche hier vernachlässigt wurde. Bei der 2. Nebenserie bietet sich ein ähnliches Bild, hier liegen die berechenten Werte sogar noch ein Wenig näher zusammen (2.6-σ-Umgebung) und die Fitwahrscheinlichkeit ist mit 93% noch ein wenig besser, wobei hierzu auch beigetragen haben dürfte, dass es insgesamt nur vier Messwerte gibt, an die sich die Funktion besser anfitten lässt. Abschließend lässt sich bei diesem Versuch festhalten, dass es teilweise schwierig war gute Werte aus den Daten zu gewinnen und daher die Anzahl der verwertbaren Datenpunkte, insbesondere bei der zweiten Nebenserie, sehr gelitten hat. Teilweise wäre eine weitere Diskussion der Verwertbarkeit der verwendeten Datenpunkte sicher angebracht, diese würde jedoch optimaler Weise weitere Messungen, in welchen Rauschen durch mehr Messungen und eventuelles Ausschalten von Störquellen (Tageslicht durchs Fenster, andere Lichter) minimiert würde, getragen werdem und ist daher hier nicht abschließend Durchführbar.