lichtquellen natrium teil-checkpoint

March 8, 2019

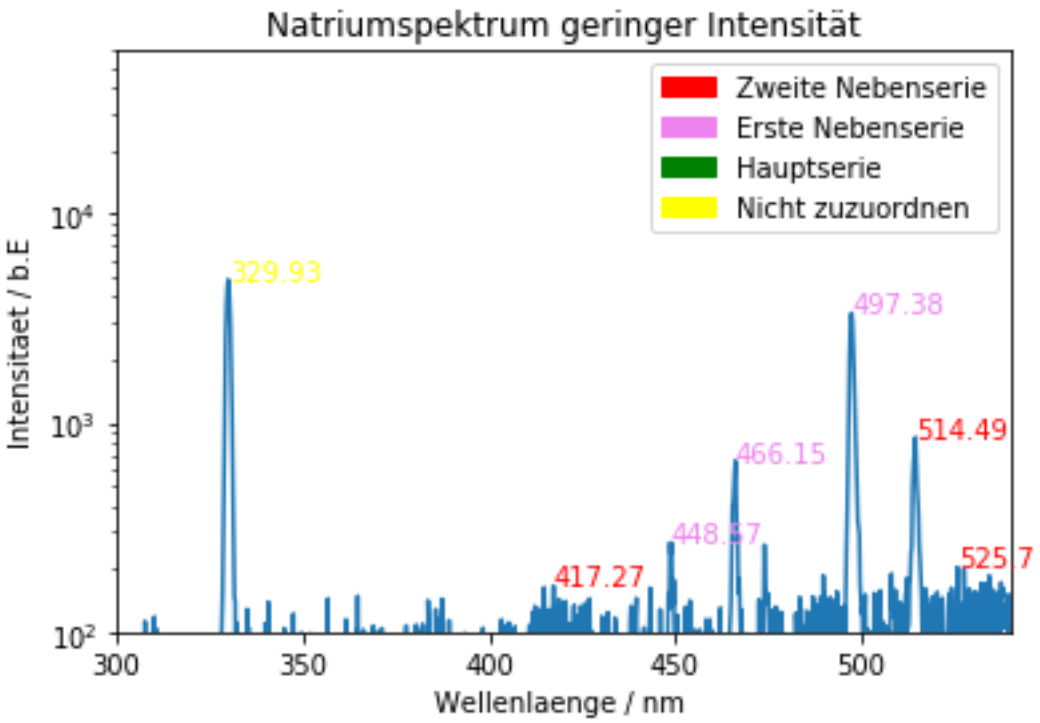
In [12]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from scipy import signal
import matplotlib.patches as mpatches

E_3p=-(1.2398e3/(818.75)+13.605/3**2)
print(E_3p)
E_3s=E_3p-1.2398e3/589
delta_s=-np.sqrt(-13.605/E_3s)+3
delta_p=np.sqrt(-13.605/E_3p)+3
nseries1=[1.2398e3/(-13.605/m**2-E_3p) for m in range(3,20)]
nseries2=[1.2398e3/(-13.605/(m-delta_s)**2-E_3p) for m in range(4,20)]
mainseries=[1.2398e3/(-13.605/(m-delta_p)**2-E_3s) for m in range(7,20)]
def printseries(x):
    for i, n in enumerate(x):
        print("%i: %.2f"%(i+1,n))
print("Erste Nebenserie")
printseries(nseries1)
print("Zweite Nebenserie")
printseries(nseries2)
print("hauptserie")
printseries(mainseries)
def find_series(x):
    ##return 0 for first neben series, 1 for second, 2 for main series
    d_n1=np.min((nseries1-x)**2)
    d_n2=np.min((nseries2-x)**2)
    d_m=np.min((mainseries-x)**2)
    #print(d_n1, d_n2, d_m)
    if d_n1>100 and d_n2>100 and d_m>100:
        return 3
    if d_n1<d_n2 and d_n1<d_m:
        return 0
    elif d_n2<d_n1 and d_n2<d_m:
        return 1
    elif d_m<d_n1 and d_m<d_n2:
        return 2
```

```
            else:
                raise ValueError
```

```
-3.0259262086513994
Erste Nebenserie
1: 818.75
2: 569.86
3: 499.57
4: 468.20
5: 451.12
6: 440.68
7: 433.81
8: 429.01
9: 425.54
10: 422.93
11: 420.92
12: 419.35
13: 418.08
14: 417.05
15: 416.20
16: 415.49
17: 414.89
Zweite Nebenserie
1: 1173.42
2: 622.23
3: 518.57
4: 477.50
5: 456.44
6: 444.03
7: 436.06
8: 430.61
9: 426.71
10: 423.82
11: 421.61
12: 419.89
13: 418.52
14: 417.41
15: 416.50
16: 415.74
hauptserie
1: 968.70
2: 355.23
3: 293.31
4: 271.92
5: 261.71
6: 255.98
7: 252.42
8: 250.05
```

```
9: 248.38
10: 247.17
11: 246.26
12: 245.56
13: 245.01
```

```
In [2]: def comma_to_float(valstr):
            return float(valstr.replace(',','.'))

        lamb_n_left, inten_n_left= np.loadtxt('natrium links.txt', skiprows=17, encoding="latin
                                              converters={0:comma_to_float, 1:comma_to_float},
                                             comments='>',unpack=True)

        plt.plot(lamb_n_left,inten_n_left)
        plt.title('Natriumspektrum geringer Intensität')
        plt.xlabel('Wellenlaenge / nm ')
        plt.ylabel('Intensitaet / b.E')
        plt.yscale('log')
        plt.ylim((100,60000))
        plt.xlim((300,540))

        (maxima,)=signal.argrelmax(inten_n_left, order=40)
        for i in maxima:
            if 150<inten_n_left[i] and lamb_n_left[i]<550:
                if find_series(lamb_n_left[i])==0:
                    color="violet"
                elif find_series(lamb_n_left[i])==1:
                    color="red"
                elif find_series(lamb_n_left[i])==2:
                    color="green"
                elif find_series(lamb_n_left[i])==3:
                    color="yellow"
                plt.annotate(str(lamb_n_left[i]),(lamb_n_left[i],inten_n_left[i]), color=color)
        red_patch = mpatches.Patch(color='red', label='Zweite Nebenserie')
        violet_patch = mpatches.Patch(color='violet', label='Erste Nebenserie')
        green_patch = mpatches.Patch(color='green', label='Hauptserie')
        yellow_patch = mpatches.Patch(color='yellow', label='Nicht zuzuordnen')


        plt.legend(handles=[red_patch, violet_patch, green_patch, yellow_patch])


        plt.show()
```

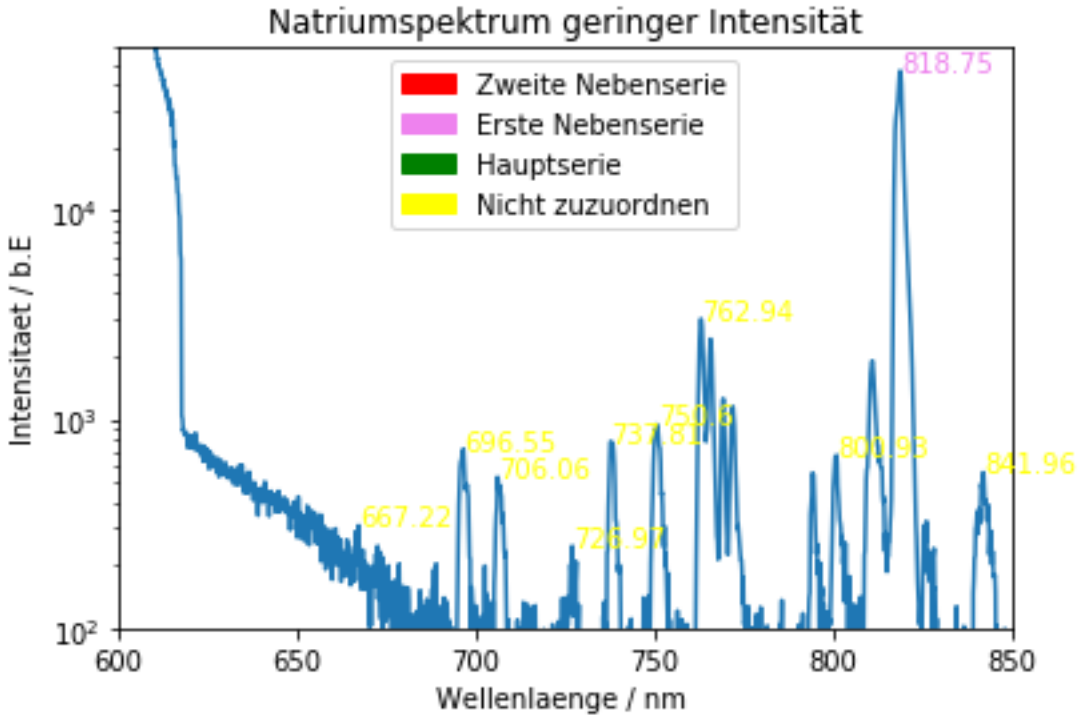



```
In [14]: lamb_n_right, inten_n_right= np.loadtxt("natrium rechts.txt", skiprows=17, encoding="
                                                converters={0:comma_to_float, 1:comma_to_float},
                                               comments='>',unpack=True)
```

```
plt.plot(lamb_n_right,inten_n_right)
plt.title('Natriumspektrum geringer Intensität')
plt.xlabel('Wellenlaenge / nm ')
plt.ylabel('Intensitaet / b.E')
plt.yscale('log')
plt.ylim((100,60000))
plt.xlim((600,850))
```

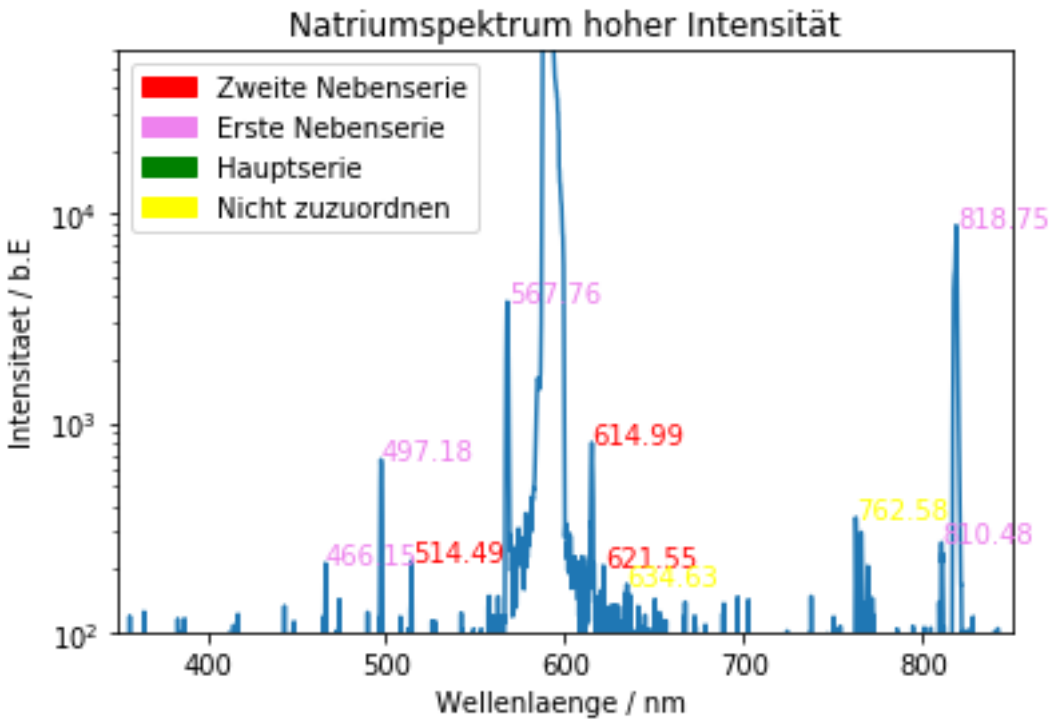
```
(maxima,)=signal.argrelmax(inten_n_right, order=40)
for i in maxima:
    if 150<inten_n_right[i]<60000 and 850>lamb_n_right[i]>650:
        if find_series(lamb_n_left[i])==0:
            color="violet"
        elif find_series(lamb_n_left[i])==1:
            color="red"
        elif find_series(lamb_n_left[i])==2:
            color="green"
        elif find_series(lamb_n_left[i])==3:
            color="yellow"
```

```
        plt.annotate(str(lamb_n_right[i]),(lamb_n_right[i],inten_n_right[i]), color=co
red_patch = mpatches.Patch(color='red', label='Zweite Nebenserie')
yellow_patch = mpatches.Patch(color='violet', label='Erste Nebenserie')
green_patch = mpatches.Patch(color='green', label='Hauptserie')

yellow_patch = mpatches.Patch(color='yellow', label='Nicht zuzuordnen')


plt.legend(handles=[red_patch, violet_patch, green_patch, yellow_patch])



plt.show()
```



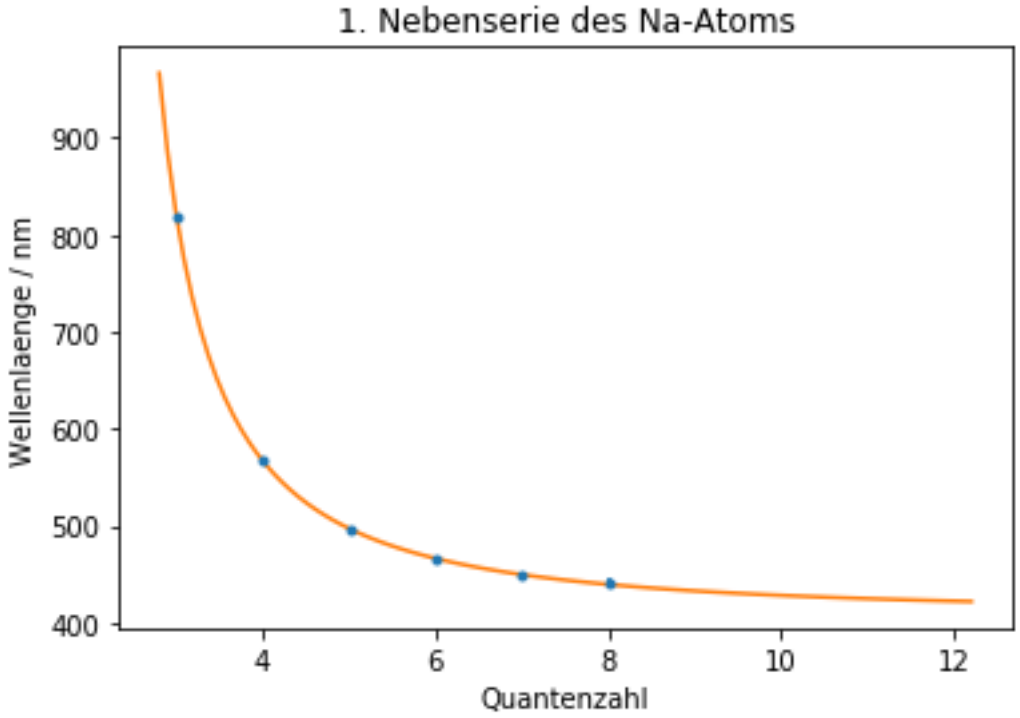
In [4]:
```
%matplotlib inline

lamb_n_high, inten_n_high= np.loadtxt(r'natrium ges.txt', skiprows=17, encoding="latin-
                                      converters={0:comma_to_float, 1:comma_to_float},
                                     comments='>',unpack=True)

plt.plot(lamb_n_high,inten_n_high)
plt.title('Natriumspektrum hoher Intensität')
plt.xlabel('Wellenlaenge / nm ')
```

```
plt.ylabel('Intensitaet / b.E')
plt.yscale('log')
plt.ylim((100,60000))
plt.xlim((350,850))

(maxima,)=signal.argrelmax(inten_n_high, order=20)
for i in maxima:
    if 150<inten_n_high[i]<60000 and 400<lamb_n_high[i]<850:
        if find_series(lamb_n_left[i])==0:
            color="violet"
            label="Erste Nebenserie"
        elif find_series(lamb_n_left[i])==1:
            color="red"
            label="Zweite Nebenserie"
        elif find_series(lamb_n_left[i])==2:
            color="green"
            label="Hauptserie"
        elif find_series(lamb_n_left[i])==3:
            color="yellow"

        plt.annotate(str(lamb_n_high[i]),(lamb_n_high[i],inten_n_high[i]), color=color)
red_patch = mpatches.Patch(color='red', label='Zweite Nebenserie')
yellow_patch = mpatches.Patch(color='violet', label='Erste Nebenserie')
green_patch = mpatches.Patch(color='green', label='Hauptserie')

yellow_patch = mpatches.Patch(color='yellow', label='Nicht zuzuordnen')


plt.legend(handles=[red_patch, violet_patch, green_patch, yellow_patch])

plt.show()
```




```
In [5]: wellenl=np.array([819,567,497,465,449,442])
        fehler=np.array([2,2,2,2,2,3])
        quantenz=np.arange(3,9)

        from scipy.optimize import curve_fit
        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, wellenl, sigma=fehler, p0=para)
        print("E_Ry=",popt[0],", Standardfehler=",np.sqrt(pcov[0][0]))
        print("E_3p=",popt[1],", Standardfehler=",np.sqrt(pcov[1][1]))
        print("D_d=",popt[2],", Standardfehler=",np.sqrt(pcov[2][2]))
        chi2_=np.sum((fit_func(quantenz,*popt)-wellenl)**2/fehler**2)
        dof=len(quantenz)-3 #dof:degrees of freedom
        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('1. Nebenserie des Na-Atoms')
```

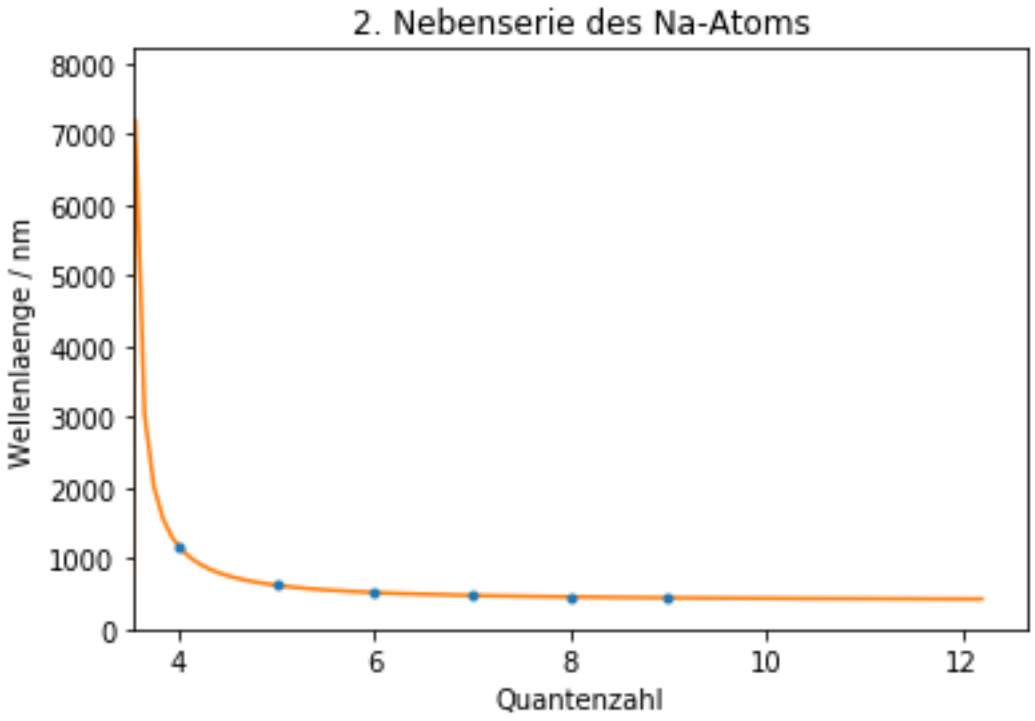
```
        x=np.linspace(2.8,12.2,100)
        plt.plot(x,fit_func(x,*popt))
        plt.show()

E_Ry= -12.864213709698324 , Standardfehler= 0.6599486519701852
E_3p= -3.027350950631146 , Standardfehler= 0.01305900666142453
D_d= 0.08472085327976983 , Standardfehler= 0.06329274611136763
chi2= 1.2168980739112198
chi2_red= 0.40563269130373997
Wahrscheinlichkeit= 75.0 %
```




```
In [6]: wellenl2=np.array([1173.8,621,514,473,458,443])
        fehler2=np.array([1,2,2,2,3,2])
        quantenz2=np.arange(4,10)

        from scipy.optimize import curve_fit
        def fit_func2(m,E_Ry,E_3p,D_s):
            return 1.2398E3/(E_Ry/(m-D_s)**2-E_3p)
        para2=[-13.6,-3,0.88]
        popt2,pcov2=curve_fit(fit_func2, quantenz2, wellenl2, sigma=fehler2, p0=para2)
        print("E_Ry=",popt2[0],", Standardfehler=",np.sqrt(pcov2[0][0]))
        print("E_3p=",popt2[1],", Standardfehler=",np.sqrt(pcov2[1][1]))
        print("D_s=",popt2[2],", Standardfehler=",np.sqrt(pcov2[2][2]))
```

```
chi2_2=np.sum((fit_func(quantenz2,*popt2)-wellenl2)**2/fehler2**2)
dof2=len(quantenz2)-3 #dof:degrees of freedom
chi2_red2=chi2_/dof2
print("chi2=", chi2_2)
print("chi2_red=",chi2_red2)
from scipy.stats import chi2
prob=round(1-chi2.cdf(chi2_,dof),2)*100
print("Wahrscheinlichkeit=",prob,"%")
plt.errorbar(quantenz2,wellenl2,fehler2, 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_func2(x,*popt2))
plt.xlim(3.55)
plt.ylim(0)
plt.show()
```

```
E_Ry= -13.586733939148928 , Standardfehler= 0.850601865639722
E_3p= -3.0397460808867485 , Standardfehler= 0.020807672295928583
D_s= 1.3827957405771867 , Standardfehler= 0.06918262703728632
chi2= 4.697323168860562
chi2_red= 0.40563269130373997
Wahrscheinlichkeit= 75.0 %
```



In [ ]: