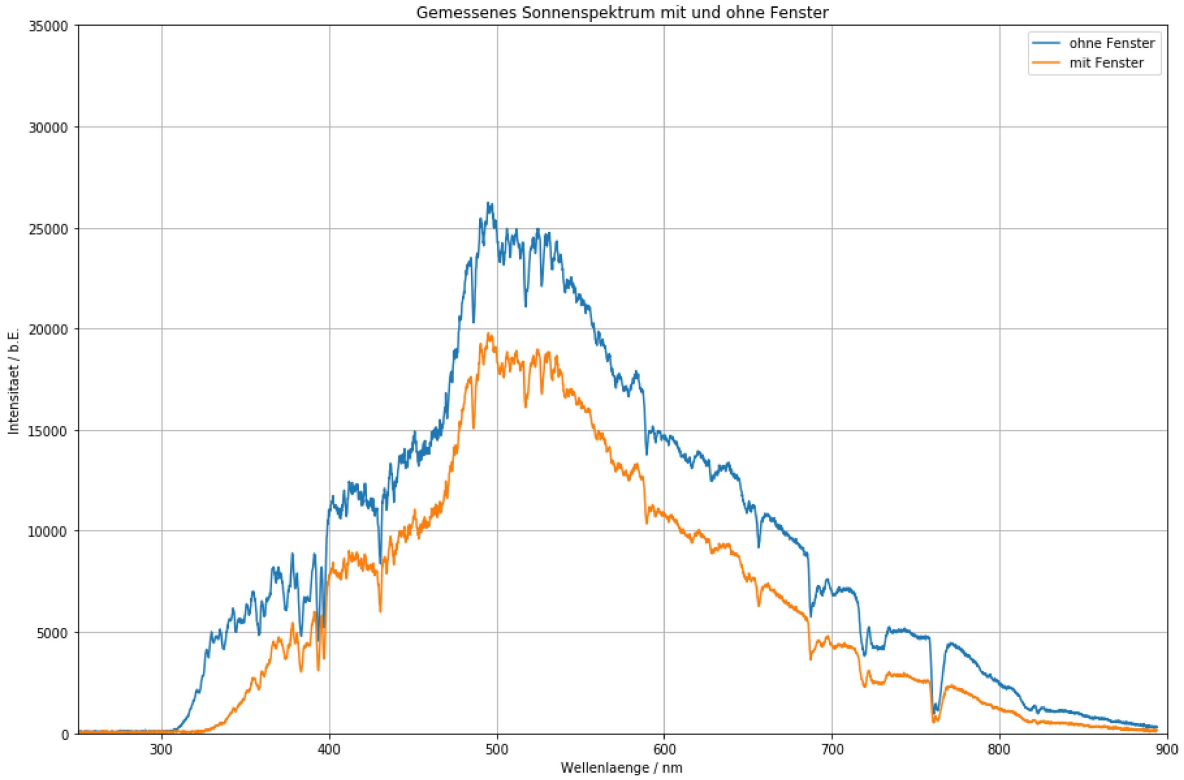
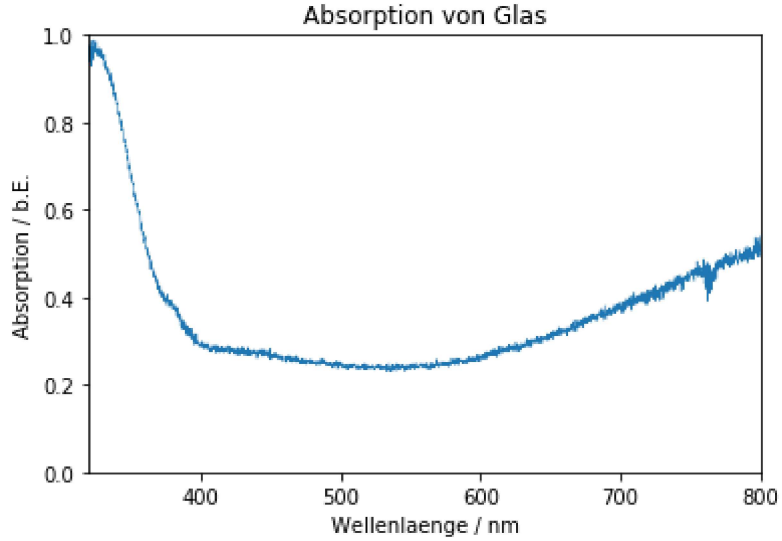
In [15]:

```
%matplotlib inline
# importiere Bibliotheken
import matplotlib.pyplot as plt
import numpy as np
# Kommazahlen mit , ausgeben (statt .)
def comma_to_float (valstr):
    return float (valstr.decode("utf-8").replace(',','.'))

# Importiere Daten
# Ohne Fenster
lamb_og, inten_og = np.loadtxt(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234
                               skiprows =17 , converters = { 0 :comma_to_float, 1 :comma_to
                               comments = '>',unpack=True)
# Mit Fenster
lamb_mg, inten_mg = np.loadtxt(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234
                               skiprows =17 , converters = { 0 :comma_to_float, 1 :comma_to
                               comments = '>',unpack=True)


# Plot für ohne Fenster
plt.figure(figsize=(15,10))
plt.plot(lamb_og, inten_og, label = 'ohne Fenster' )
# Plot für mit 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()
# Werte- / Definitionsbereich
plt.ylim(( 0 , 35000 ))
plt.xlim(( 250 , 900))
# Speichern
plt.savefig(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234 - Lichtquellen\Son
```



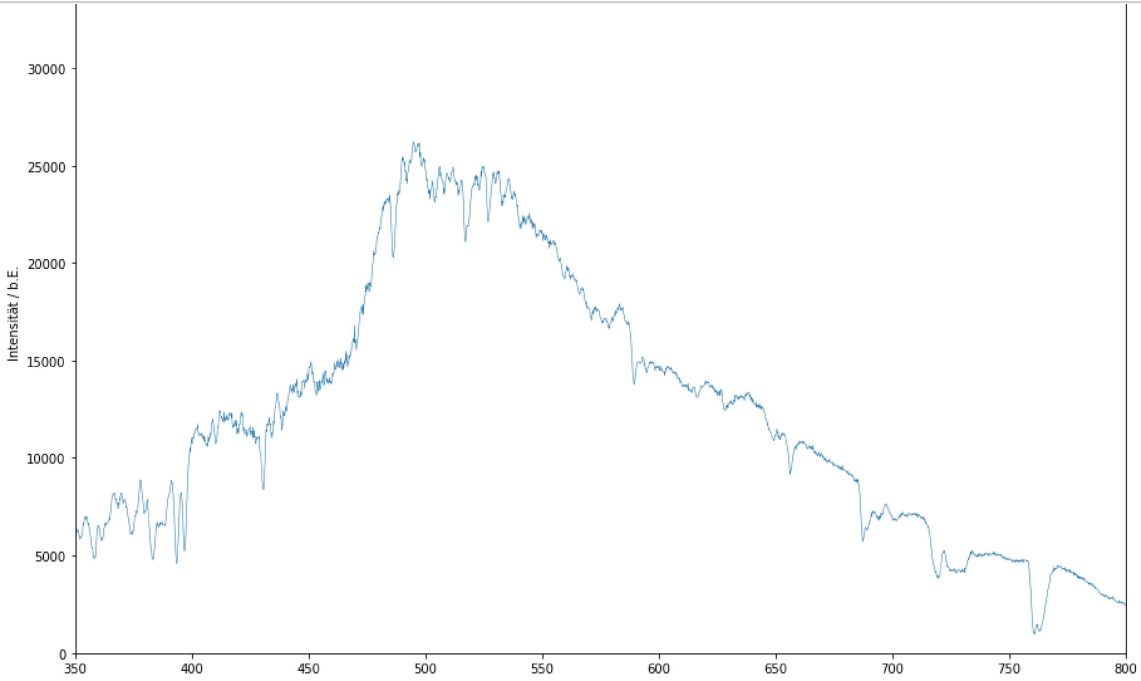
In [16]:

```
# Berechnung der Absorption von Glas
# Werte die gleich Null sind werden übersprungen
A =1- inten_mg[inten_og>0] / inten_og[inten_og>0]
#Hier wird der zweite Plot initialisiert
plt.plot(lamb_mg[inten_og>0], A, linewidth = 0.5)
plt.title( 'Absorption von Glas' )
plt.xlabel( 'Wellenlaenge / nm' )
plt.ylabel( 'Absorption / b.E.' )
plt.ylim(( 0 , 1))
plt.xlim(( 320 , 800 ))
#datei speichern
plt.savefig(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234 - Lichtquellen\Abs
```



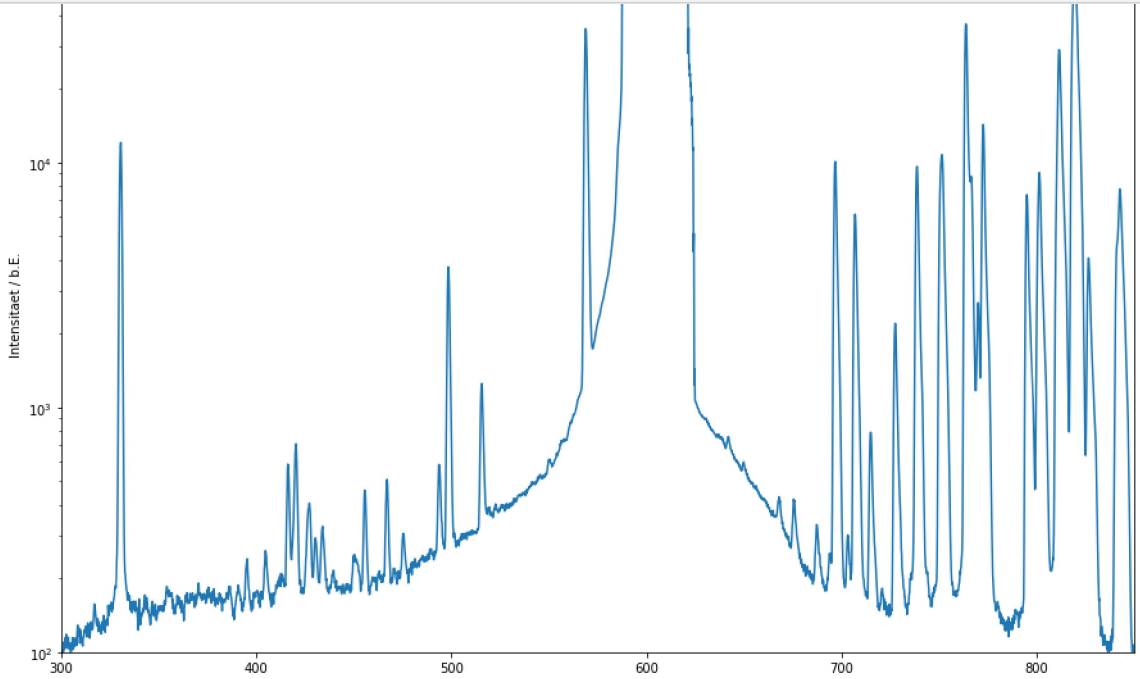
In [17]:

```
#plot zur Bestimmung der Frauenhoferlinien
plt.figure(figsize=(15,10))
plt.plot(lamb_og, inten_og, linewidth = 0.5)
plt.title('Sonnenspektrum')
plt.xlabel('Wellenlänge /nm')
plt.ylabel('Intensität / b.E.')
plt.ylim((0, 35000))
plt.xlim((350, 800))
plt.savefig(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234 - Lichtquellen\Fra
```



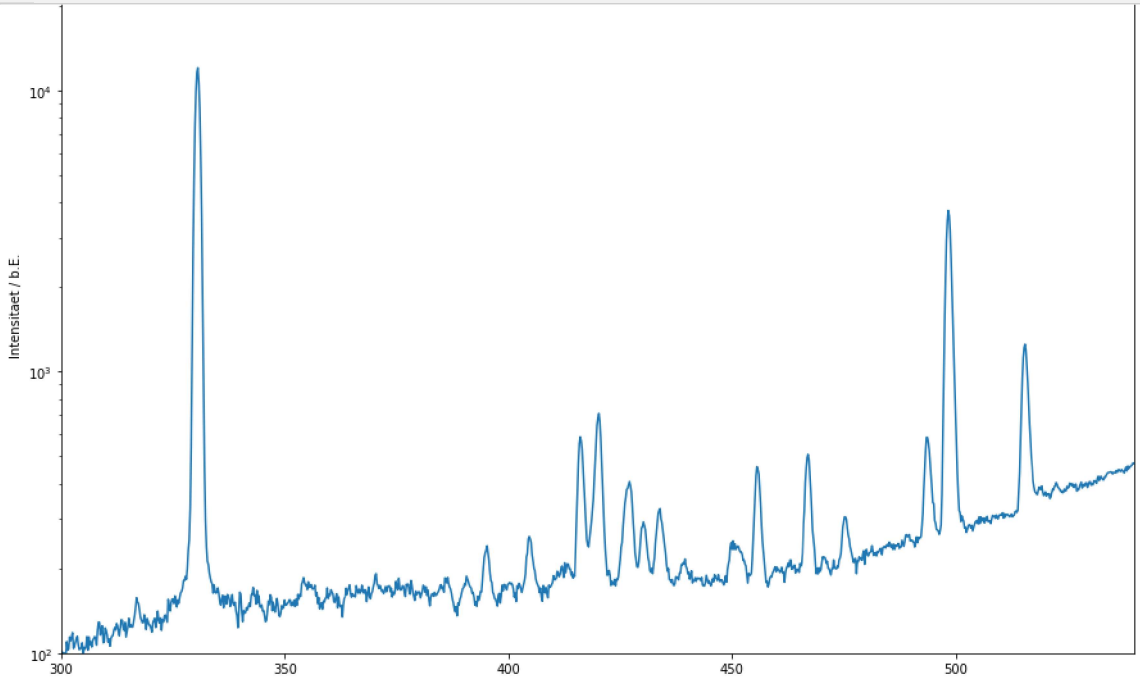
In [18]:

```
#Natrimspektum bei hoher Intensität (gesamt)
lamb_og, inten_og = np.loadtxt(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234
                               skiprows =17 , converters = { 0 :comma_to_float, 1 :comma_to
                               unpack = True)
plt.figure(figsize=(15,10))
plt.plot(lamb_og, inten_og)
plt.title( 'Spektrallinien hoher Intensität bei der Natriummessung' )
plt.xlabel( 'Wellenlaenge / nm' )
plt.ylabel( 'Intensitaet / b.E.' )
plt.yscale( 'log' )
plt.ylim(( 100 , 70000 ))
plt.xlim(( 300 , 850 ))
plt.savefig(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234 - Lichtquellen\Nat
            format = "pdf")
```



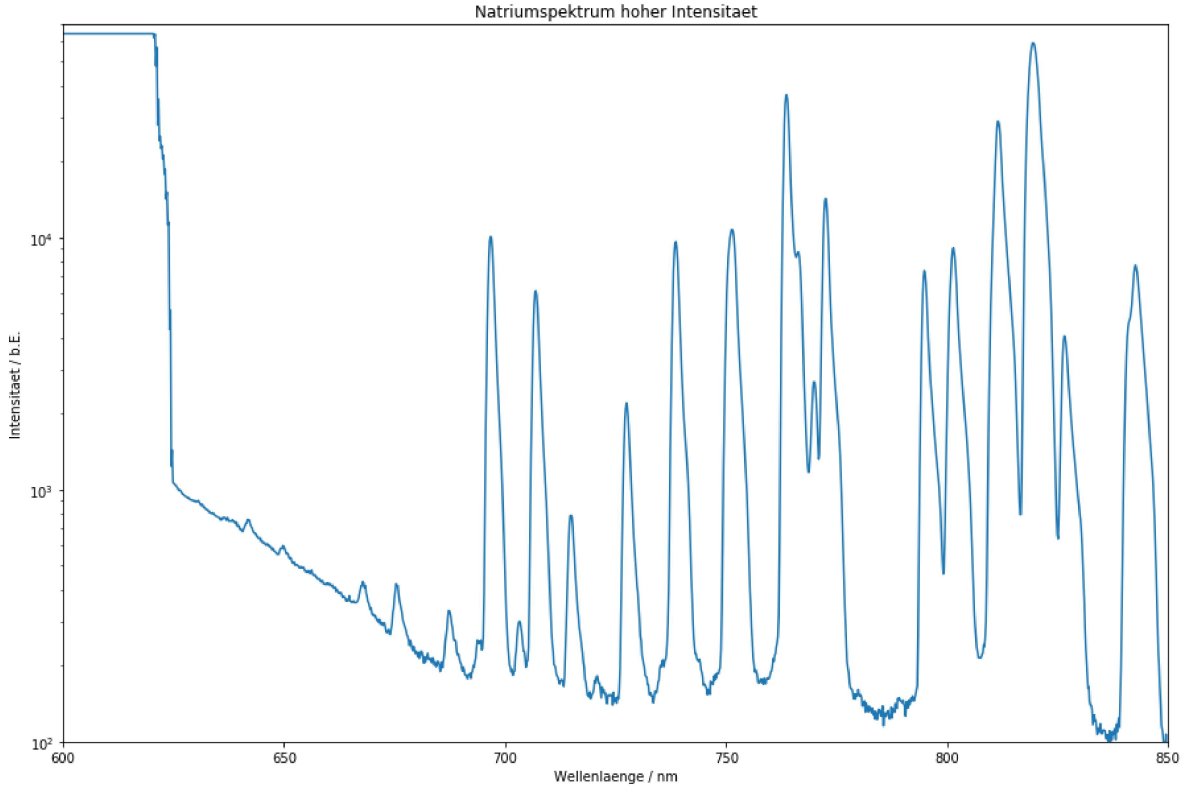
In [20]:

```
#Natriumspektrum, hoher Intensität (300nm - 540nm)

lamb_og, inten_og = np.loadtxt(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234
                               skiprows =17 , converters = { 0 :comma_to_float, 1 :comma_to
                               comments = '>' , unpack = True)
plt.figure(figsize=(15,10))
plt.plot(lamb_og, inten_og)
plt.title( 'Natriumspektrum hoher Intensitaet' )
plt.xlabel( 'Wellenlaenge / nm' )
plt.ylabel( 'Intensitaet / b.E.' )
plt.yscale( 'log' )
plt.ylim(( 100 , 30000 ))
plt.xlim(( 300 , 540 ))
plt.savefig(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234 - Lichtquellen\Nat
            format = "pdf" )
```



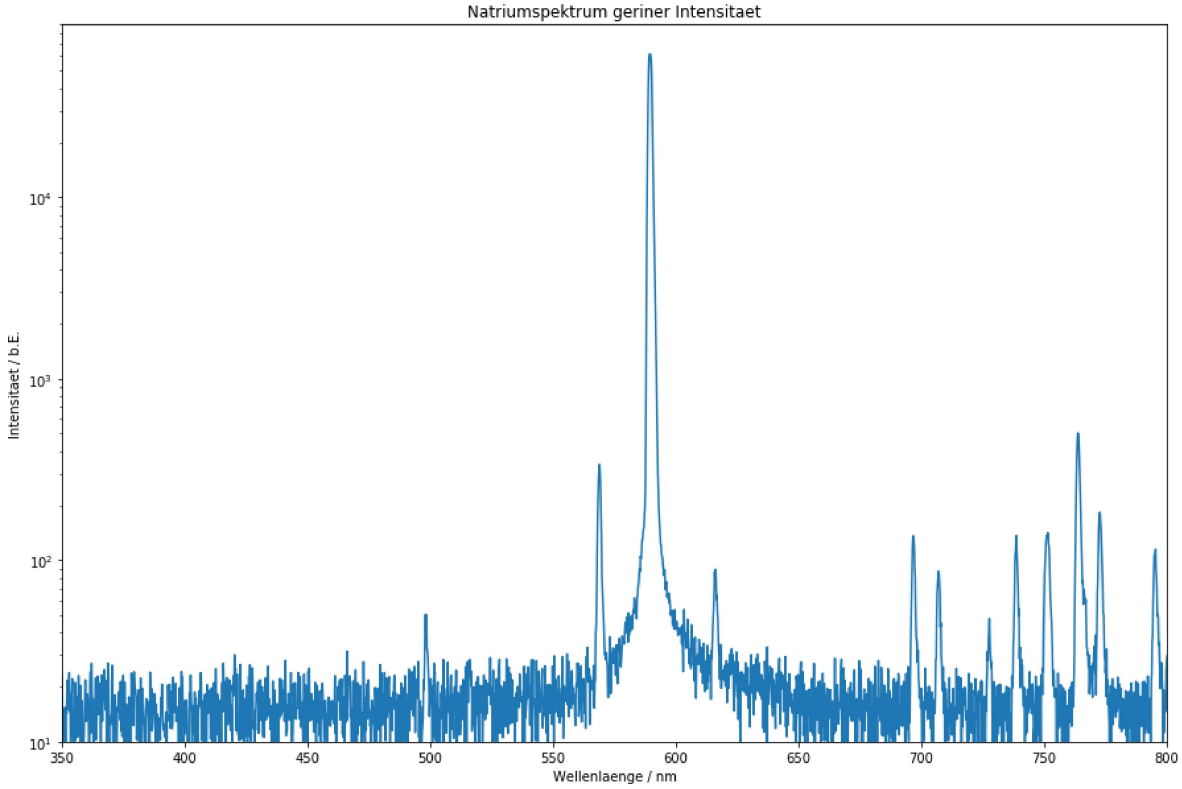
In [21]:

```
#Natriumspektrum, hoher Intensität (600nm - 850nm)


lamb_og, inten_og = np.loadtxt(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234
                               skiprows =17 , converters = { 0 :comma_to_float, 1 :comma_to
                               comments = '>' , unpack = True)
plt.figure(figsize=(15,10))
plt.plot(lamb_og, inten_og)
plt.title( 'Natriumspektrum hoher Intensitaet' )
plt.xlabel( 'Wellenlaenge / nm' )
plt.ylabel( 'Intensitaet / b.E.' )
plt.yscale( 'log' )
plt.ylim(( 100 , 70000 ))
plt.xlim(( 600 , 850 ))
plt.savefig(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234 - Lichtquellen\Nat
            format = "pdf" )
```



In [22]:

```
#Natriumspektrum, hohe Intensität

lamb_og, inten_og = np.loadtxt(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234
                               skiprows =17 , converters = { 0 :comma_to_float, 1 :comma_to
                               comments = '>' , unpack = True)
plt.figure(figsize=(15,10))
plt.plot(lamb_og, inten_og)
plt.title( 'Natriumspektrum geriner Intensitaet' )
plt.xlabel( 'Wellenlaenge / nm' )
plt.ylabel( 'Intensitaet / b.E.' )
plt.yscale( 'log' )
plt.ylim(( 10 , 90000 ))
plt.xlim(( 350 , 800 ))
plt.savefig(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234 - Lichtquellen\Nat
            format = "pdf" )
```



In [23]:

```
#Erwartete Linien für die 1. Nebenserie
for m in range (3,13):
    l=1.2398E3/(-13.605/(m)**2+3.025)
    d=(1.2398E3/((-13.605/(m)**2+3.025))**2*0.0014)
    print('m={m:2d}, lambda={l:6.1f}, deltaLambda={d:1.1f}'.format(m=m,l=l,d=d))
```

```
m= 3, lambda= 819.3, deltaLambda=0.8
m= 4, lambda= 570.1, deltaLambda=0.4
m= 5, lambda= 499.8, deltaLambda=0.3
m= 6, lambda= 468.4, deltaLambda=0.2
m= 7, lambda= 451.3, deltaLambda=0.2
m= 8, lambda= 440.8, deltaLambda=0.2
m= 9, lambda= 433.9, deltaLambda=0.2
m=10, lambda= 429.2, deltaLambda=0.2
m=11, lambda= 425.7, deltaLambda=0.2
m=12, lambda= 423.1, deltaLambda=0.2
```

In [24]:

```
#Erwartete Linien für die 2. Nebenserie
for m in range (4,10):
    l=1.2398E3/(-13.605/(m-1.37)**2+3.025)
    d=((33735/((m-1.37)**3*(3.025-(13.605/(1.37-m)**2))**2)*0.0006)**2+
       (1.2398E3/((-13.605/(m-1.37)**2+3.025))**2*0.0014)**2)**(1/2)
    print('m={m:2d}, lambda={l:6.1f}, deltaLambda={d:1.1f}'.format(m=m,l=l,d=d))
```

```
m= 4, lambda=1171.7, deltaLambda=1.8
m= 5, lambda= 622.2, deltaLambda=0.5
m= 6, lambda= 518.7, deltaLambda=0.3
m= 7, lambda= 477.6, deltaLambda=0.3
m= 8, lambda= 456.6, deltaLambda=0.2
m= 9, lambda= 444.2, deltaLambda=0.2
```

In [25]:

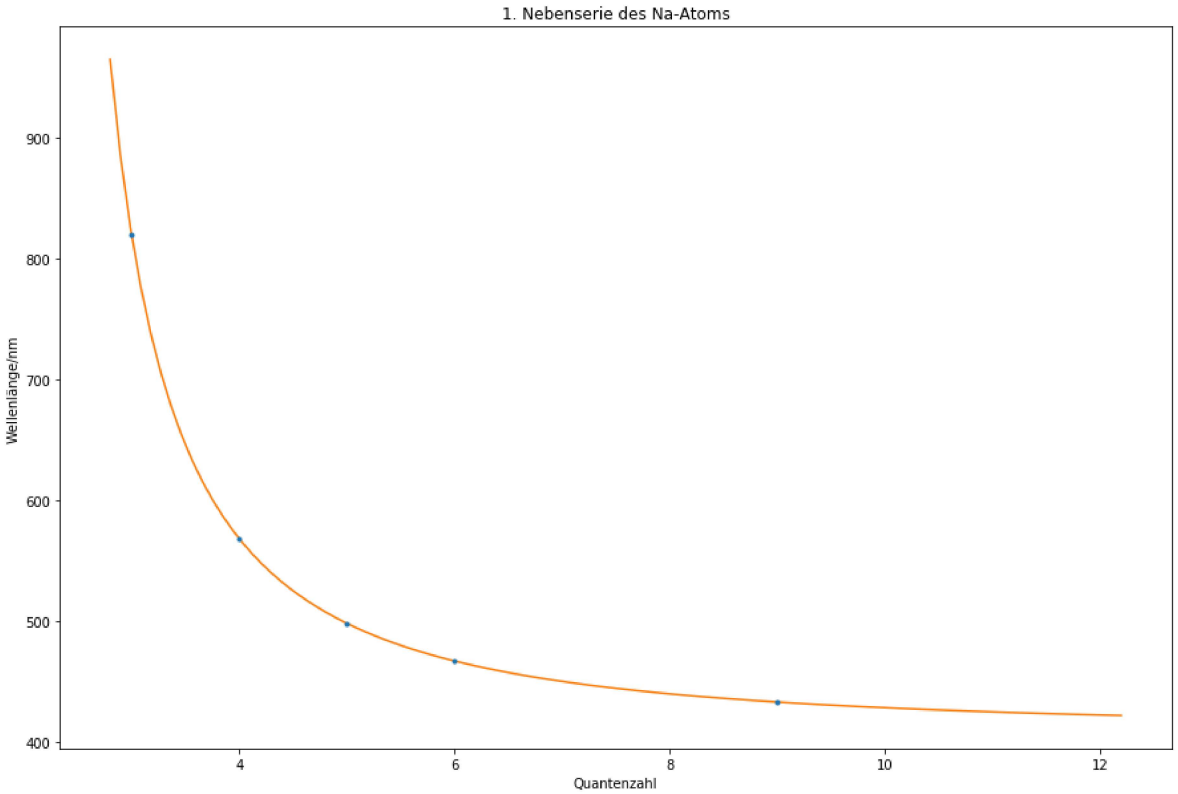
```
#Erwartete Linien für die Hauptserie
for m in range (4,6):
    l=1.2398E3/(-13.605/(m-0.8792)**2+5.13)
    d=((33735/((m-1.37)**3*(3.025-(13.605/(0.8792-m)**2))**2)*0.0009)**2+
       (1.2398E3/((-13.605/(m-0.8792)**2+5.13))**2*0.0017)**2)**(1/2)
    print('m={m:2d}, lambda={l:6.1f}, deltaLambda={d:1.1f}'.format(m=m,l=l,d=d))
```

```
m= 4, lambda= 332.1, deltaLambda=0.6
m= 5, lambda= 286.4, deltaLambda=0.2
```

In [26]:

```
#Bestimmung der Serienenergien und der l-Abhängigen Korrekturfaktoren
wellenl=np.array([819.4, 568.9, 498.4, 467.1, 433.9])
fehler=np.array([1.1, 0.4, 0.4, 0.3, 0.4])
quantenz=np.array([3, 4, 5, 6, 9])

plt.figure(figsize=(15,10))
plt.errorbar(quantenz,wellenl,fehler, fmt=".")
plt.xlabel('Quantenzahl')
plt.ylabel('Wellenlänge/nm')
plt.title('1. Nebenserie des Na-Atoms')

from scipy.optimize import curve_fit
def fit_func1(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_func1, quantenz, wellenl, sigma=fehler ,p0=para)

x = np.linspace(2.8, 12.2, 100)
plt.plot(x, fit_func1(x, *popt))
plt.savefig(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234 - Lichtquellen\Kor

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]))
```

```
E_Ry= -13.2125823756 ,Standardfehler= 0.255571640643
E_3p= -3.02487449916 ,Standardfehler= 0.00469692388246
D_d= 0.0442102837285 ,Standardfehler= 0.0253541724774
```



In [27]:

```
chi2_=np.sum((fit_func1(quantenz,*popt)-wellenl)**2/fehler**2)
dof=len(quantenz)-3 #dof:degreesof freedom, Freheitsgrad
chi2_red=chi2_/dof
print("chi2=", chi2_)
print("chi2_red=",chi2_red)
```

```
chi2= 3.13711095333
chi2_red= 1.56855547666
```

In [28]:

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

```
Wahrscheinlichkeit: 21.0 %
```

In [29]:

```
#Bestimmung der Serienenergien und der l-Abhängigen Korrekturfaktoren
wellen2=np.array([515.5, 475.3, 455.6])
fehler=np.array([0.4, 0.3, 0.3])
quantenz=np.array([6, 7, 8])

plt.figure(figsize=(15,10))
plt.errorbar(quantenz,wellen2,fehler, fmt=".")
plt.xlabel('Quantenzahl')
plt.ylabel('Wellenlänge/nm')
plt.title('2. Nebenserie des Na-Atoms')

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)

para = [-13.6,-3,1.4]
popt, pcov = curve_fit(fit_func2, quantenz, wellen2, sigma=fehler ,p0=para)

x = np.linspace(4.2, 9, 100)
plt.plot(x, fit_func2(x, *popt))
plt.savefig(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\234 - Lichtquellen\Kor

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]))
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-
packages\scipy\optimize\minpack.py:779: OptimizeWarning: Covariance of the p
arameters could not be estimated
  category=OptimizeWarning)
```

```
E_Ry= -10.1118008691 ,Standardfehler= inf
E_3p= -2.98700465279 ,Standardfehler= inf
D_d= 1.83161992457 ,Standardfehler= inf
```