234auswertung

April 1, 2025

[25]:
```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
from scipy.signal import argrelextrema
from scipy.optimize import curve_fit
from scipy.stats import chi2

plt.rcParams.update({'font.size': 14})

%run ../lib.ipynb

class Consts:
    hc = 1.2398 * 10**(3) #nm eV
    E_Ry = -13.605 # eV
```

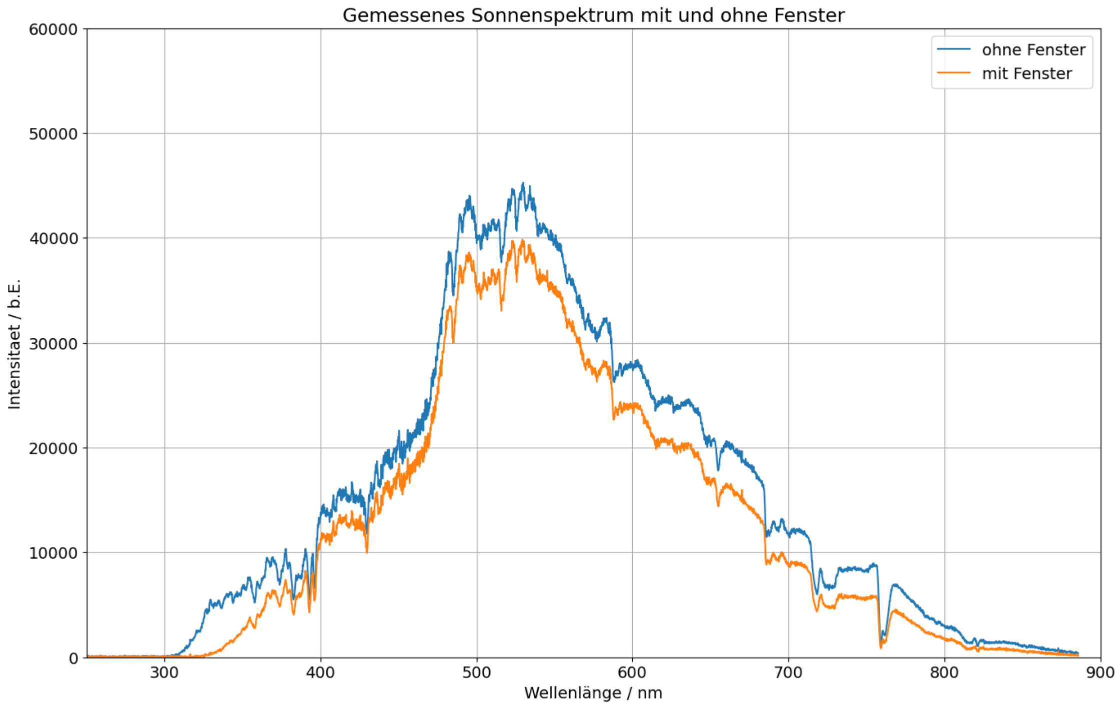
[26]:
```
def comma_to_float(valstr):
    return float(valstr.replace(',','.'))
```

0.0.1 Sonnenspektrum und Fraunhoferlinien

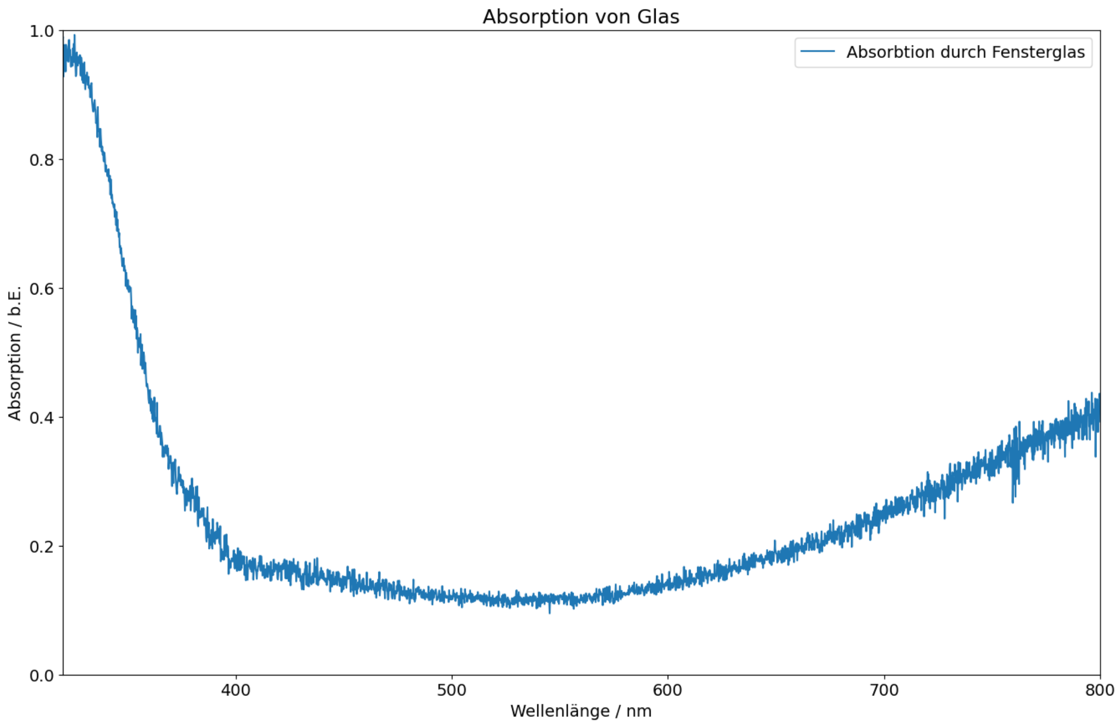
[27]:
```
lamb_og, inten_og = np.loadtxt('sonnenlichtspektrum_ohne_glas.txt', skiprows=17,
                        converters= {0:comma_to_float, 1:comma_to_float},
                        comments='>', unpack=True)

lamb_mg, inten_mg=np.loadtxt('sonnenlichtspektrum_durch_glas.txt', skiprows=17,
                        converters= {0:comma_to_float, 1:comma_to_float},
                        comments='>', unpack=True)
```

[28]:
```
plt.figure(figsize=(16,10))
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('Wellenlänge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.legend()
plt.grid()
plt.ylim((0,60000))
plt.xlim((250,900))
```

```
plt.savefig('out/himmel_m_o_g.png', format='png', bbox_inches='tight')
```



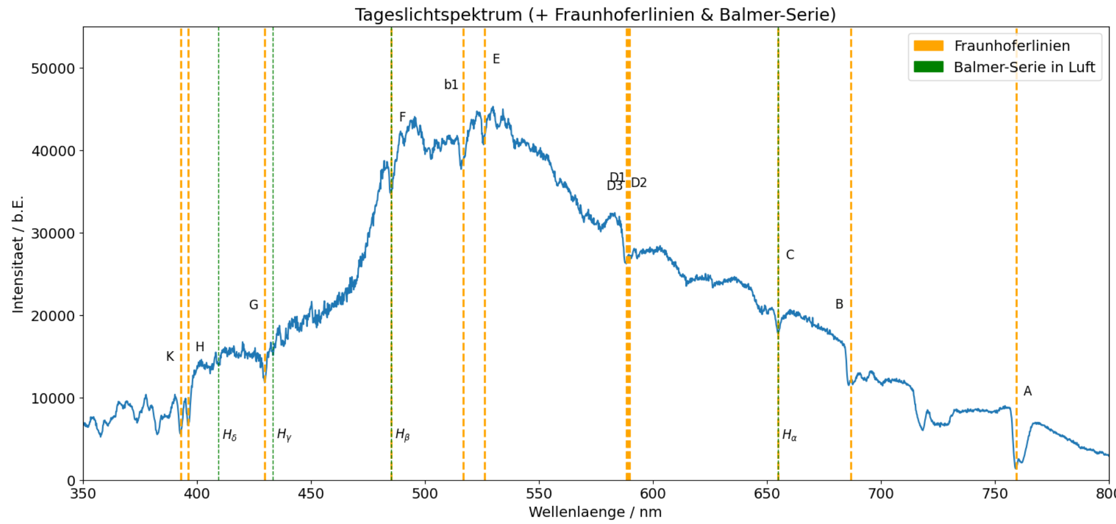
[29]:
```
A = 1 - inten_mg[600:] / inten_og[600:]

plt.figure(figsize=(16,10))
plt.plot(lamb_mg[600:], A, label='Absorbtion durch Fensterglas')
plt.title('Absorption von Glas')
plt.xlabel('Wellenlänge / nm')
plt.ylabel('Absorption / b.E.')
plt.ylim((0,1))
plt.xlim((320,800))
plt.legend()
plt.savefig("out/absorption_glas.png", format='png', bbox_inches='tight')
```



[30]:
```python
# lokale minima, abgelesen aus textdatei:
mins = [
    ('K', 393.0, 1, 5533.61, 393.4),
    ('H', 396.1, 1, 6619.03, 396.8),
    ('G', 429.8, 1, 11786.06, 430.8),
    ('F', 485.2, 1, 34493.94, 486.1),
    ('b1', 516.7, 1, 38450.61, 518.4),
    ('E', 526.2, 1, 41561.48, 527.0),
    ('D3', 588.4, 1, 26230.38, 587.6),
    ('D2', 589.0, 1, 26549.87, 589.0),
    ('D1', 589.7, 1, 27166.31, 589.6),
    ('C', 655.0, 1, 17813.23, 656.3),
    ('B', 686.7, 1, 11865.93, 686.7),
    ('A', 759.4, 1, 1339.37, 759.4)
]

balmer_air_nist = [
    (r'$H_\alpha$', 655.0, 1, 656.3),
    (r'$H_\beta$', 485.2, 1, 486.1),
    (r'$H_\gamma$', 433.4, 1, 434.0),
    (r'$H_\delta$', 409.5, 1, 410.1),
]
```

```python
plt.figure(figsize=(18,8))
plt.plot(lamb_og, inten_og)
plt.title('Tageslichtspektrum (+ Fraunhoferlinien & Balmer-Serie)')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.ylim((0,55000))
plt.xlim((350,800))
i = 0
for m in mins:
    print(f'{m[0]}\t :: expected = {m[4]}nm, measured = {m[1]}nm, diff =␣
 ↪{ValErr(m[1], m[2]).sigmadiff_fmt(ValErr(m[4], 0))}')
    plt.axvline(x=m[1], linewidth=2, linestyle='--', color='orange')
    plt.text(x=m[1] + (5 * (-1 if i % 2 == 0 else 1)), y=(m[3]+9000), s=m[0],␣
 ↪ha='center', fontsize=12)
    i = i + 1
#plt.xticks(np.arange(350, 801, 5))

for b in balmer_air_nist:
    print(f'{b[0]}\t :: expected = {b[3]}nm, measured = {b[1]}nm, diff =␣
 ↪{ValErr(b[1], b[2]).sigmadiff_fmt(ValErr(b[3], 0))}')
    plt.axvline(x=b[1], linewidth=1, linestyle='--', color='green')
    plt.text(x=b[1] + 5, y=5000, s=b[0], ha='center', fontsize=12)

fraunhofer_patch = mpatches.Patch(color='orange', label='Fraunhoferlinien')
balmer_air_patch = mpatches.Patch(color='green', label='Balmer-Serie in Luft')

plt.legend(handles=[fraunhofer_patch, balmer_air_patch])
plt.savefig("out/spektrum_fraunhofer_balmer.png", format='png',␣
 ↪bbox_inches='tight')
```

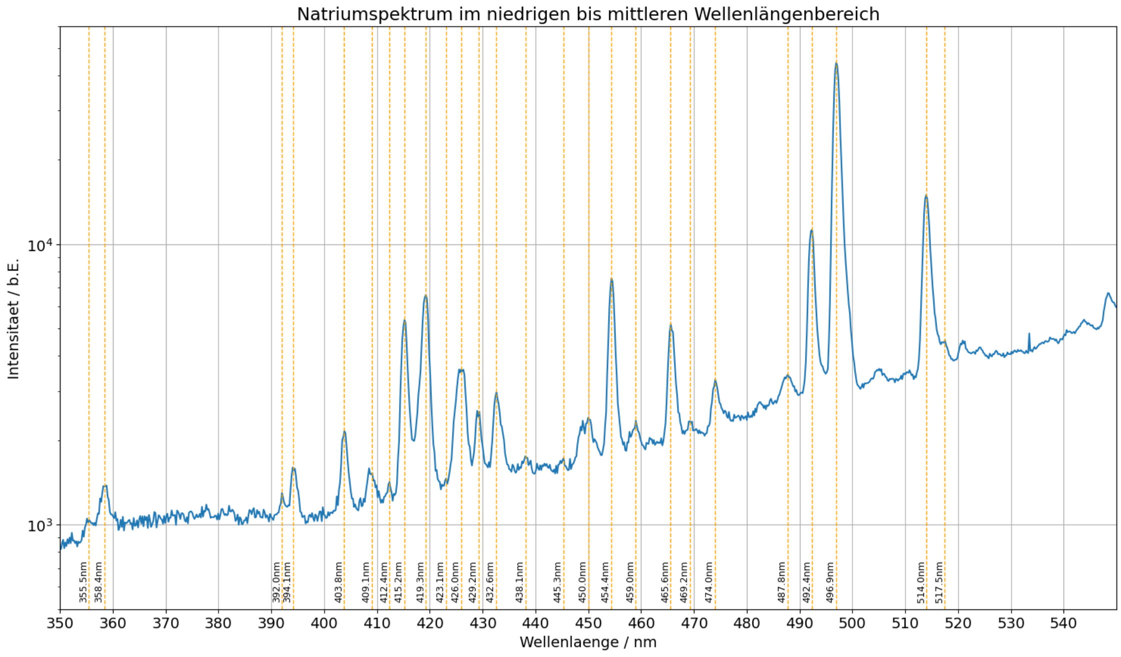
```
K        :: expected = 393.4nm, measured = 393.0nm, diff = 0.4
H        :: expected = 396.8nm, measured = 396.1nm, diff = 0.7
G        :: expected = 430.8nm, measured = 429.8nm, diff = 1.0
F        :: expected = 486.1nm, measured = 485.2nm, diff = 0.91
b1       :: expected = 518.4nm, measured = 516.7nm, diff = 1.7
E        :: expected = 527.0nm, measured = 526.2nm, diff = 0.8
D3       :: expected = 587.6nm, measured = 588.4nm, diff = 0.8
D2       :: expected = 589.0nm, measured = 589.0nm, diff = 0.0
D1       :: expected = 589.6nm, measured = 589.7nm, diff = 0.11
C        :: expected = 656.3nm, measured = 655.0nm, diff = 1.3
B        :: expected = 686.7nm, measured = 686.7nm, diff = 0.0
A        :: expected = 759.4nm, measured = 759.4nm, diff = 0.0
$H_\alpha$       :: expected = 656.3nm, measured = 655.0nm, diff = 1.3
$H_\beta$        :: expected = 486.1nm, measured = 485.2nm, diff = 0.91
$H_\gamma$       :: expected = 434.0nm, measured = 433.4nm, diff = 0.61
$H_\delta$       :: expected = 410.1nm, measured = 409.5nm, diff = 0.61
```



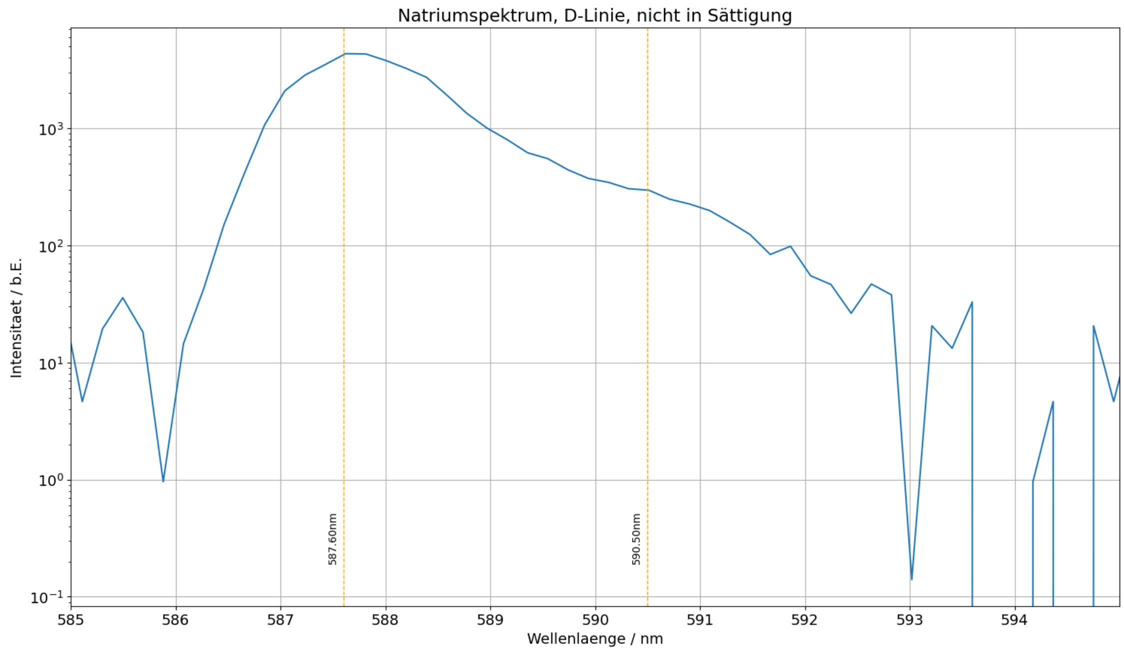
0.0.2 Auswertung des Natriumspektrums

[31]:
```
# Aufgabe 3

lamb_na_a3_500, inten_na_a3_500=np.loadtxt('na_a3_500nm_saett.txt', skiprows=17,
        converters= {0:comma_to_float, 1:comma_to_float},
        comments='>', unpack=True)

plt.figure(figsize=(18,10))
plt.plot(lamb_na_a3_500[586:], inten_na_a3_500[586:])
plt.title('Natriumspektrum im niedrigen bis mittleren Wellenlängenbereich')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.yscale('log')
plt.ylim((500,60000))
plt.xlim((350,550))
plt.xticks(np.arange(350, 550, 10))
plt.grid()

l3 = [(355.5, 1, 1039.16),
(358.4, 1, 1367.92),
(392.0, 1, 1295.52),
(394.1, 1, 1598.57),
(403.8, 1, 2159.1),
(409.1, 1, 1526.74),
(412.4, 1, 1418.91),
(415.2, 1, 5372.36),
(419.3, 1, 6594.08),
(423.1, 1, 1462.68),
```

```
(426.0, 1, 3580.95),
(429.2, 1, 2424.05),
(432.6, 1, 2966.71),
(438.1, 1, 1753.96),
(445.3, 1, 1729.02),
(450.0, 1, 2404.7),
(454.4, 1, 7544.74),
(459.0, 1, 2348.89),
(465.6, 1, 5187.63),
(469.2, 1, 2338.7),
(474.0, 1, 3294.8),
(487.8, 1, 3435.13),
(492.4, 1, 11274.4),
(496.9, 2, 44422.62),
(514.0, 1, 14992.8),
(517.5, 1, 4546.21)]



for l in l3:
    plt.axvline(x=l[0], linewidth=1, linestyle='--', color='orange')
    #plt.axvline(x=l[0]+l[1], linewidth=0.5, linestyle=':', color='gray')
    #plt.axvline(x=l[0]-l[1], linewidth=0.5, linestyle=':', color='gray')
    plt.text(x=l[0]-1, y=540, rotation=90, s=f'{l[0]:0.1f}nm', ha='center',␣
 ↪fontsize=9, color='black')

plt.savefig("out/na_spek_350_550.png", format='png', bbox_inches='tight')
```

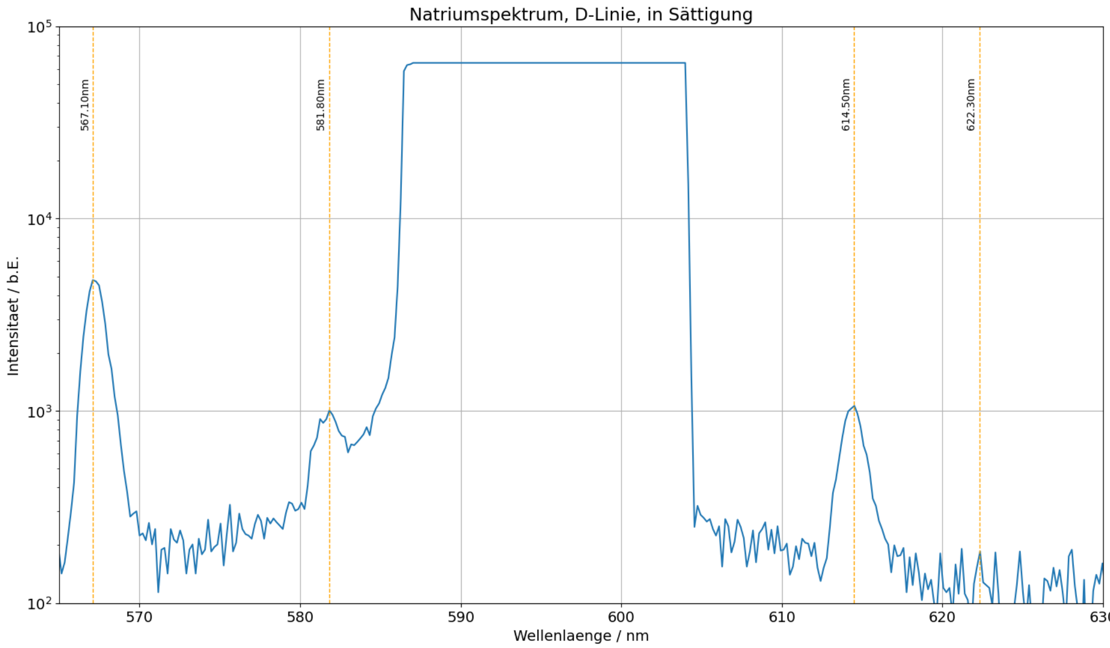


[32]:
```python
# Aufgabe 4, D-Linie nicht in Sättigung

lamb_na_a4_589_ks, inten_na_a4_589_ks=np.loadtxt('na_a4_589nm_keine_saett.txt',␣
↪skiprows=17,
        converters= {0:comma_to_float, 1:comma_to_float},
        comments='>', unpack=True)

plt.figure(figsize=(18,10))
plt.plot(lamb_na_a4_589_ks, inten_na_a4_589_ks)
plt.title('Natriumspektrum, D-Linie, nicht in Sättigung')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.yscale('log')
#plt.ylim((0,60000))
plt.xlim((585,595))
plt.xticks(np.arange(585, 595, 1))
plt.grid()

l4_ks = [
    (587.6, 1, 4340.19),
    (590.5, 1, 295.86)
]

for l in l4_ks:
    plt.axvline(x=l[0], linewidth=1, linestyle='--', color='orange')
    #plt.axvline(x=l[0]+l[1], linewidth=1, linestyle=':', color='gray')
    #plt.axvline(x=l[0]-l[1], linewidth=1, linestyle=':', color='gray')
    plt.text(x=l[0]-0.1, y=0.2, rotation=90, s=f'{l[0]:0.2f}nm', ha='center',␣
↪fontsize=10, color='black')

plt.savefig("out/na_spek_dlinie_nichtsaett.png", format='png',␣
↪bbox_inches='tight')
```



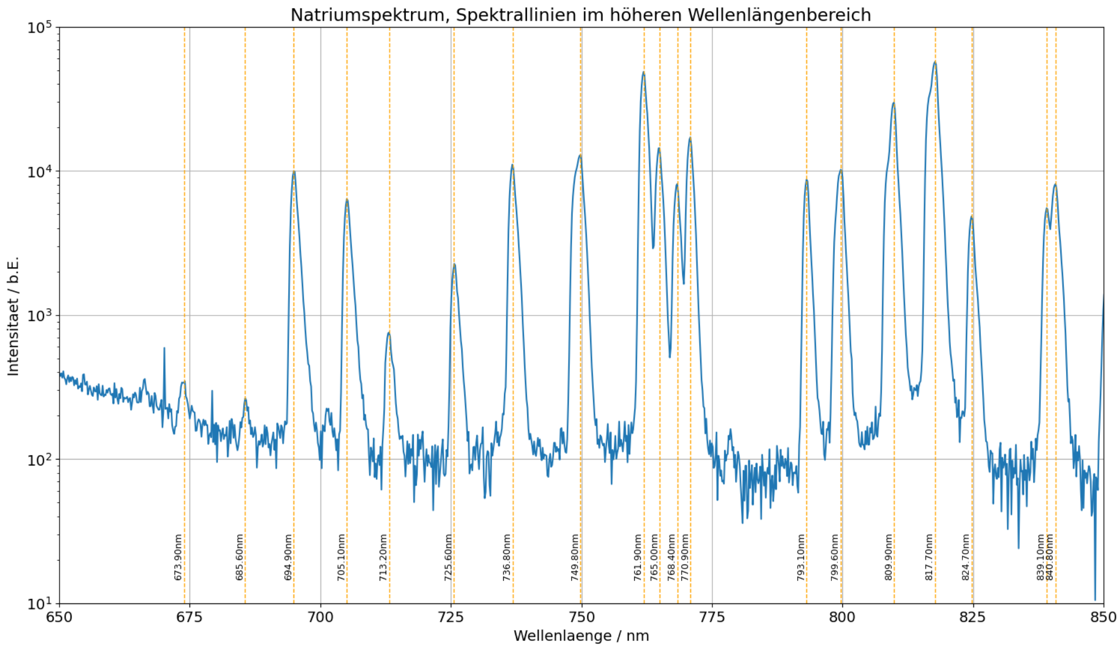
[33]:
```
# Aufgabe 4, D-Linie in Sättigung

lamb_na_a4_589_s, inten_na_a4_589_s=np.loadtxt('na_a4_589nm_saett.txt',␣
↪skiprows=17,
        converters= {0:comma_to_float, 1:comma_to_float},
        comments='>', unpack=True)

plt.figure(figsize=(18,10))
plt.plot(lamb_na_a4_589_s, inten_na_a4_589_s)
plt.title('Natriumspektrum, D-Linie, in Sättigung')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.yscale('log')
plt.ylim((100,100000))
plt.xlim((565,630))
plt.grid()

l4_s = [
    (567.1, 1, 4787.3),
    (581.8, 1, 1004.7),
    (614.5, 1, 1062.04),
    (622.3, 1, 185.51)
]

for l in l4_s:
    plt.axvline(x=l[0], linewidth=1, linestyle='--', color='orange')
```

```
    #plt.axvline(x=l[0]+l[1], linewidth=1, linestyle=':', color='gray')
    #plt.axvline(x=l[0]-l[1], linewidth=1, linestyle=':', color='gray')
    plt.text(x=l[0]-0.5, y=30000, rotation=90, s=f'{l[0]:0.2f}nm', ha='center',␣
 ↪fontsize=10, color='black')

plt.savefig("out/na_spek_dlinie_saett.png", format='png', bbox_inches='tight')
```



[34]:
```
# Aufgabe 5, Linien Hoher Wellenlänge

lamb_na_a5_650, inten_na_a5_650=np.loadtxt('na_a5_650nm_bis_850nm.txt',␣
 ↪skiprows=17,
        converters= {0:comma_to_float, 1:comma_to_float},
        comments='>', unpack=True)

plt.figure(figsize=(18,10))
plt.plot(lamb_na_a5_650, inten_na_a5_650)
plt.title('Natriumspektrum, Spektrallinien im höheren Wellenlängenbereich')
plt.xlabel('Wellenlaenge / nm')
plt.ylabel('Intensitaet / b.E.')
plt.yscale('log')
plt.ylim((10,100000))
plt.xlim((650,850))
plt.grid()

l_a5 = [
```

```
    (673.9, 1, 338.15),
    (685.6, 1, 264.02),
    (694.9, 1, 9924.29),
    (705.1, 1, 6364.51),
    (713.2, 1, 755.53),
    (725.6, 1, 2266.11),
    (736.8, 1, 11056.81),
    (749.8, 1, 12873.36),
    (761.9, 1, 48696.02),
    (765.0, 1, 14367.15),
    (768.4, 1, 8108.97),
    (770.9, 1, 17108.15),
    (793.1, 1, 8694.69),
    (799.6, 1, 10187.65),
    (809.9, 1, 29784.67),
    (817.7, 1, 57133.65),
    (824.7, 1, 4831.81),
    (839.1, 1, 5503.13),
    (840.8, 1, 8105.69)
]

for l in l_a5:
    plt.axvline(x=l[0], linewidth=1, linestyle='--', color='orange')
    #plt.axvline(x=l[0]+l[1], linewidth=0.5, linestyle=':', color='gray')
    #plt.axvline(x=l[0]-l[1], linewidth=0.5, linestyle=':', color='gray')
    plt.text(x=l[0]-1, y=15, rotation=90, s=f'{l[0]:0.2f}nm', ha='center',
 ↪fontsize=9, color='black')

plt.savefig("out/na_spek_650_850.png", format='png', bbox_inches='tight')
```



[35]:

```
# Erwartete Linien für die 1. Nebenserie: md -> 3p

lambda_3 = ValErr(817.7, 1) #nm

E_3p = (Consts.E_Ry / 3**2) - (Consts.hc / lambda_3)

print(E_3p.strfmtf(5,0,'E_3p'))

meas_1ns_vals = [
    817.7,
    567.1,
    496.9,
    469.2,
    450.0,
    438.1,
    432.6,
    429.2,
    426.0,
    423.1
]

meas_1ns_errs = np.array([1,1,2,1,1,1,1,1,1,1,1])

for m in range(3,13):
    l= Consts.hc / ((Consts.E_Ry / m**2) - E_3p)
```

```
    print(f'm = {m:2d}, lambda = {l.strfmtf(3,0)}, lambda_measured =␣
↪{(meas_1ns_vals[m-3]):6.2f}, diff = {ValErr(meas_1ns_vals[m-3],␣
↪meas_1ns_errs[m-3]).sigmadiff_fmt(l)}')
```

```
E_3p = -3.02787 ± 0.00185
m =  3, lambda = 817.700 ± 1.000, lambda_measured = 817.70, diff = 0.01
m =  4, lambda = 569.353 ± 0.485, lambda_measured = 567.10, diff = 2.03
m =  5, lambda = 499.181 ± 0.373, lambda_measured = 496.90, diff = 1.13
m =  6, lambda = 467.857 ± 0.327, lambda_measured = 469.20, diff = 1.28
m =  7, lambda = 450.801 ± 0.304, lambda_measured = 450.00, diff = 0.77
m =  8, lambda = 440.381 ± 0.290, lambda_measured = 438.10, diff = 2.2
m =  9, lambda = 433.510 ± 0.281, lambda_measured = 432.60, diff = 0.88
m = 10, lambda = 428.726 ± 0.275, lambda_measured = 429.20, diff = 0.46
m = 11, lambda = 425.254 ± 0.270, lambda_measured = 426.00, diff = 0.72
m = 12, lambda = 422.651 ± 0.267, lambda_measured = 423.10, diff = 0.44
```

[36]:
```
## Erwartete Linien für die 2. Nebensetie: ms -> 3p

E_3s = E_3p - (Consts.hc / ValErr(590.5, 1))
Delta_s_val = 3 - np.sqrt(Consts.E_Ry / E_3s.val)
Delta_s_err = (np.sqrt(np.abs(Consts.E_Ry)) / (2 * np.abs(E_3s.val) ** (3/2)))␣
↪* E_3s.err
Delta_s = ValErr(Delta_s_val, Delta_s_err)

print_all(E_3s.strfmtf(5,0, 'E_3s'), Delta_s.strfmtf(5,0, 'Delta s'))

meas_2ns_vals = [
    622.34,
    517.5,
    474.0,
    454.4,
    445.3,
]

meas_2ns_errs = np.ones(5)

for m in range(4,10):
    l_val = Consts.hc / ((Consts.E_Ry / (m - Delta_s_val)**2) - E_3p.val)
    l_err = l_val * np.sqrt(E_3p.relerr()**2 + (2*Delta_s.relerr())**2)
    l = ValErr(l_val, l_err)
    if m == 4:
         print(f'm = {m:2d}, lambda = {l.strfmtf(3,0)}')
    else:
        print(f'm = {m:2d}, lambda = {l.strfmtf(3,0)}, lambda_measured =␣
↪{(meas_2ns_vals[m-5]):6.2f}, diff = {ValErr(meas_2ns_vals[m-5],␣
↪meas_2ns_errs[m-5]).sigmadiff_fmt(l)}')
```

```
E_3s = -5.12745 ± 0.00401
```

```
Delta s = 1.37108 ± 0.00064
m =  4, lambda = 1170.366 ± 1.302
m =  5, lambda = 621.527 ± 0.692, lambda_measured = 622.34, diff = 0.67
m =  6, lambda = 518.112 ± 0.577, lambda_measured = 517.50, diff = 0.53
m =  7, lambda = 477.124 ± 0.531, lambda_measured = 474.00, diff = 2.76
m =  8, lambda = 456.100 ± 0.508, lambda_measured = 454.40, diff = 1.52
m =  9, lambda = 443.719 ± 0.494, lambda_measured = 445.30, diff = 1.42
```

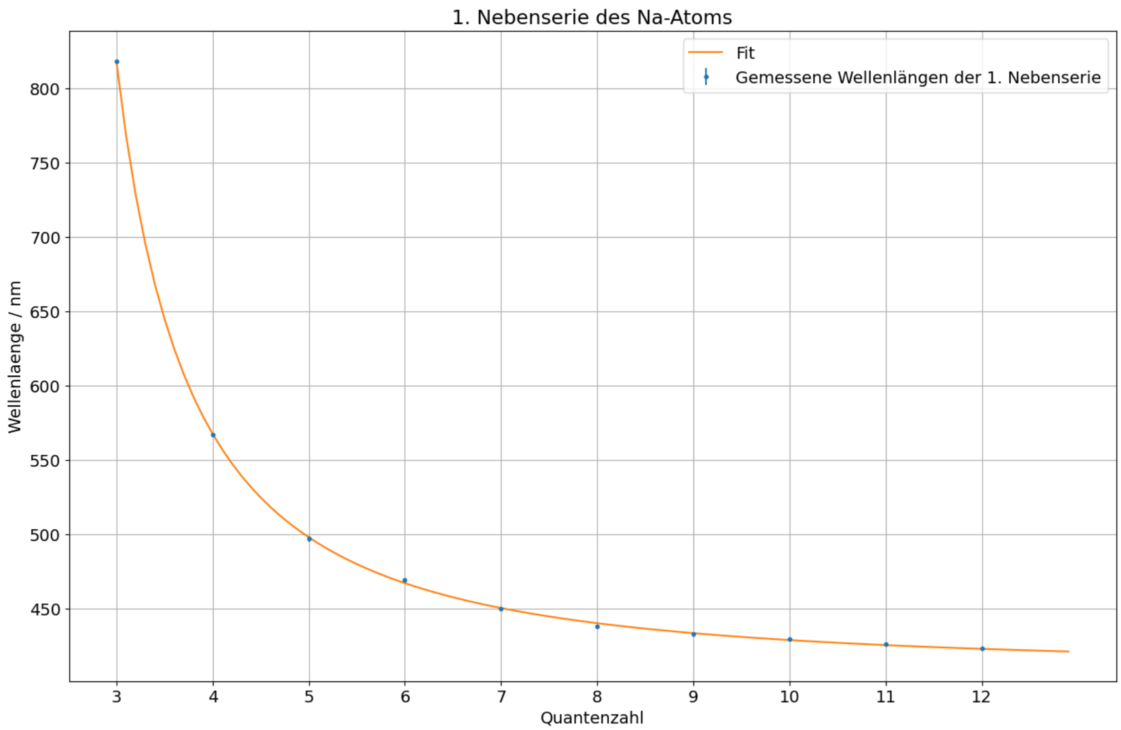
[37]:
```
# Erwartete Linien für die Hauptserie: mp -> 3s

Delta_p_val = 3 - np.sqrt(Consts.E_Ry / E_3p.val)
Delta_p_err = (np.sqrt(np.abs(Consts.E_Ry)) / (2 * np.abs(E_3p.val) ** (3/2)))␣
 ↪* E_3p.err
Delta_p = ValErr(Delta_p_val, Delta_p_err)

print_all(E_3s.strfmtf(5,0, 'E_3s'), Delta_p.strfmtf(5,0, 'Delta s'))

for m in range(4,6):
    l_val = Consts.hc / ((Consts.E_Ry / (m - Delta_p_val)**2) - E_3s.val)
    l_err = l_val * np.sqrt(E_3s.relerr()**2 + (2*Delta_p.relerr())**2)
    l = ValErr(l_val, l_err)

    print(f'm = {m:2d}, lambda = {l.strfmtf(3,0)}')
```

```
E_3s = -5.12745 ± 0.00401
Delta s = 0.88027 ± 0.00065
m =  4, lambda = 332.423 ± 0.555
m =  5, lambda = 286.603 ± 0.478
```

[38]:
```
# Bestimmung der Rydbergenergie, E_3p und Delta_d für die 1. Nebenserie

qz = np.arange(3,13)

plt.figure(figsize=(16,10))
plt.errorbar(qz, meas_1ns_vals, meas_1ns_errs, fmt='.', label='Gemessene␣
 ↪Wellenlängen der 1. Nebenserie')
plt.xlabel('Quantenzahl')
plt.ylabel('Wellenlaenge / nm')
plt.title('1. Nebenserie des Na-Atoms')
plt.xticks(qz)
plt.grid()



def fit_func_1ns(m, E_Ry, E_3p, D_d):
    return Consts.hc / ( E_Ry / (m-D_d)**2 - E_3p)

start_params = [-13.6,-3,-0.02]
```

```python
popt_1ns, pcov_1ns = curve_fit(fit_func_1ns, qz, meas_1ns_vals, sigma =
    meas_1ns_errs, p0 = start_params)

chi2_1ns = np.sum((fit_func_1ns(qz, *popt_1ns) - meas_1ns_vals)**2 /
    meas_1ns_errs**2)
dof_1ns = len(qz) - 3 #dof:degrees of freedom, Freiheitsgrad
chi2_red_1ns = chi2_1ns / dof_1ns
print("chi2 = ", chi2_1ns)
print("chi2_red = ", chi2_red_1ns)
prob_1ns = round(1 - chi2.cdf(chi2_1ns, dof_1ns), 2) * 100
print("Wahrscheinlichkeit:", prob_1ns, "%")

plt.plot(np.arange(3,13, 0.1), fit_func_1ns(np.arange(3,13, 0.1), *popt_1ns),
    label='Fit')
plt.legend()

plt.savefig("out/na_1ns_fit.png", format='png', bbox_inches='tight')

E_Ry_1ns = ValErr.fromFit(popt_1ns, pcov_1ns, 0)
E_3p_1ns = ValErr.fromFit(popt_1ns, pcov_1ns, 1)
D_d_1ns = ValErr.fromFit(popt_1ns, pcov_1ns, 2)

print_all(
    E_Ry_1ns.strfmtf(4, 0, 'E_Ry'),
    E_Ry_1ns.sigmadiff_fmt(ValErr(Consts.E_Ry, 0)),
    E_3p_1ns.strfmtf(4, 0, 'E_3p'),
    E_3p_1ns.sigmadiff_fmt(E_3p),
    D_d_1ns.strfmtf(4, 0, 'Δ_d'))
```

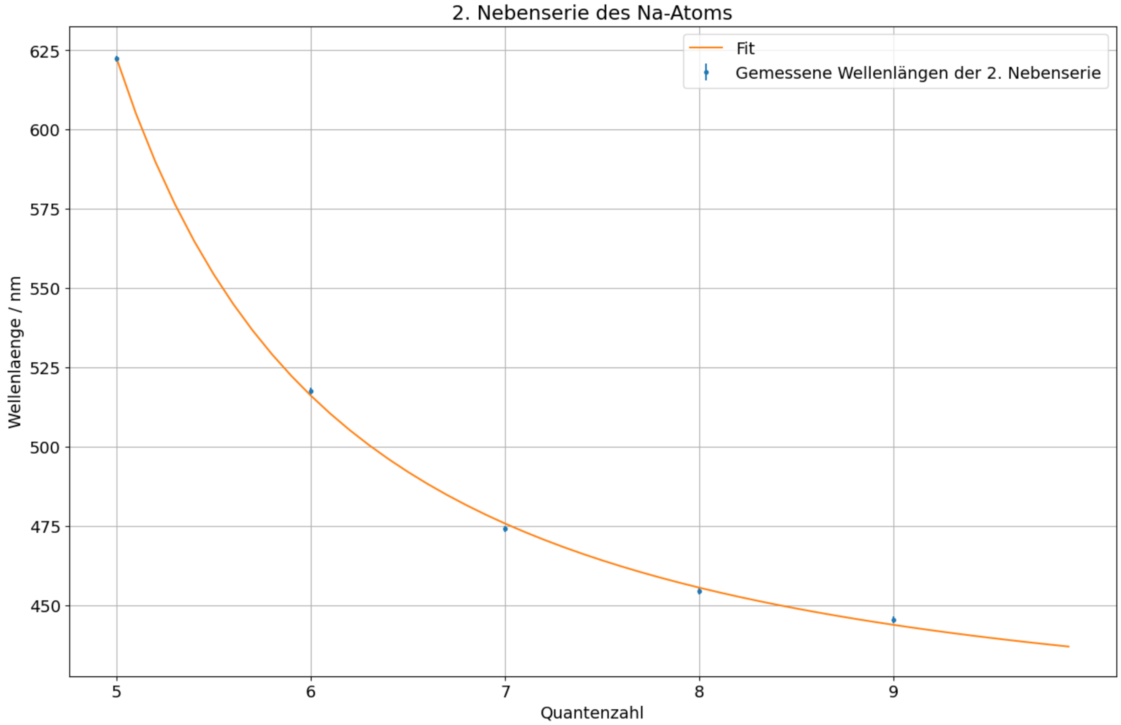
```
chi2 =  10.526902292499985
chi2_red =  1.503843184642855
Wahrscheinlichkeit: 16.0 %
E_Ry = -12.9778 ± 0.4102
1.53
E_3p = -3.0233 ± 0.0059
0.75
Δ_d = 0.0655 ± 0.0419
```



[39]:
```
# Bestimmung der Rydbergenergie, E_3p und Delta_d für die 2. Nebenserie

qz = np.arange(5,10)

plt.figure(figsize=(16,10))
plt.errorbar(qz, meas_2ns_vals, meas_2ns_errs, fmt='.', label='Gemessene␣
↪Wellenlängen der 2. Nebenserie')
plt.xlabel('Quantenzahl')
plt.ylabel('Wellenlaenge / nm')
plt.title('2. Nebenserie des Na-Atoms')
plt.xticks(qz)
plt.grid()


def fit_func_2ns(m, E_Ry, E_3p, D_s):
    return Consts.hc / ( E_Ry / (m-D_s)**2 - E_3p)

start_params = [-13.6,-3,1.5]
popt_2ns, pcov_2ns = curve_fit(fit_func_2ns, qz, meas_2ns_vals, sigma =␣
↪meas_2ns_errs, p0 = start_params)

chi2_2ns = np.sum((fit_func_2ns(qz, *popt_2ns) - meas_2ns_vals)**2 /␣
↪meas_2ns_errs**2)
```

```python
dof_2ns = len(qz) - 3 #dof:degrees of freedom, Freiheitsgrad
chi2_red_2ns = chi2_2ns / dof_2ns
print("chi2 = ", chi2_2ns)
print("chi2_red = ", chi2_red_2ns)
prob_2ns = round(1 - chi2.cdf(chi2_2ns, dof_2ns), 2) * 100
print("Wahrscheinlichkeit:", prob_2ns, "%")

plt.plot(np.arange(5,10, 0.1), fit_func_2ns(np.arange(5,10, 0.1), *popt_2ns),
 ↪label='Fit')
plt.legend()

plt.savefig("out/na_2ns_fit.png", format='png', bbox_inches='tight')

E_Ry_2ns = ValErr.fromFit(popt_2ns, pcov_2ns, 0)
E_3p_2ns = ValErr.fromFit(popt_2ns, pcov_2ns, 1)
D_s_2ns = ValErr.fromFit(popt_2ns, pcov_2ns, 2)

print_all(
    E_Ry_2ns.strfmtf(4, 0, 'E_Ry'),
    E_Ry_2ns.sigmadiff_fmt(ValErr(Consts.E_Ry, 0)),
    E_3p_2ns.strfmtf(4, 0, 'E_3p'),
    E_3p_2ns.sigmadiff_fmt(E_3p),
    D_s_2ns.strfmtf(4, 0, 'Δ_s'),
    D_s_2ns.sigmadiff_fmt(Delta_s))
```

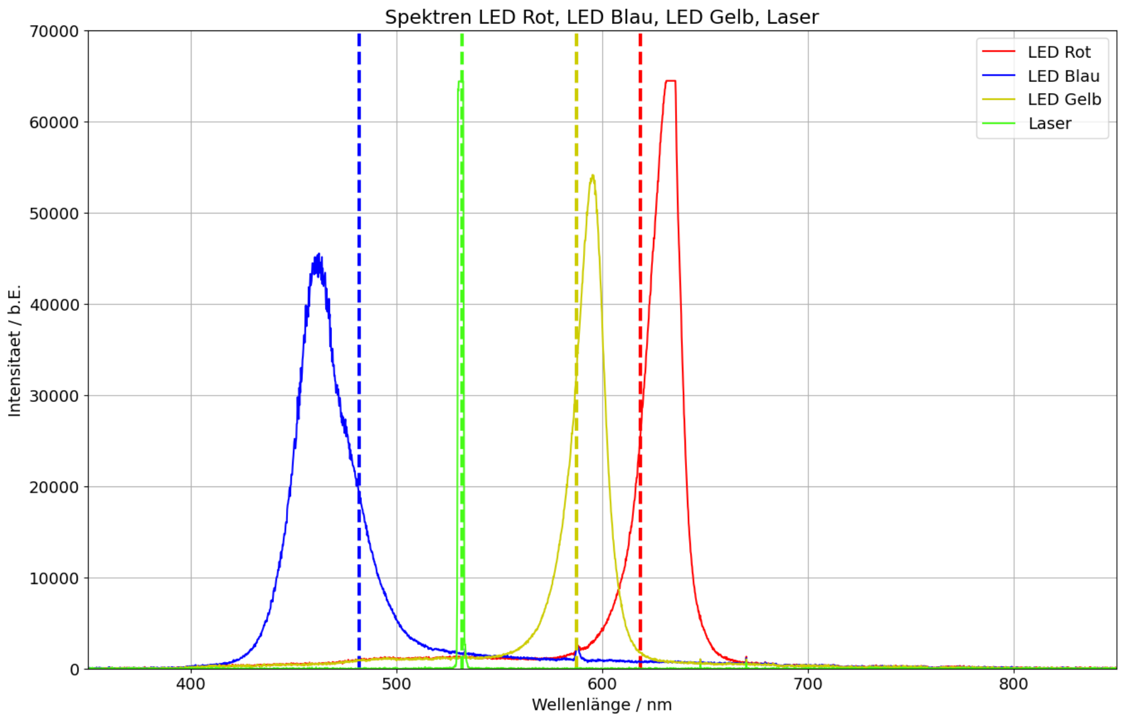
```
chi2 =  8.553390527826295
chi2_red =  4.2766952639131475
Wahrscheinlichkeit: 1.0 %
E_Ry = -11.5866 ± 2.0081
1.01
E_3p = -3.0068 ± 0.0313
0.68
Δ_s = 1.6217 ± 0.2454
1.03
```

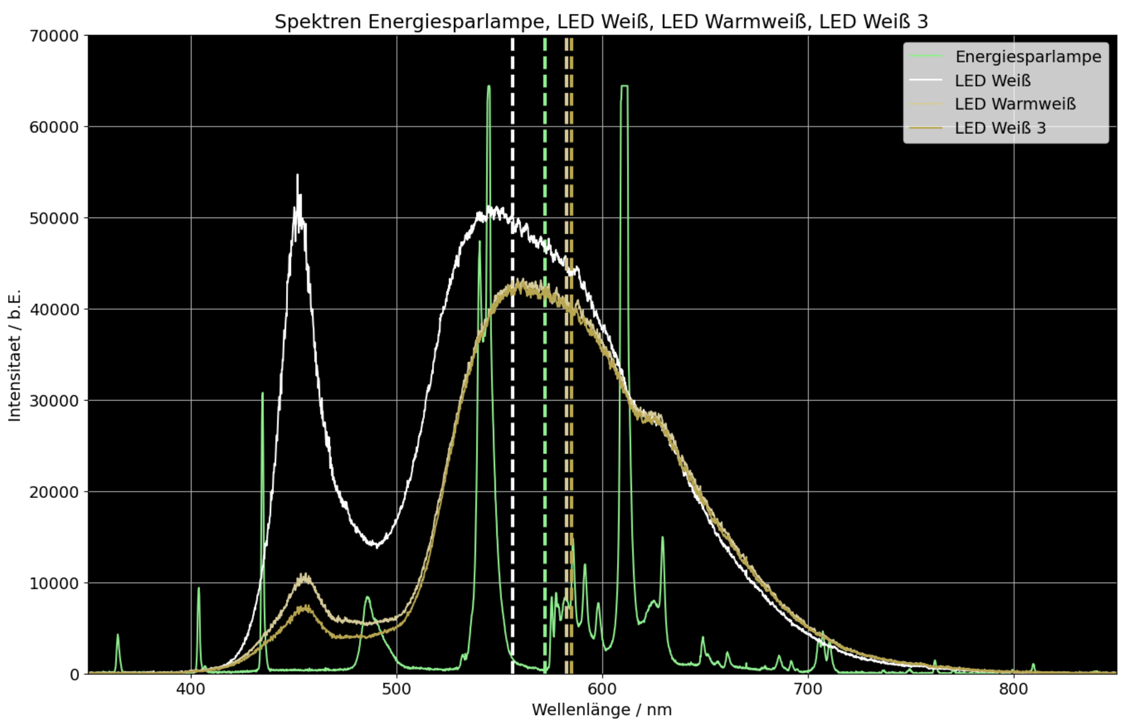


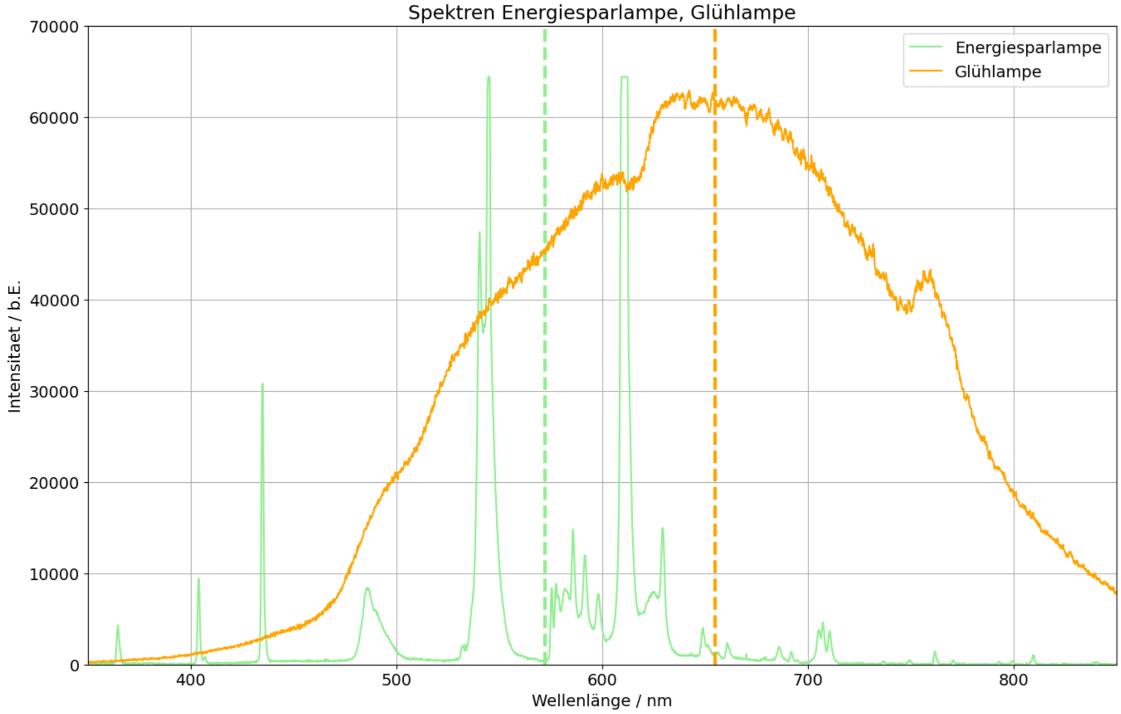
0.0.3 LED Spektren

[40]:
```
all_spectra = [
    ('LED Weiß', 'spektrum_led_weiss.txt', 'white'),
    ('LED Warmweiß', 'spektrum_led_warmweiss.txt', '#d6cb9a'),
    ('LED Weiß 3', 'spektrum_led_weiss3.txt', '#b8a651'),
    ('LED Rot', 'spektrum_led_rot.txt', 'red'),
    ('LED Blau', 'spektrum_led_blau.txt', 'blue'),
    ('LED Gelb', 'spektrum_led_gelb.txt', '#cccc00'),
    ('LED Orange', 'spektrum_led_orange.txt', 'orange'),
    ('Energiesparlampe', 'energiesparlampe.txt', 'lightgreen'),
    ('Glühlampe', 'gluehlampe.txt', 'orange'),
    ('Laser', 'laser.txt', '#47fb18'),
]

def draw_spectra(spectra):
    plt.figure(figsize=(16,10))
    for spec in spectra:

        spec_lam, spec_int = np.loadtxt(spec[1], skiprows=17,
                                        converters= {0:comma_to_float, 1:
 ↪comma_to_float},
                                        comments='>', unpack=True)
```

```
        plt.plot(spec_lam, spec_int, color=spec[2], label=spec[0])

        wavelen_sum = 0
        inten_sum = 0
        for si in zip(spec_lam, spec_int):
            if si[1] > 0:
                wavelen_sum = wavelen_sum + si[0] * si[1]
                inten_sum = inten_sum + si[1]

        wavelen_mean = wavelen_sum / inten_sum

        plt.axvline(x=wavelen_mean, linewidth=3, linestyle='--', color=spec[2])


    plt.title('Spektren ' + ', '.join([s[0] for s in spectra]))
    plt.xlabel('Wellenlänge / nm')
    plt.ylabel('Intensitaet / b.E.')
    plt.legend()
    plt.grid()
    plt.ylim((0,70000))
    plt.xlim((350,850))


draw_spectra(all_spectra[3:6] + all_spectra[9:10])
plt.savefig('out/led_vergleich.png', format='png', bbox_inches='tight')

draw_spectra(all_spectra[7:8] + all_spectra[0:3])
ax = plt.gca()
ax.set_facecolor('black')
plt.savefig('out/led_und_energiespar.png', format='png', bbox_inches='tight')

draw_spectra(all_spectra[7:9])
plt.savefig('out/energiespar_und_glueh.png', format='png', bbox_inches='tight')
```


Spektren LED Rot, LED Blau, LED Gelb, Laser
LED Rot
LED Blau
LED Gelb
Laser
Intensitaet / b.E.
Wellenlänge / nm


Spektren Energiesparlampe, LED Weiß, LED Warmweiß, LED Weiß 3
Energiesparlampe
LED Weiß
LED Warmweiß
LED Weiß 3
Intensitaet / b.E.
Wellenlänge / nm


Spektren Energiesparlampe, Glühlampe
Energiesparlampe
Glühlampe
Intensitaet / b.E.
Wellenlänge / nm
70000
60000
50000
40000
30000
20000
10000
0
400
500
600
700
800

```
[ ]:
```