255auswertung

April 9, 2025

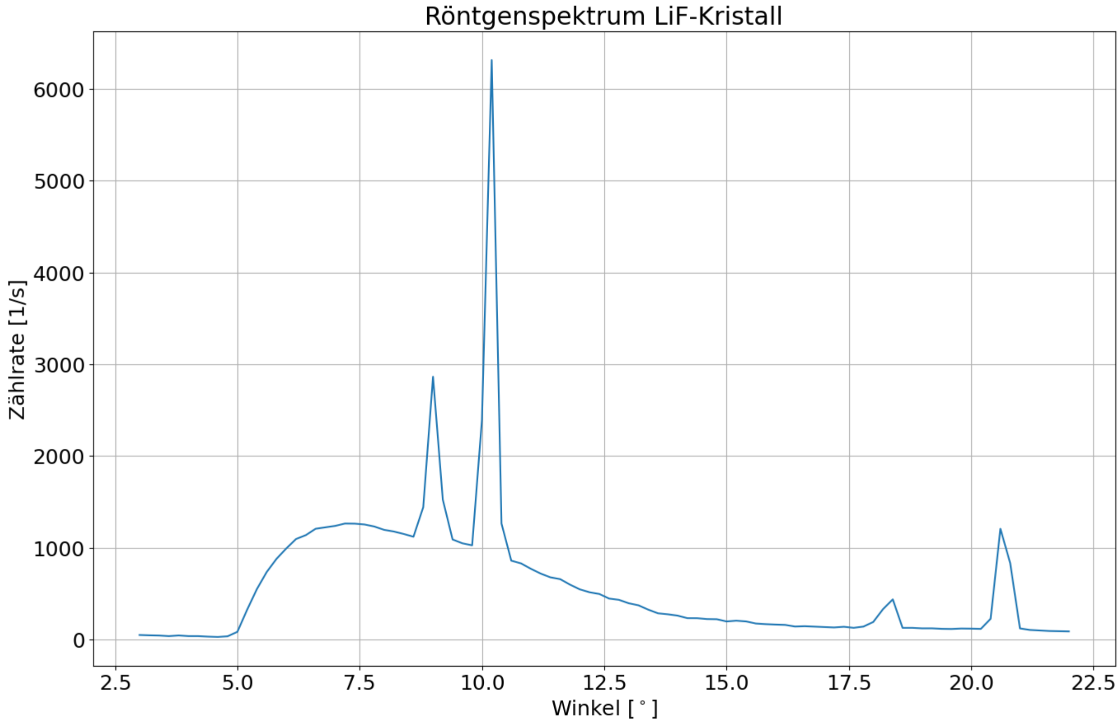
[37]:
```python
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
from scipy.signal import argrelextrema
from scipy.optimize import curve_fit
from scipy.stats import chi2

plt.rcParams.update({'font.size': 18})

%run ../lib.ipynb
LibFormatter.OutputType = 'latex'

class Consts:
    hc = 1.2398 * 10**(3) #nm eV
    E_Ry = -13.605 # eV
    E_inf = 3.2898 * 10**(15) # Hz
    c = 2.9979 * 10**(8)  # m/s
    e = 1.6022 * 10**(-19) # C
    h =  6.6261 * 10**(-34) # Js
    N_A = 6.0221 * 10**(23)
```

[38]:
```python
def comma_to_float(valstr):
    return float(valstr.replace(',','.'))

def countrate_error(ctrt, t):
    return [np.sqrt(x*t) / t for x in ctrt]
```

[39]:
```python
# fit functions
def gauss_func(x, A, mu, sig, c):
    return A/(np.sqrt(2 * np.pi)*sig)*np.exp(-(x-mu)**2 / (2 * sig**2)) + c

def linear_func(x, a, b):
    return a * x + b
```

[40]:
```python
lif_ang, lif_countrate = np.loadtxt('aufgabe1.txt', skiprows=0,
                        converters= {0:comma_to_float, 1:comma_to_float},
                        comments='>', unpack=True)
```

[41]:
```
plt.figure(figsize=(16,10))
plt.plot(lif_ang, lif_countrate)
plt.title('Röntgenspektrum LiF-Kristall')
plt.xlabel(r'Winkel [$^\circ$]')
plt.ylabel(r'Zählrate [1/s]')
plt.grid()
plt.savefig('out/spektrum_lif_komplett.png', format='png', bbox_inches='tight')
```



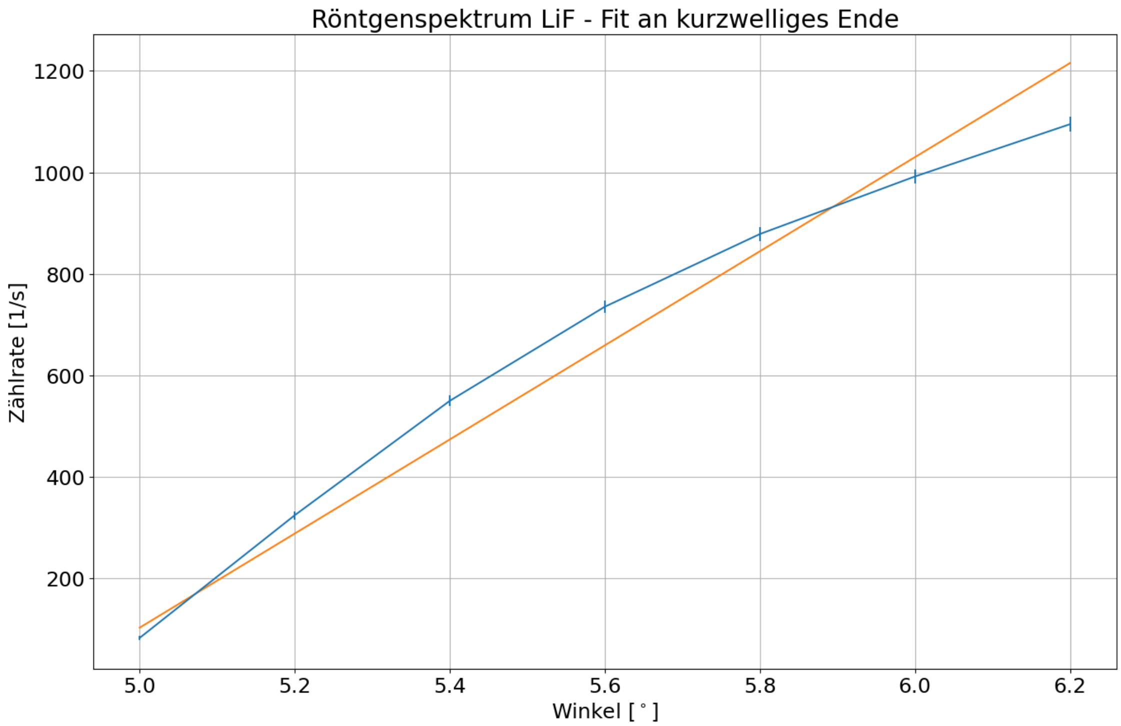
0.0.1 Aufgabe 1

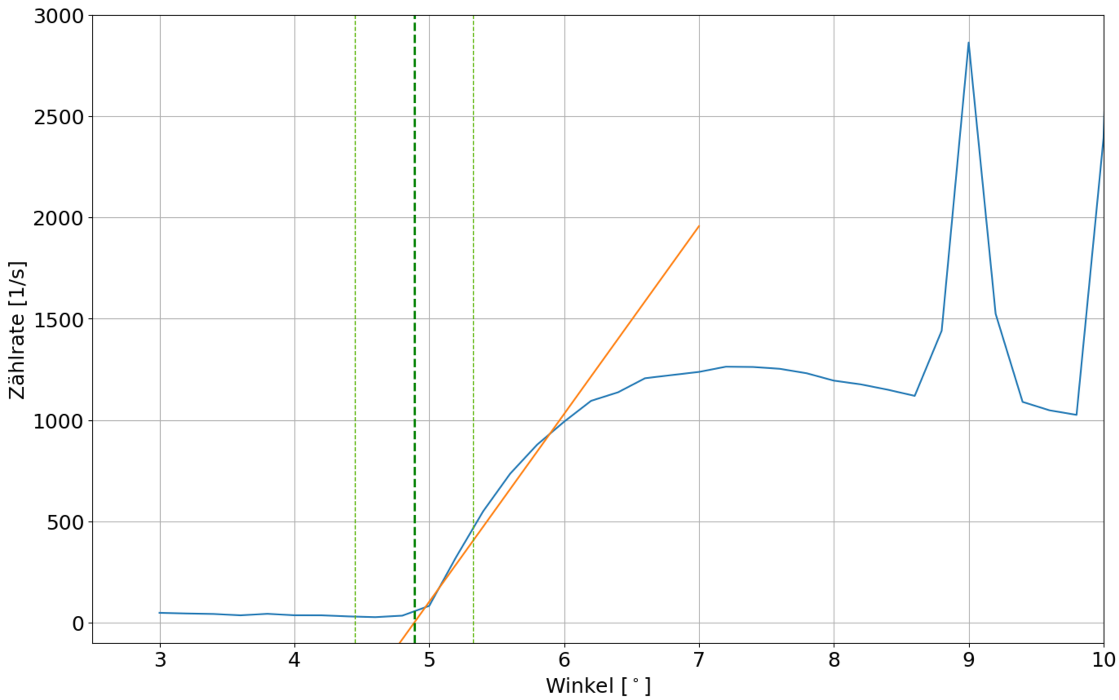
[42]:
```
plt.figure(figsize=(16,10))

lowlim = 10
uplim = 17

lif_countrate_errs = countrate_error(lif_countrate[lowlim:uplim], 5)

plt.errorbar(lif_ang[lowlim:uplim], lif_countrate[lowlim:uplim],␣
↪yerr=lif_countrate_errs)
plt.title('Röntgenspektrum LiF - Fit an kurzwelliges Ende')
plt.xlabel(r'Winkel [$^\circ$]')
plt.ylabel(r'Zählrate [1/s]')
#plt.ylim((0,3000))
```

```
plt.grid()


start_params = [0, 0]
popt_lif_lgrenz_lin, pcov_lif_lgrenz_lin = curve_fit(linear_func,␣
 ↪lif_ang[lowlim:uplim], lif_countrate[lowlim:uplim], sigma =␣
 ↪lif_countrate_errs, p0 = start_params)
lif_lin_a = ValErr.fromFit(popt_lif_lgrenz_lin, pcov_lif_lgrenz_lin, 0)
lif_lin_b = ValErr.fromFit(popt_lif_lgrenz_lin, pcov_lif_lgrenz_lin, 1)


# 0 = a x + b <=> x = -b/a
lif_lin_nullst = (-1) * (lif_lin_b / lif_lin_a)

plt.plot(lif_ang[lowlim:uplim], linear_func(lif_ang[lowlim:uplim],␣
 ↪*popt_lif_lgrenz_lin))
plt.savefig('out/spektrum_lif_linear_fit.png', format='png',␣
 ↪bbox_inches='tight')

plt.figure(figsize=(16,10))
plt.plot(lif_ang, lif_countrate)
plt.plot(lif_ang[lowlim-4:uplim+4], linear_func(lif_ang[lowlim-4:uplim+4],␣
 ↪*popt_lif_lgrenz_lin))
plt.axvline(x=lif_lin_nullst.val, linewidth=2, linestyle='--', color='green')
plt.axvline(x=lif_lin_nullst.val+lif_lin_nullst.err, linewidth=1,␣
 ↪linestyle='--', color='#56b300')
plt.axvline(x=lif_lin_nullst.val-lif_lin_nullst.err, linewidth=1,␣
 ↪linestyle='--', color='#56b300')
plt.ylim((-100,3000))
plt.xlim((2.5,10))
plt.xlabel(r'Winkel [$^\circ$]')
plt.ylabel(r'Zählrate [1/s]')
plt.grid()

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

# 2d sin(theta) = n lambda <=> lambda = 2d sin(theta) / n (n = 1)
# theta = arcsin (n lambda / 2 d)

lif_d = 201.4 * 10**(-12)

lif_lam_grenz_val = 2 * lif_d * np.sin(np.deg2rad(lif_lin_nullst.val))
lif_lam_grenz_err = 2 * lif_d * np.cos(np.deg2rad(lif_lin_nullst.val)) * np.
 ↪deg2rad(lif_lin_nullst.err)
lif_lam_grenz = ValErr(lif_lam_grenz_val, lif_lam_grenz_err)
```

```python
# l_grenz = hc / Ue <=> h = l_grenz U e / c
U_a1 = 35 * 10**3
h_a1 = lif_lam_grenz * (U_a1 * Consts.e / Consts.c)

# theta(n=2) = arcsin(lambda/d)

lif_ang_2ord_val = np.rad2deg(np.arcsin(lif_lam_grenz.val / lif_d))
lif_ang_2ord_err = (lif_lam_grenz.err / lif_d) * (1 / np.sqrt(1 -␣
↪(lif_lam_grenz.val**2/lif_d**2)))
lif_ang_2ord = ValErr(lif_ang_2ord_val, lif_ang_2ord_err)

print_all(
    lif_lin_a,
    lif_lin_b,
    lif_lin_nullst.strfmtf2(6, 0, 'beta'),
    lif_lam_grenz.strfmtf2(6, -12, ' _grenz'),
    h_a1.strfmtf2(6, -34, 'h'),
    h_a1.sigmadiff_fmt(ValErr(Consts.h, 0)),
    lif_ang_2ord.strfmtf2(6, 0, 'beta, 2. ord'))
```

```
ValErr(926.9388981212614, 56.62078398042604)
ValErr(-4531.531907160432, 297.4967190001228)
beta = 4.888706 \pm 0.438383
 _grenz = (34.326828 \pm 3.070703) \cdot 10^{-12}
h = (6.420980 \pm 0.574388) \cdot 10^{-34}
0.36
beta, 2. ord = 9.813464 \pm 0.015473
```


Röntgenspektrum LiF - Fit an kurzwelliges Ende
Zählrate [1/s]
Winkel [°]
200
400
600
800
1000
1200
5.0
5.2
5.4
5.6
5.8
6.0
6.2


Zählrate [1/s]
Winkel [°]
0
500
1000
1500
2000
2500
3000
3
4
5
6
7
8
9
10

0.0.2 Aufgabe 2

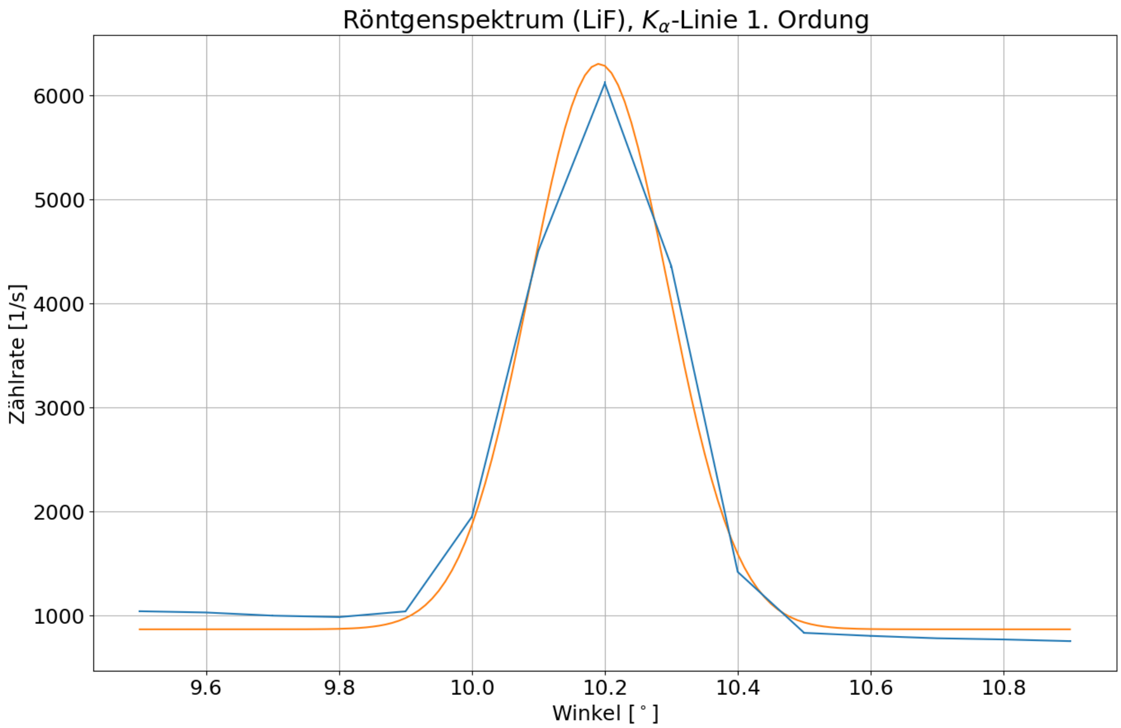
[43]:
```
lif_kalph1_ang, lif_kalph1_countrate = np.loadtxt('aufgabe2/kalpha1.txt',␣
↪skiprows=0,
                        converters= {0:comma_to_float, 1:comma_to_float},
                        comments='>', unpack=True)

lif_kbeta1_ang, lif_kbeta1_countrate = np.loadtxt('aufgabe2/kbeta1.txt',␣
↪skiprows=0,
                        converters= {0:comma_to_float, 1:comma_to_float},
                        comments='>', unpack=True)

lif_kalph2_ang, lif_kalph2_countrate = np.loadtxt('aufgabe2/kalpha2.txt',␣
↪skiprows=0,
                        converters= {0:comma_to_float, 1:comma_to_float},
                        comments='>', unpack=True)

lif_kbeta2_ang, lif_kbeta2_countrate = np.loadtxt('aufgabe2/kbeta2.txt',␣
↪skiprows=0,
                        converters= {0:comma_to_float, 1:comma_to_float},
                        comments='>', unpack=True)

def find_peak(peak_ang, peak_countrate, peak_order, offset, title, fname):
    pf_lowlim = 0 + offset
    pf_uplim = len(peak_ang) - offset

    peak_countrate_errs = countrate_error(peak_countrate[pf_lowlim:pf_uplim],␣
↪20)

    plt.figure(figsize=(16,10))
    plt.errorbar(peak_ang[pf_lowlim:pf_uplim], peak_countrate[pf_lowlim:
↪pf_uplim], yerr=peak_countrate_errs)
    plt.title(title)
    plt.xlabel(r'Winkel [$^\circ$]')
    plt.ylabel(r'Zählrate [1/s]')
    plt.grid()

    start_params = [500, np.mean(peak_ang[pf_lowlim:pf_uplim]), np.
↪std(peak_ang[pf_lowlim:pf_uplim]), np.min(peak_countrate[pf_lowlim:
↪pf_uplim])]
    #print(start_params)

    popt_peak, pcov_peak = curve_fit(
        gauss_func,
        peak_ang[pf_lowlim:pf_uplim],
        peak_countrate[pf_lowlim:pf_uplim],
        sigma = peak_countrate_errs,
```

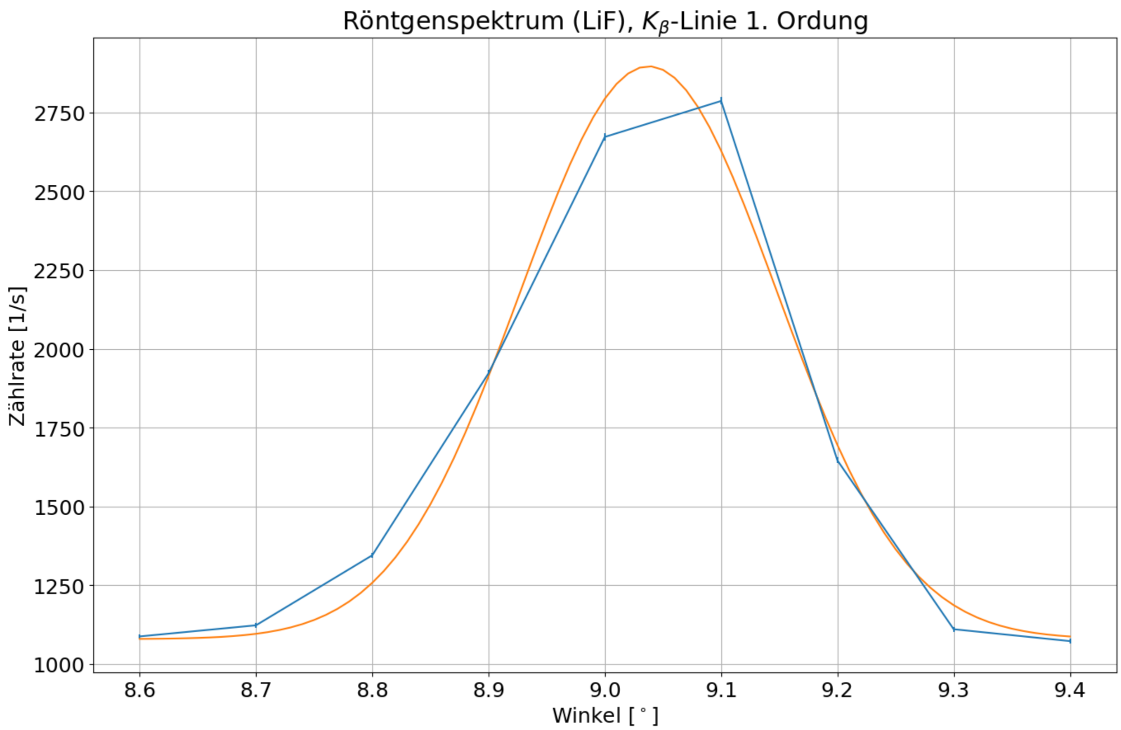
```
        p0 = start_params)



    peak_A = ValErr.fromFit(popt_peak, pcov_peak, 0)
    peak_mu = ValErr.fromFit(popt_peak, pcov_peak, 1)
    peak_sig = ValErr.fromFit(popt_peak, pcov_peak, 2)
    peak_c = ValErr.fromFit(popt_peak, pcov_peak, 3)

    peak_fwhm = 2 * np.sqrt(2 * np.log(2)) * peak_sig

    print(peak_fwhm.val, peak_fwhm.val / 2, peak_sig.val, peak_sig.val / np.
 ↪sqrt(len(peak_ang[pf_lowlim:pf_uplim])))

    angle_err = peak_sig.val / np.sqrt(len(peak_ang[pf_lowlim:pf_uplim]))

    contin_angle = np.arange(peak_ang[pf_lowlim], peak_ang[pf_uplim-1], 0.01)
    plt.plot(contin_angle, gauss_func(contin_angle, *popt_peak))
    #plt.axvline(x=peak_mu.val, linewidth=2, linestyle='--', color='green')
    #plt.axvline(x=peak_mu.val + angle_err, linewidth=1, linestyle='--',␣
 ↪color='#56b300')
    #plt.axvline(x=peak_mu.val - angle_err, linewidth=1, linestyle='--',␣
 ↪color='#56b300')

    plt.savefig(f'out/{fname}.png', format='png', bbox_inches='tight')

    peak_lam_val = (2 * 201.4 * 10**(-12) / peak_order) * np.sin(np.
 ↪deg2rad(peak_mu.val))
    peak_lam_err = (2 * 201.4 * 10**(-12) / peak_order) * np.cos(np.
 ↪deg2rad(peak_mu.val)) * np.deg2rad(angle_err)
    peak_lam = ValErr(peak_lam_val, peak_lam_err)

    print_all(title,
              peak_A.strfmtf(4, 0, 'A'),
              peak_mu.strfmtf(4, 0, ' '),
              peak_sig.strfmtf(4, 0, ' '),
              peak_fwhm.strfmtf(4, 0, 'fwhm'),
              peak_lam.strfmtf2(6, -12, ' '),
             '')

    return (peak_A, peak_mu, peak_sig, peak_c, peak_lam)

peak_dat_kalpha1 = find_peak(lif_kalph1_ang, lif_kalph1_countrate, 1, 1,␣
 ↪r'Röntgenspektrum (LiF), $K_{\alpha}$-Linie 1. Ordung', 'lif_kalpha_1ord')
peak_dat_kbeta1 = find_peak(lif_kbeta1_ang, lif_kbeta1_countrate, 1, 1,␣
 ↪r'Röntgenspektrum (LiF), $K_{\beta}$-Linie 1. Ordung', 'lif_kbeta_1ord')
```

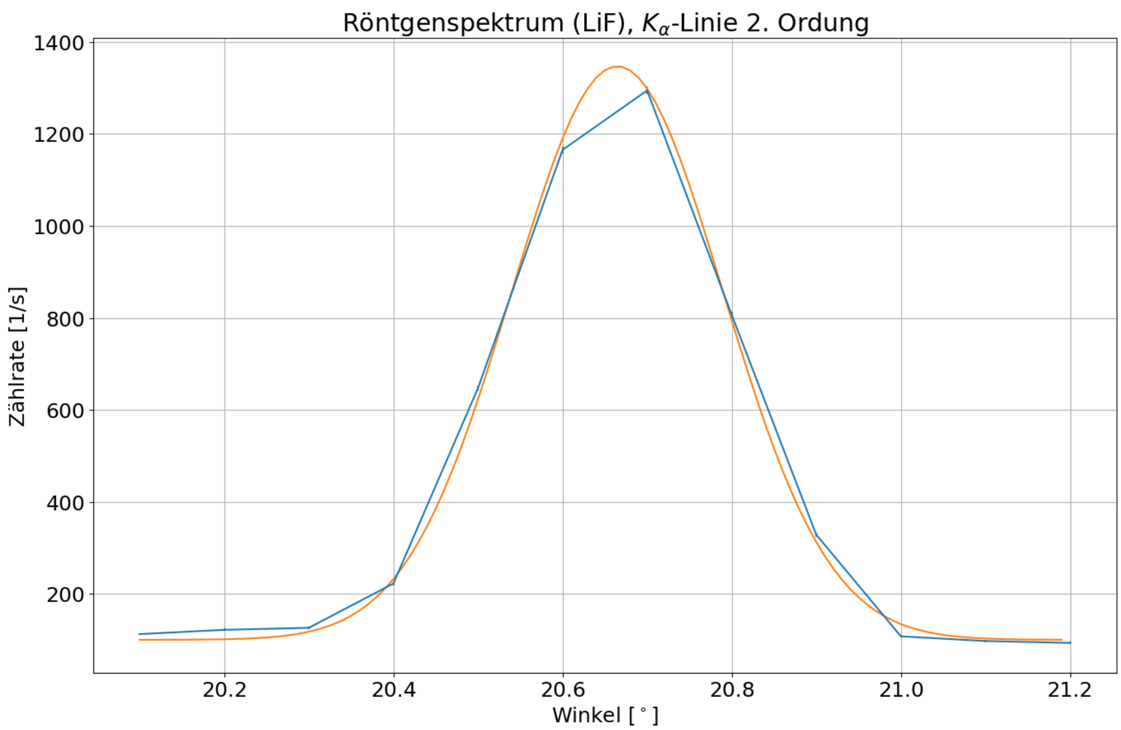
```python
peak_dat_kalpha2 = find_peak(lif_kalph2_ang, lif_kalph2_countrate, 2, 0,␣
 ↪r'Röntgenspektrum (LiF), $K_{\alpha}$-Linie 2. Ordung', 'lif_kalpha_2ord')
peak_dat_kbeta2 = find_peak(lif_kbeta2_ang, lif_kbeta2_countrate, 2, 0,␣
 ↪r'Röntgenspektrum (LiF), $K_{\beta}$-Linie 2. Ordung', 'lif_kbeta_2ord')

kalpha_mean_lit = ValErr(71.1 * 10**(-12))
kbeta_mean_lit = ValErr(63.1 * 10**(-12))

kalpha_mean = (peak_dat_kalpha1[4] + peak_dat_kalpha2[4]) / 2
kbeta_mean = (peak_dat_kbeta1[4] + peak_dat_kbeta2[4]) / 2

print_all('Mittelwerte', kalpha_mean.strfmtf2(6, -12, '  K_ '), kalpha_mean.
 ↪sigmadiff_fmt(kalpha_mean_lit), kbeta_mean.strfmtf2(6, -12, '  K_ '),␣
 ↪kbeta_mean.sigmadiff_fmt(kbeta_mean_lit))
```

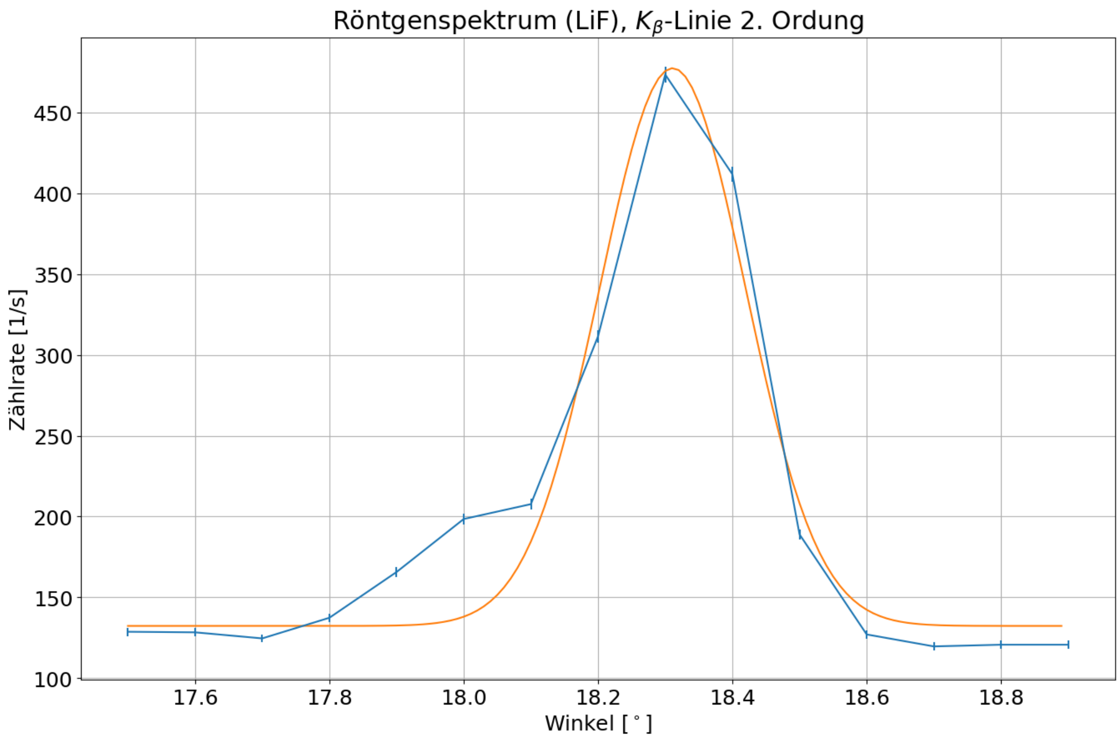
```
0.2449426466010601 0.12247132330053005 0.1040175647892622 0.0268572197427909
Röntgenspektrum (LiF), $K_{\alpha}$-Linie 1. Ordung
A = 1417.9660 \pm 64.8681
  = 10.1912 \pm 0.0049
  = 0.1040 \pm 0.0045
fwhm = 0.2449 \pm 0.0105
  = (71.268530 \pm 0.185832) \cdot 10^{-12}

0.2598405224074471 0.12992026120372355 0.11034411013943615 0.03678137004647872
Röntgenspektrum (LiF), $K_{\beta}$-Linie 1. Ordung
A = 503.0489 \pm 37.1957
  = 9.0377 \pm 0.0064
  = 0.1103 \pm 0.0075
fwhm = 0.2598 \pm 0.0176
  = (63.273447 \pm 0.255370) \cdot 10^{-12}

0.29438118966863097 0.14719059483431549 0.12501218099014522 0.03608790817332128
Röntgenspektrum (LiF), $K_{\alpha}$-Linie 2. Ordung
A = 390.5916 \pm 10.7041
  = 20.6648 \pm 0.0033
  = 0.1250 \pm 0.0031
fwhm = 0.2944 \pm 0.0074
  = (71.074234 \pm 0.118691) \cdot 10^{-12}

0.2556687909958171 0.12783439549790854 0.10857253892301431 0.028033309006952733
Röntgenspektrum (LiF), $K_{\beta}$-Linie 2. Ordung
A = 93.8456 \pm 9.1049
  = 18.3110 \pm 0.0106
  = 0.1086 \pm 0.0104
fwhm = 0.2557 \pm 0.0246
  = (63.274851 \pm 0.093550) \cdot 10^{-12}
```

```
Mittelwerte
  K_  = (71.171382 \pm 0.110251) \cdot 10^{-12}
0.65
  K_  = (63.274149 \pm 0.135983) \cdot 10^{-12}
1.29
```




Röntgenspektrum (LiF), K_β-Linie 1. Ordung
Zählrate [1/s]
Winkel [$^\circ$]
2750
2500
2250
2000
1750
1500
1250
1000
8.6
8.7
8.8
8.9
9.0
9.1
9.2
9.3
9.4


Röntgenspektrum (LiF), K_α-Linie 2. Ordung
Zählrate [1/s]
Winkel [$^\circ$]
1400
1200
1000
800
600
400
200
20.2
20.4
20.6
20.8
21.0
21.2



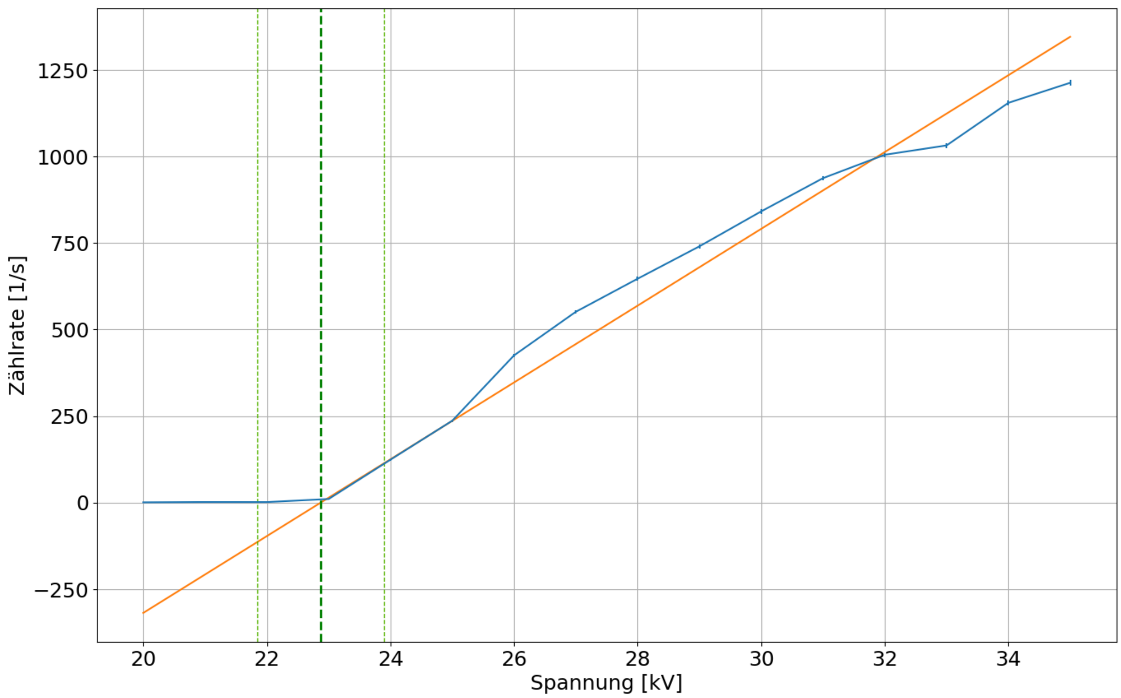
0.0.3 Aufgabe 3

[44]:
```
lif_ang75_volt, lif_ang75_countrate = np.loadtxt('aufgabe3.txt', skiprows=0,
                        converters= {0:comma_to_float, 1:comma_to_float},
                        comments='>', unpack=True)
```

[45]:
```
lowlim_einsp = 3
uplin_einsp = len(lif_ang75_volt)

lif_ang75_countrate_errs = countrate_error(lif_ang75_countrate, 20)

plt.figure(figsize=(16,10))
plt.errorbar(lif_ang75_volt, lif_ang75_countrate, yerr=lif_ang75_countrate_errs)
#plt.title(title)
plt.xlabel(r'Spannung [kV]')
plt.ylabel(r'Zählrate [1/s]')
plt.grid()

start_params = [0, 0]
popt_lif_einsp, pcov_lif_einsp = curve_fit(
    linear_func,
    lif_ang75_volt[lowlim_einsp:uplin_einsp],
    lif_ang75_countrate[lowlim_einsp:uplin_einsp],
```

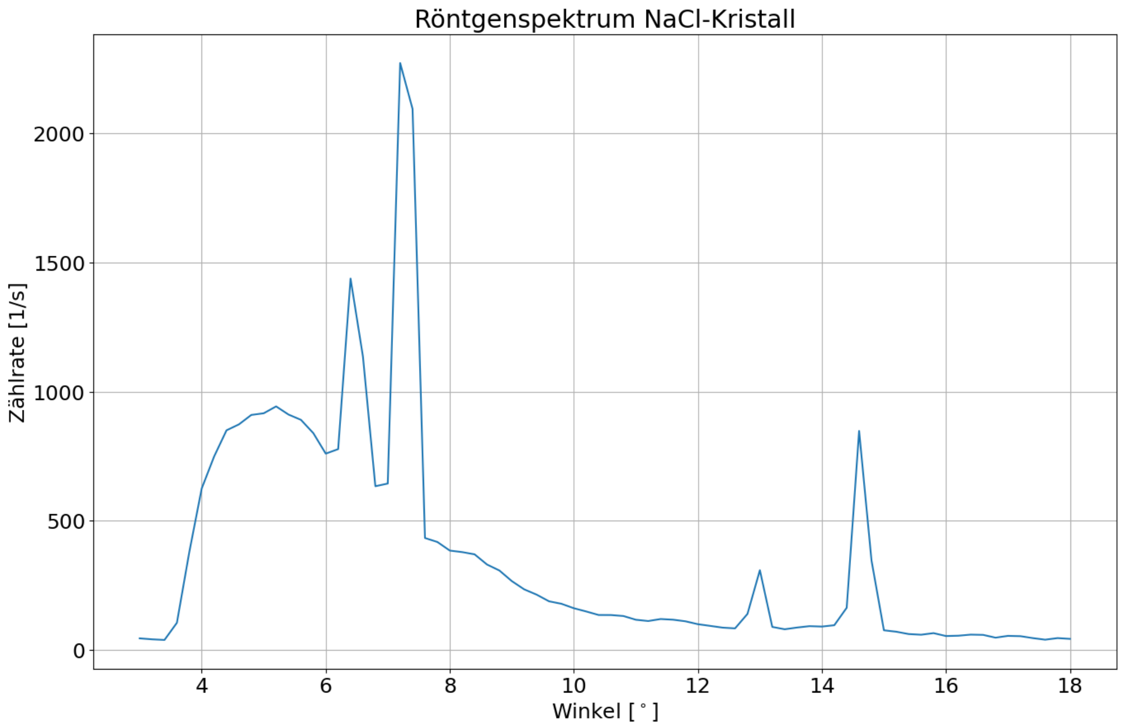
```
    sigma = lif_ang75_countrate_errs[lowlim_einsp:uplin_einsp],
    p0 = start_params)

lif_einsp_a = ValErr.fromFit(popt_lif_einsp, pcov_lif_einsp, 0)
lif_einsp_b = ValErr.fromFit(popt_lif_einsp, pcov_lif_einsp, 1)



plt.plot(lif_ang75_volt, linear_func(lif_ang75_volt, *popt_lif_einsp))

# 0 = a x + b <=> x = -b/a
lif_einsp_nullst = (-1) * (lif_einsp_b / lif_einsp_a)

plt.axvline(x=lif_einsp_nullst.val, linewidth=2, linestyle='--', color='green')
plt.axvline(x=lif_einsp_nullst.val+lif_einsp_nullst.err, linewidth=1,
 ↪linestyle='--', color='#56b300')
plt.axvline(x=lif_einsp_nullst.val-lif_einsp_nullst.err, linewidth=1,
 ↪linestyle='--', color='#56b300')

plt.savefig(f'out/lif_ang75_volt_fit.png', format='png', bbox_inches='tight')

# 2d sin(theta) = n lambda <=> lambda = 2d sin(theta) / n (n = 1)

lif_einsp_lam_val = 2 * 201.4 * 10**(-12) * np.sin(np.deg2rad(7.5))
lif_einsp_lam_err = 2 * 201.4 * 10**(-12) * np.cos(np.deg2rad(7.5)) * np.
 ↪deg2rad(0.05)
lif_einsp_lam = ValErr(lif_einsp_lam_val, lif_einsp_lam_err)

# l_grenz = hc / Ue <=> h = l_grenz U e / c
U_a3 = lif_einsp_nullst * 10**3
h_a3 = lif_einsp_lam * (U_a3 * Consts.e / Consts.c)

print_all(lif_einsp_a,
          lif_einsp_b,
          lif_einsp_nullst.strfmtf2(6, 0, 'U'),
          lif_einsp_lam.strfmtf2(6, -12, ' '),
          h_a3.strfmtf2(6, -34, 'h'),
          h_a3.sigmadiff_fmt(ValErr(Consts.h, 0)))
```

```
ValErr(110.96963341748018, 3.434413817817252)
ValErr(-2538.133262327901, 82.21087923807104)
U = 22.872323 \pm 1.024665
  = (52.575950 \pm 0.348502) \cdot 10^{-12}
h = (6.426833 \pm 0.291052) \cdot 10^{-34}
0.69
```



0.0.4 Aufgabe 4

[46]:
```
nacl_ang, nacl_countrate = np.loadtxt('aufgabe4.txt', skiprows=0,
                        converters= {0:comma_to_float, 1:comma_to_float},
                        comments='>', unpack=True)

plt.figure(figsize=(16,10))
plt.plot(nacl_ang, nacl_countrate)
plt.title(r'Röntgenspektrum NaCl-Kristall')
plt.xlabel(r'Winkel [$^\circ$]')
plt.ylabel(r'Zählrate [1/s]')
plt.grid()
plt.savefig('out/spektrum_nacl_komplett.png', format='png', bbox_inches='tight')
```

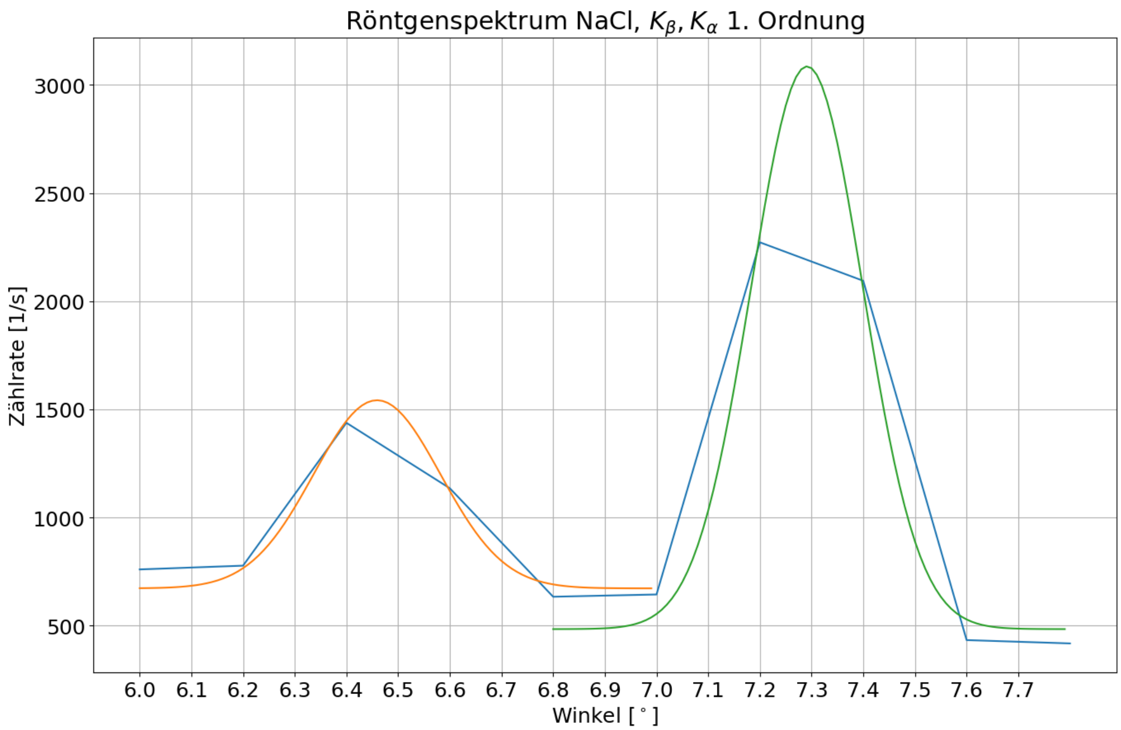


[47]:
```
nacl_lowlim = 15
nacl_uplim = 25

plt.figure(figsize=(16,10))
plt.plot(nacl_ang[nacl_lowlim:nacl_uplim], nacl_countrate[nacl_lowlim:
 ↪nacl_uplim])
plt.title(r'Röntgenspektrum NaCl, $K_{\beta}, K_{\alpha}$ 1. Ordnung')
plt.xlabel(r'Winkel [$^\circ$]')
plt.ylabel(r'Zählrate [1/s]')
plt.xticks(np.arange(nacl_ang[nacl_lowlim], nacl_ang[nacl_uplim-1], 0.1))
plt.grid()
#plt.savefig('out/himmel_m_o_g.png', format='png', bbox_inches='tight')

def find_nacl_peak(nacl_peak_low, nacl_peak_up):

    peak_countrate_errs = countrate_error(nacl_countrate[nacl_peak_low:
 ↪nacl_peak_up], 5)

    start_params = [500, np.mean(nacl_ang[nacl_peak_low:nacl_peak_up]), np.
 ↪std(nacl_ang[nacl_peak_low:nacl_peak_up]), 500]

    popt_nacl_k_peak, pcov_nacl_k_peak = curve_fit(
        gauss_func,
```

```
        nacl_ang[nacl_peak_low:nacl_peak_up],
        nacl_countrate[nacl_peak_low:nacl_peak_up],
        sigma=peak_countrate_errs,
        p0 = start_params)

    contin_angle = np.arange(nacl_ang[nacl_peak_low], nacl_ang[nacl_peak_up-1], 0.01)
    plt.plot(contin_angle, gauss_func(contin_angle, *popt_nacl_k_peak))

    peak_mu = ValErr.fromFit(popt_nacl_k_peak, pcov_nacl_k_peak, 1)
    peak_sig = ValErr.fromFit(popt_nacl_k_peak, pcov_nacl_k_peak, 2)

    angle_err = peak_sig.val / np.sqrt(len(nacl_ang[nacl_peak_low:nacl_peak_up]))
    #plt.axvline(x=peak_mu.val, linewidth=2, linestyle='--', color='green')
    #plt.axvline(x=peak_mu.val + angle_err, linewidth=1, linestyle='--', color='#56b300')
    #plt.axvline(x=peak_mu.val - angle_err, linewidth=1, linestyle='--', color='#56b300')

    return ValErr(peak_mu.val, angle_err)

nacl_kbeta1_angle = find_nacl_peak(15, 21)
nacl_kalpha1_angle = find_nacl_peak(19, 25)

plt.savefig(f'out/nacl_k_1ord_fit.png', format='png', bbox_inches='tight')

# 2d sin( ) = n   <=> d = n   / 2 sin( )

def calc_d(lam, ang):
    lam_val = lam.val
    lam_err = lam.err

    ang_val = np.deg2rad(ang.val)
    ang_err = np.deg2rad(ang.err)

    sin_ang_val = np.sin(ang_val)
    cos_ang_val = np.cos(ang_val)

    d_val = (1/2) * (lam_val / sin_ang_val)
    d_err = (1/2) * np.sqrt((lam_err / sin_ang_val) ** 2 + ((lam_val * cos_ang_val * ang_err) / sin_ang_val ** 2) ** 2)
    return ValErr(d_val, d_err)

nacl_d_kbeta = calc_d(kbeta_mean, nacl_kbeta1_angle)
nacl_d_kalpha = calc_d(kalpha_mean, nacl_kalpha1_angle)
```

```
nacl_d_mean = (nacl_d_kbeta + nacl_d_kalpha) / 2

nacl_M_Mol = 58.44 # g
nacl_rho = 2.164 * 100**(3) # g / m^3

# N_A = 1/2 * M_{Mol} /   d^3

N_A_val = (1/2) * (nacl_M_Mol / nacl_rho) * (1 / nacl_d_mean.val**3)
N_A_err = (1/2) * (nacl_M_Mol / nacl_rho) * (3 * nacl_d_mean.err / nacl_d_mean.
 ↪val**4)
N_A = ValErr(N_A_val, N_A_err)

# NaCL a, https://de.wikipedia.org/wiki/Natriumchlorid-Struktur
nacl_a_lit = 0.564 * 10**(-9)

print_all(
    kalpha_mean.strfmtf2(6, -12, ' ,K_ '),
    kbeta_mean.strfmtf2(6, -12, ' ,K_ '), '',
    nacl_kalpha1_angle.strfmtf2(6, 0, ' ,K_ '),
    nacl_kbeta1_angle.strfmtf2(6, 0, ' ,K_ '), '',
    nacl_d_kalpha.strfmtf2(6, -12, 'd,K_ '),
    nacl_d_kbeta.strfmtf2(6, -12, 'd,K_ '),
    nacl_d_mean.strfmtf2(6, -12, 'd'),
    (nacl_d_mean * 2).strfmtf2(6, -12, 'a'),
    (nacl_d_mean * 2).sigmadiff_fmt(ValErr(nacl_a_lit, 0)),
    N_A.strfmtf2(6, 23, 'N_A'),
    N_A.sigmadiff_fmt(ValErr(Consts.N_A, 0)))
```

```
 ,K_  = (71.171382 \pm 0.110251) \cdot 10^{-12}
 ,K_  = (63.274149 \pm 0.135983) \cdot 10^{-12}

 ,K_  = 7.291215 \pm 0.044213
 ,K_  = 6.459243 \pm 0.050100

d,K_  = (280.395432 \pm 1.746003) \cdot 10^{-12}
d,K_  = (281.227410 \pm 2.254563) \cdot 10^{-12}
d = (280.811421 \pm 1.425796) \cdot 10^{-12}
a = (561.622843 \pm 2.851593) \cdot 10^{-12}
0.84
N_A = (6.097877 \pm 0.092884) \cdot 10^{23}
0.82
```



Röntgenspektrum NaCl, K_β, K_α 1. Ordnung
Zählrate [1/s]
3000
2500
2000
1500
1000
500
6.0
6.1
6.2
6.3
6.4
6.5
6.6
6.7
6.8
6.9
7.0
7.1
7.2
7.3
7.4
7.5
7.6
7.7
Winkel [$^\circ$]

[ ]: