reflek.py
2021-07-30T11:15+02:00

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from scipy.optimize import curve_fit, root
from scipy.signal import find_peaks
from scipy.stats import sem
from uncertainties import ufloat

#Daten einlesen, I_max aus Detectorscan
data_dif = np.genfromtxt('data/Omega2ThetaScan2_difuse.UXD', unpack = True)
data_ref = np.genfromtxt('data/Omega2ThetaScan2.UXD', unpack = True)
I_max = 1637959.6850157096 *5 #mal fünf wegen unterschiedlicher messzeiten
                              #beim Detectorscan 1s hier 5s
a_g = 0.56 #Geometriewinkel
d_0 = 0.24 #Strahlbreite in mm
D = 20 #Laenge Probe in mm
a_g_berechnet = np.rad2deg(np.arcsin(d_0/D))
print('Geometriewinkel gemessen: ', a_g, ', berechnet: ', a_g_berechnet, ', Differenz: ', abs(a_g-a_g_berechnet), ', Abweichung: ', abs(a_g-a_g_berechnet)/a_g_berechnet)
lambda_0 = 1.54*10**(-10) # wellenlänge in m

#x_dif und x_ref sind im Grunde das selbe also wird im weiteren verlauf nur eins genutzt
x_dif = data_dif[0,:]
x_ref = data_ref[0,:]
x = x_ref #schneiden ersten und letzten wert ab um teilen durch null
          #zu verhindern und weil die nicht wichtig für auswertung sind

I_dif = data_dif[1,:]#hier gilt das selbe wie für x
I_ref = data_ref[1,:]

R_dif = I_dif / I_max #berechnen den der reflektivität
R_ref = I_ref / I_max

R_abs = R_ref - R_dif #absolute refflektivität


```

```
#Geometriefaktor berechnen
G = np.ones(len(R_abs))
G[x<a_g] = D/d_0 * np.sin(np.deg2rad(x[x<a_g])) #Geometriefaktor
print('Geometriefaktor: ', np.sum(G)/len(G))
R_G = R_abs*G #Korrektur geometriewinkel

####Peaks finden#############################################################################

peaks_bereich = (x>=0.3) & (x<=1.11)
#Durch log der peaks kann eine linie gezogen werden:
def f(x,b,c):
    return b*x+c

# Curve Fit für find_peaks
params, pcov = curve_fit(f,x[peaks_bereich],np.log(R_G[peaks_bereich]))
R_fit = np.exp(f(x[peaks_bereich],*params))

# Minima der Kissig-Oszillation finden
idx_peaks, peak_props = find_peaks(-(R_G[peaks_bereich]-R_fit),
    distance=7)
idx_peaks += np.where(peaks_bereich)[0][0]

#Schichtdicke
delta_x = np.diff(np.deg2rad(x[idx_peaks]))
delta_x_mean = ufloat(np.mean(delta_x),sem(delta_x))

d = lambda_0 / (2*delta_x_mean)
print('Schichtdicke berechnet: ' , d )

n1 = 1.
z1 = 0.
k = 2*np.pi/lambda_0

#Koeffizienten
delta2 = 0.5*10**(-6)
delta3 = 6.2*10**(-6)
sigma1 = 8.5*10**(-10) # m
sigma2 = 5.5*10**(-10) # m
z2 = 8.63*10**(-8) # m #(Schichtdicke)

def parrat_rau(a_i,delta2,delta3,sigma1,sigma2,z2):
    n2 = 1. - delta2
    n3 = 1. - delta3
```

```
78
79      a_i = np.deg2rad(a_i)
80
81      kz1 = k * np.sqrt(np.abs(n1**2 - np.cos(a_i)**2))
82      kz2 = k * np.sqrt(np.abs(n2**2 - np.cos(a_i)**2))
83      kz3 = k * np.sqrt(np.abs(n3**2 - np.cos(a_i)**2))
84
85      r12 = (kz1 - kz2) / (kz1 + kz2) * np.exp(-2 * kz1 * kz2 * sigma1**2)
86      r23 = (kz2 - kz3) / (kz2 + kz3) * np.exp(-2 * kz2 * kz3 * sigma2**2)
87
88      x2 = np.exp(-2j * kz2 * z2) * r23
89      x1 = (r12 + x2) / (1 + r12 * x2)
90      R_parr = np.abs(x1)**2
91
92      return R_parr
93
94  params = [delta2,delta3,sigma1,sigma2,z2]
95  # params, cov = curve_fit(parrat_rau, x[1:301], np.log(R_G[1:301])) #
    ↪  Curve_fit möchte nicht
96
97  R_parr = (parrat_rau(x[41:301], *params))
98
99
100 # Kritischer Winkel
101 x_c2 = np.rad2deg(np.sqrt(2*delta2))
102 x_c3 = np.rad2deg(np.sqrt(2*delta3))
103
104 print('kritischer Winkel poly: ', x_c2, 'kritischer Winkel silli: ',
    ↪  x_c3)
105
106 # Ideale Fresnelreflektivität
107 a_c_Si = 0.223
108 R_ideal = (a_c_Si / (2 * x[41:301]))**4
109
110
111 print('Paramter für fit: ', *params)
112 #plotten
113 plt.figure()
114 plt.plot(x[1:301], R_dif[1:301]/10, label='Diffuser Scan / 10',
    ↪  linewidth=0.5)
115 plt.plot(x[1:301], R_ref[1:301] /10, label='Reflektivitätsscan / 10',
    ↪  linewidth=0.5)
116 plt.plot(x[1:301], R_abs[1:301], label='Reflektivität', linewidth=0.5)
```

```
117 plt.plot(x[41:301], R_ideal, label='Ideal Reflektivität nach Fresnel',
    ↪ linewidth=0.5)
118 plt.plot(x[41:301], R_parr, '-', label='Theoriekurve', linewidth=0.5)
119 plt.plot(x[1:301], R_G[1:301], '-', label=r'Reflektivität$\cdot G$',
    ↪ linewidth=0.5)
120 plt.plot(x[idx_peaks], R_G[idx_peaks], 'rx',
    ↪ label='Oszillationsminima',alpha=0.8, ms=2.0)
121 plt.grid()
122 plt.yscale('log')
123 plt.legend(loc='upper right',prop={'size': 8})
124 plt.xlabel(r'$\alpha_\text{i} \,/\, \si{\degree}$')
125 plt.ylabel(r'$R$')
126 plt.tight_layout(pad=0.15, h_pad=1.08, w_pad=1.08)
127 plt.savefig('build/reflek.pdf')
```