O11 - 624847 (Polarisation durch Reflexion)

Date: 2023-04-28
Tags: O11
Created by: Philipp Schubert

Versuch O11 - Polarisation durch Reflexion an Glas

Zuerst tragen Sie ganz oben in der "Title" Zeile hinter "O11" ihre Matrikelnummern ein (Bsp: "O11 - 333444, 456678").

Ab hier befüllen Sie bitte alle blau markierten Textfelder, die Tabellen und ersetzen Sie die Platzhalter Grafiken entsprechend.

Abgebende Person ist Philipp Schubert, 624847 - die Daten wurden gemeinsam mit Eduard Hebecker erfasst.

Name	Matrikelnummer
Philipp Schubert	624847

1. Vor Ort: Rohdaten aufnehmen

Tragen Sie hier den Referenzwert ein

Der Versuchsaufbau wurde gemäß der Versuchsanleitung sowohl für die Messung bei paralleler Polarisation als auch für die Messung bei senkrechter Polarisation kalibriert. Daraus ergeben sich Referenzwerte für die Sensorausgangsspannung:

- U_{ep} beschreibt den Referenzwert für die (Sensorausgangs-)Spannung bei paralleler Polarisation
- U_{es} beschreibt den Referenzwert für die (Sensorausgangs-)Spannung bei senkrechter Polarisation

U_{ep} (V),	4,4600
U_{es} (V)	4,5000

Tab. 1: Tragen Sie hier die Reflektierten Intensitäten ein (winkel- und polarisationsabhängig):

Auf Grund technischer Einschränkungen des Aufbaus konnten keine Messwerte für 5 Grad erhoben werden.

Parallele Polarisation				**Senkrechte Polarisation**			
θ_e (°)	U_r (V)	θ_e (°)	U_r (V)	θ_e (°)	U_r (V)	θ_e (°)	U_r (V)
		47.5	0.0250			47.5	0.4310

7.5	0.1730	50	0.0140	7.5	0.192	50	0.5110
10	0.1650	52.5	0.0070	10	0.1850	52.5	0.5640
12.5	0.1790	55	0.0020	12.5	0.1880	55	0.6660
15	0.1680	57.5	0.0010	15	0.1910	57.5	0.7770
17.5	0.1740	60	0.0070	17.5	0.1970	60	0.9120
20	0.1490	62.5	0.0230	20	0.2100	62.5	0.9420
22.5	0.1410	65	0.0510	22.5	0.2140	65	1.0440
25	0.1310	67.5	0.0970	25	0.2280	67.5	1.1500
27.5	0.1220	70	0.1780	27.5	0.2370	70	1.4040
30	0.1130	72.5	0.2840	30	0.2520	72.5	1.5730
32.5	0.1040	75	0.4440	32.5	0.2750	75	1.9650
35	0.0990	77.5	0.6170	35	0.2870	77.5	2.4430
37.5	0.0610	80	0.9020	37.5	0.3100	80	2.7630
40	0.0550	82.5	1.3650	40	0.3370	82.5	3.0330
42.5	0.0440	85	1.9950	42.5	0.3670	85	3.2220
45	0.0340			45	0.4030		

(Beispiel Jupyter Notebook, das diese Tabelle direkt ausliest [bitte Experiment ID anpassen!])

2. Auswertung der Daten

Kommentare:

- *Alle Abbildungen haben aussagekräfige Bildunterschriften, Messunsicherheiten werden mit eingezeichnet, Datenpunkte werden nicht mit Linien verbunden.*
  Aussagekräftig heißt, dass alles, was man sehen kann kurz erklärt wird (z.B. "durchgezogene Linie zeigt die Fitfunktion f") und Überraschendes erwähnt wird (z.B. krasse Ausreißer).
- *Alle finalen Messergebnisse (hier nur der Brechungsindex) werden mit Messunsicherheiten angegeben.*

2.1. Messunsicherheiten

Geben Sie hier die von Ihnen genutzten Formeln zur Berechnung der Messunsicherheiten zu den von Ihnen **direkt gemessenen Größen** R und $alpha$ an. Das sind die Messunsicherheiten, die Sie auch für die graphischen Darstellungen und Kurvenanpassungen benötigen. Denken Sie daran, dass es systematische und statistische Messunsicherheiten gibt. (Anmerkung. R wurde in diesem Versuch *nicht* direkt gemessen)

In diesem Experiment wurde die Spannung U direkt gemessen und der Winkel α eingestellt. Wir können also eine Messunsicheriet für die Spannung und den eingestellten Winkel ermitteln. Da jeder für die jeweilige Polarisierungsrichtung ein Winkel nur einmal eingestellt wurde und eine Spannung gemessen wurde, können

wir keinen statistische Messunsicherheit e_z bestimmen. *(Vgl.: Skript:* Physikalisches Grundpraktikum Einführung in die Messung, Auswer- tung und Darstellung experimenteller Ergebnisse in der Physik; Müller, 2007)
Die Messunsicherheit u für Spannung u_U und Winkel u_α ergibt sich also rein aus der jeweiligen systematischen Messunsicherheit e_s.

$$u = e_s$$

Folgende Quellen für systematische Unsicherheiten der Winkeleinstellung wurden betrachtet:

- Skala
- Ableseungenauigkeit

Da keine Informationen über die Skala vorliegt, wird die Messungenauigkeit für Winkeleinstellung pro Winkel mit einer halben Skaleneinheit angenommen:

$$u_\alpha = 0,5$$

Folgende Quellen für systematische Unsicherheiten der Spannungsmessung wurden betrachtet:

- Messgerätefehler
- Kalibrierung des Messaufbaus vor Versuchsbeginn
- Einstellung der Blende vor Aufnahme eines Messwerts nach geändertem Winkel
- Abdunklung des Raumes

Für die Messunsicherheit in der Spannungsmessung wird nach Rücksprache mit dem Praktikumsleiter folgendes angenommen:

$$u_U = (0,03 \cdot U) \pm 0,005$$

Um den Fehler in R zu bestimmen, wird die Gaußsche Fehlerfortpflanzung benutzt:

$$u_R = \frac{1}{\sqrt{2U_{max}\sqrt{\frac{U}{U_{max}}}}} \cdot u_U$$

2.1. Grafik der Experimentelle Daten

Fügen Sie eine Grafiken $\sqrt{R}$ gegen α mit beiden Polarisationen (s- und p-pol) in das folgenden freie Feld ein:

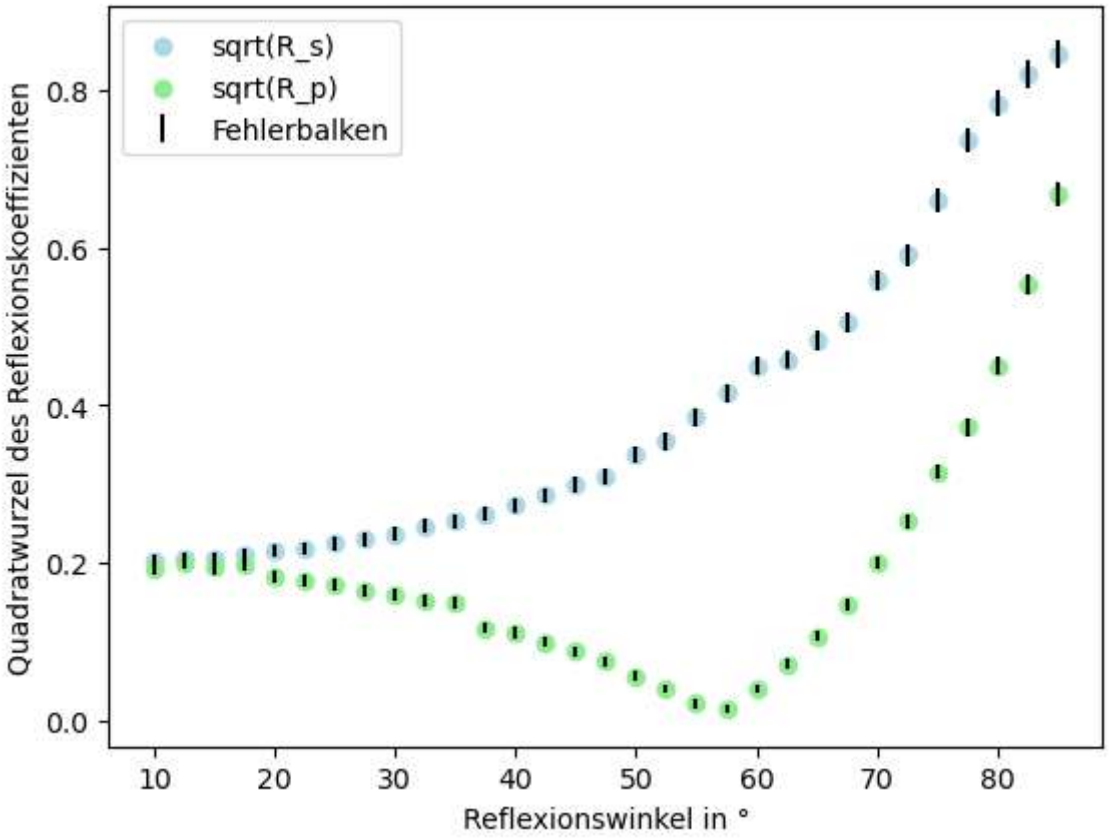


Abb.1: Abhängigkeit der Wurzel des Reflexionsvermögen vom Relexionswinkel für parallele und senkrechte Polarisierung, inklusive Fehlerbalken

2.2. Bestimmen des Brechungsindex des Glases:

Erläutern Sie in 1-2 Sätzen Ihre Vorgehensweise zum Bestimmen des Brechungsindex des Glas-Halbzylinders. Gehen Sie auch auf die Bestimmung von Messunsicherheiten (statistisch / systematisch) ein:

Die Berechnung des Brechungsindex inklusive der Messunsicherheit wird mittels der Fit-Funktion *curve_fit* aus der *scipy* Bibliothek in Python bestimmt. Hierfür wurde zunächst die tatsächliche, theoretische Kurve aus der Gleichung in der Versuchsanleitung definiert und mit den erhobenen Daten gefittet.
Die Definition der Kurven sind im angehängten Python Skript nachzulesen. (ln 13ff.)
Wie oben beschrieben spielt nur die systematische Messunsicherheit eine Rolle.

Brechungsindex n_2 (inkl. Messunsicherheit):	1.512 ± 0.019

2.3. Theorie versus Experiment:

Fügen Sie erneut die Grafiken $\sqrt{R}$ gegen α wie unter 2.1. ein aber diesmal zusammen mit den Funktionen, die die Daten laut Model beschreiben sollen. Das Model füttern Sie mit dem gefundenen Brechungsindex aus 2.2:

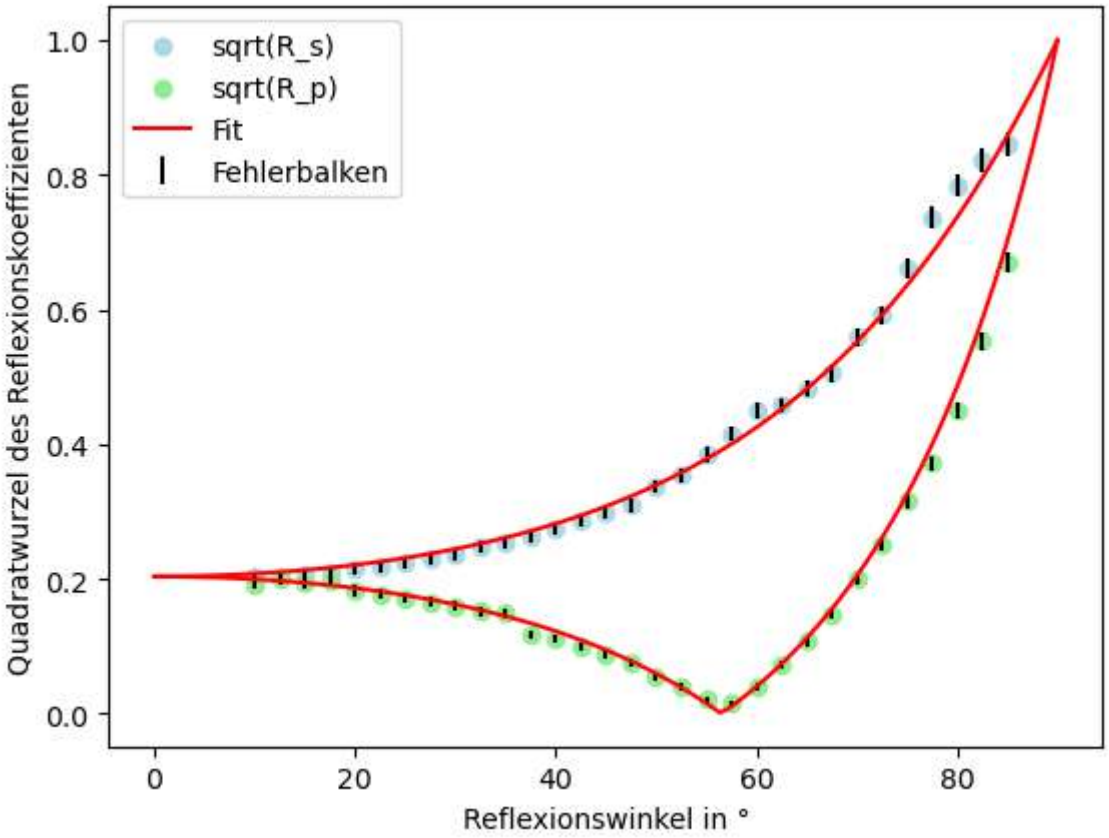


Abb.2: Abhängigkeit der Wurzel des Reflexionsvermögen vom Relexionswinkels, ergänzt um die jeweils theoretischen Funktionseschreibungen, für parallele und senkrechte Polarisierung, inklusive Fehlerbalken

3. Diskussion

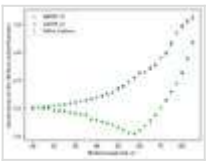
0-4 Sätze: Welche Überraschungen gab es? Was ist jeweils die wahrscheinlichste Erklärung?

Der Versuch ist sehr gut verlaufen. Wir konnten bereits zu Beginn des Versuchs feststellen, dass die Messung für 5° durch die Einschränkung des Messaufbaus nicht realisierbar sind. Zunächst waren wir für die Messung kleiner Winkel verwundert, dass die gemessene Spannung, entgegen der Annahme, sinkt. Diese Spannungsreduktion im Bereich Hundertstel Volt stellte sich in der Auswertung jedoch als nicht problematisch dar.

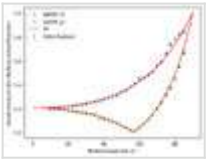
Der geringste gemessene Spannungswert für parallele Polarisation wurde bei 57,5° gemessen und passt damit zur theoretischen Annahme des Brewsterwinkels aus der Praktikumsanleitung.

Attached files

unknown.png
sha256: 3913949205de1b45863c2d4e0fcb5e3850d078bb1a0c35bf55868ae8eaf2983f



unknown.png
sha256: 0c9383be01b54163a7caa629e251f7f57ef649c18d6084708de7165dfae6ae5a



Unique eLabID: 20230428-fd6c6f422f5c56f4bfd04e20d3ab3603c3c0f539
Link: https://elabftw.physik.hu-berlin.de/experiments.php?mode=view&id=700

Platz 3

Schritte	Parallel (90)	Senkrecht (0/180)
5	0,0515V	
7,5	0,173V	0,192V
10	0,165V	0,195V
12,5	0,179V	0,188V
15	0,165V	0,151V
17,5	0,174V	0,197V
20	0,149V	0,210V
22,5	0,141V	0,214V
25	0,131V	0,228V
27,5	0,122V	0,237V
30	0,113V	0,252V
32,5	0,104V	0,275V
35	0,090V	0,287V
37,5	0,061V	0,310V
40	0,055V	0,337V
42,5	0,044V	0,367V
45	0,034V	0,403V
47,5	0,025V	0,437V
50	0,014V	0,511V
52,5	0,007V	0,564V
55	0,002V	0,666V
57,5	0,001V	0,777V
60	0,007V	0,912V
62,5	0,023V	0,942V
65	0,051V	1,044V
67,5	0,097V	1,150V
70	0,178V	1,404V
72,5	0,294V	1,573V
75	0,444V	1,965V
77,5	0,617V	2,443V
80	0,902V	2,763V
82,5	1,365V	3,033V
85	1,995V	3,222V
0°	4,46V	4,50V

24.4.23

Erkan

In [2]:
```
pip install --user elabapy
```

```
Requirement already satisfied: elabapy in /Users/philippschubert/.local/l
ib/python3.9/site-packages (1.0.0)
Requirement already satisfied: requests in /Users/philippschubert/opt/ana
conda3/lib/python3.9/site-packages (from elabapy) (2.28.1)
Requirement already satisfied: charset-normalizer<3,>=2 in /Users/philipp
schubert/opt/anaconda3/lib/python3.9/site-packages (from requests->elabap
y) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/philippsch
ubert/opt/anaconda3/lib/python3.9/site-packages (from requests->elabapy)
(1.26.11)
Requirement already satisfied: idna<4,>=2.5 in /Users/philippschubert/opt
/anaconda3/lib/python3.9/site-packages (from requests->elabapy) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in /Users/philippschube
rt/opt/anaconda3/lib/python3.9/site-packages (from requests->elabapy) (20
22.9.24)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
import pandas as pd
import numpy as np
import elabapy
import json
from requests.exceptions import HTTPError
import matplotlib.pyplot as plt
```

In [4]:
```
my_api_key = 'cfba8c63ed36752f733589740e339b0d8bac2d24a6355ead5f4adebb726
manager = elabapy.Manager(endpoint="https://elabftw.physik.hu-berlin.de/a
```

In [5]:
```
ExperimentID = 700
try:
    experiment = manager.get_experiment(ExperimentID)
# if something goes wrong, the corresponding HTTPError will be raised
except HTTPError as e:
    print(e)
```

In [6]:
```
type(experiment)
```

Out[6]:
```
dict
```

In [10]:
```
htmlCode = experiment["body"]
# Die folgende zeile ausführen, um den HTML Code anzeigen zu lassen:
# print(htmlCode)
```

In [11]:
```
df = pd.read_html(htmlCode, match='Parallele Polarisation')
df[0]
```

Out[11]:

	0	1	2	3	4	5	6
0	Parallele Polarisation	Parallele Polarisation	Parallele Polarisation	Parallele Polarisation	Senkrechte Polarisation	Senkrechte Polarisation	Senkrechte Polarisation
1	αe (°)	Ur (V)	αe (°)	Ur (V)	αe (°)	Ur (V)	αe (°)
2	NaN	NaN	47.5	0.0250	NaN	NaN	47.5
3	7.5	0.1730	50	0.0140	7.5	0.192	50
4	10	0.1650	52.5	0.0070	10	0.1850	52.5
5	12.5	0.1790	55	0.0020	12.5	0.1880	55
6	15	0.1680	57.5	0.0010	15	0.1910	57.5
7	17.5	0.1740	60	0.0070	17.5	0.1970	60
8	20	0.1490	62.5	0.0230	20	0.2100	62.5
9	22.5	0.1410	65	0.0510	22.5	0.2140	65
10	25	0.1310	67.5	0.0970	25	0.2280	67.5
11	27.5	0.1220	70	0.1780	27.5	0.2370	70
12	30	0.1130	72.5	0.2840	30	0.2520	72.5
13	32.5	0.1040	75	0.4440	32.5	0.2750	75
14	35	0.0990	77.5	0.6170	35	0.2870	77.5
15	37.5	0.0610	80	0.9020	37.5	0.3100	80
16	40	0.0550	82.5	1.3650	40	0.3370	82.5
17	42.5	0.0440	85	1.9950	42.5	0.3670	85
18	45	0.0340	NaN	NaN	45	0.4030	NaN

In [12]:

```
dataTable = df[0].tail(-2).astype('float') # Wir skippen die ersten 2 Zei
data = dataTable.to_numpy(dtype='float', na_value=np.nan) # Umwandeln in
Winkel = data[:, [0, 2]].T.flatten() # extrahieren der 1. und 3. Spalte
SpannungP = data[:, [1, 3]].T.flatten() # extrahieren der 2. und 4.
SpannungS = data[:, [5, 7]].T.flatten() # extrahieren der 2. und 4. Spalt
ind = ~np.isnan(Winkel)
Winkel = Winkel[ind]
SpannungS = SpannungS[ind]
SpannungP = SpannungP[ind]
SpannungS = np.delete(SpannungS, 0)
SpannungP = np.delete(SpannungP, 0)
Winkel = np.delete(Winkel, 0)
MaxSpannungS = 4.5
MaxSpannungP = 4.46
```

In [13]:
```
#definition der theoretischen Rs/Rp Funktionen
def rsKurve(x, a):
    y = (np.sin(np.radians(x - np.degrees(np.arcsin(np.sin(np.radians(x))
    return y
def rpKurve(x, a):
    y = np.sqrt((np.tan(np.radians(x - np.degrees(np.arcsin(np.sin(np.rad
    return y
```

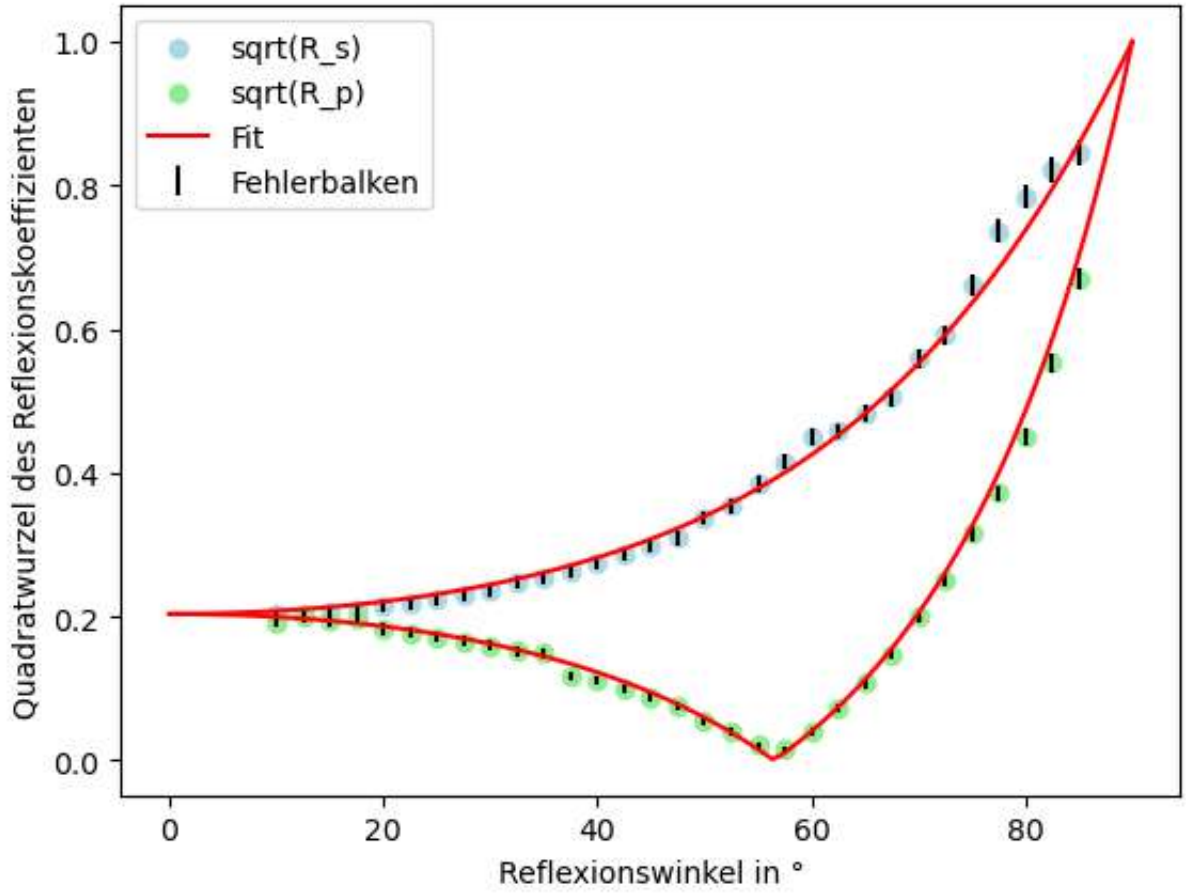
In [14]:
```
#definition der Rs/Rp Fit-Funktionen
def rs_fit(x, y):
    popt, pcov = curve_fit(rsKurve, x, y)
    return popt
def rp_fit(x, y):
    popt, pcov = curve_fit(rpKurve, x, y)
    return popt
```

In [15]:
```
#Berechnung sqrt(Rs/Rp) und Messunsicherheiten
errorS = []
errorP = []
sqrtRS = []
sqrtRP = []
xdata = np.linspace(0.0000001, 90, 100)
for i in range(len(SpannungS)):
    sqrtRS.append(np.sqrt(SpannungS[i]/MaxSpannungS))
    sqrtRP.append(np.sqrt(SpannungP[i]/MaxSpannungP))
for i in range(len(sqrtRS)):
    errorS.append(1/((2*MaxSpannungS)*np.sqrt(SpannungS[i]/MaxSpannungS))
    errorP.append(1/((2*MaxSpannungP)*np.sqrt(SpannungP[i]/MaxSpannungP))
```

In [17]:
```
from scipy.optimize import curve_fit

#Berechnung des Brechungsindex
brechungsindex = (rs_fit(Winkel, sqrtRS) + rp_fit(Winkel, sqrtRP))/2
brechungsindexunsicherheit = np.sqrt(np.diag(curve_fit(rsKurve, Winkel, s
np.sqrt(np.diag(curve_fit(rpKurve, Winkel, sqrtRP)[1]))
print(str(brechungsindex) + "+-" + str(brechungsindexunsicherheit))
```

```
[1.5119994]+-[0.0193347]
```

In [27]:
```
#Plot der Funktionen/Messwerte
#je nach Anforderungen können die Fits durch Einsatz von "#" ein oder aus

import sympy as s

plt.scatter(Winkel, sqrtRS, label="sqrt(R_s)", color="lightblue")
plt.scatter(Winkel, sqrtRP, label="sqrt(R_p)", color="lightgreen")
plt.errorbar(Winkel, sqrtRS, yerr=errorS, fmt="none", ecolor="black")
plt.errorbar(Winkel, sqrtRP, label="Fehlerbalken", yerr=errorP, fmt="none
plt.xlabel("Reflexionswinkel in °")
plt.ylabel("Quadratwurzel des Reflexionskoeffizienten")
plt.plot(xdata, rsKurve(xdata, brechungsindex), label="Fit", color="red")
plt.plot(xdata, rpKurve(xdata, brechungsindex), color="red")
plt.legend()
plt.show()
```


sqrt(R_s)
sqrt(R_p)
Fit
Fehlerbalken
Quadratwurzel des Reflexionskoeffizienten
Reflexionswinkel in °
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80

In [ ]: