Polarisation durch Reflexion

Santiago R. [redacted][1]

[1]*Institut für Physik, Humboldt Universität zu Berlin, Deutschland*
Versuchsleiter: Christopher Böttner und Vinzenz Zimmermann, Raum 213
(Abgabe: 22. März 2021; Versuchsdatum: 18.3.2021)

Bei der Reflexion von Licht an einer Ebene zwischen zwei Medien unterschiedlicher Brechzahlen n_1 und n_2 besteht ein Zusammenhang zwischen der einfallswinkelabhängigen Reflektivität $R(\alpha_k)$ und der Polarisation des einfallenden Lichts. Bei zur Ebene senkrechter Polarisation nimmt die Reflektivität stetig zu, während bei paralleler Polarisation diese zuerst bis zu einem bestimmten Winkel α_B- den Brewster Winkel- gegen $R(\alpha_B) = 0$ verläuft, bevor die Reflektivität wieder zunimmt. Im Rahmen dieses Experimentes wurde dieses Verhalten mithilfe der Reflexion eines Laserstrahls an einer halbierten Linse untersucht und für das Material derselben ein Brewster-Winkel von $\alpha_B = (55.565 \pm 0.003)°$ bestimmt sowie eine Brechzahl von $n_2 = (1.459 \pm 0.001)$

I Einleitung und Versuchsaufbau

Wenn Licht auf eine Grenzfläche zwischen zwei Medien unterschiedlicher Brechzahlen n_1 und n_2 trifft, so verläuft ein Teil des Lichtstrahls ins zweite Medium gebrochen weiter, während ein anderer Teil zurück ins erste Medium reflektiert wird. Für die Richtung der beiden Teilstrahlen liefern die Reflexions- (1) und Brechungs-Gesetze (2)

$$\alpha_e = \alpha_r \tag{1}$$

$$n_1 sin(\alpha_e) = n_2 sin(\alpha_g) \tag{2}$$

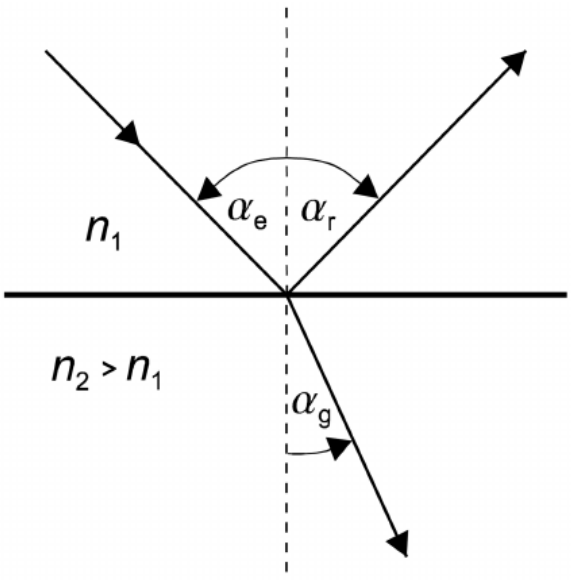


Abbildung 1: Brechung und Reflexion beim Übergang $n_1 \to n_2$

die Beziehung zwischen den Einfallswinkel α_e des Lichtstrahls und den jeweiligen Reflektions- α_r und Brechungswinkeln α_g. Welcher Anteil des Lichts gebrochen und welcher reflektiert hängt direkt von der Polarisationsrichtung des eintreffenden Lichtstrahls ab. Die Intensitätsverteilungen des reflektierten Lichtstrahls lassen sich hierbei aus den Fresnelschen Formeln für zur Einfallsebene senkrecht (3) und parallel (4) polarisiertes Licht mit dem einfallendem elektrischen Feldstärkevektor E_e herleiten als

$$E_{rs} = -E_{es} \frac{sin(\alpha_e - \alpha_g)}{sin(\alpha_e + \alpha_g)} \tag{3}$$

$$E_{rp} = E_{ep} \frac{tan(\alpha_e - \alpha_g)}{tan(\alpha_e + \alpha_g)} \tag{4}$$

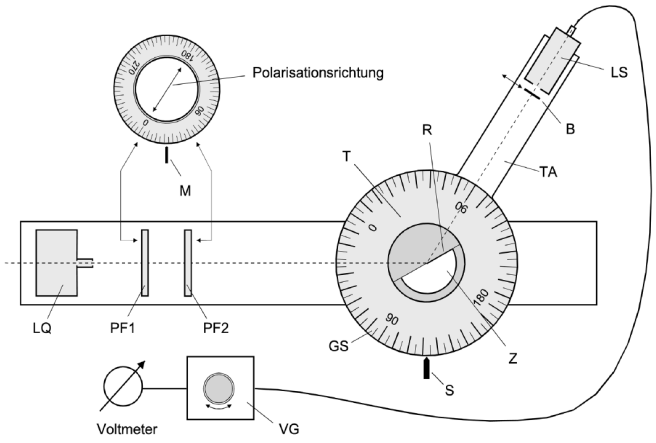


Abbildung 2: Versuchsaufbau

Die Versuchsanordnung besteht aus einen Laser der als Lichtquelle LQ dient und aus dem ein paralleles Lichtbündel auf eine frei drehbare, halbierte Zylinderlinse Z gerichtet ist. Bei dessen Weg zur Zylinderlinse verläuft der Laserstrahl jedoch zuerst durch zwei Polarisationsfilter PF1 und PF2, wobei der erstere zur Intensitätsregelung und der zweite zur Polarisierung des Laserstrahls benutzt werden konnte. Das an der als Reflektor R dienende Fläche der Linse reflektierte Lichtbündel wird dann von einem Lichtsensor LS an einem frei schwenkbaren Tragarm TA eingefangen, dessen Ausgangsspannung mithilfe des Versorgungsgerätes VG für Hintergrunssignale kompensieren und auf 0V justiert werden kann. Der Einfallswinkel α_e des Laserstrahls am Reflektor R kann im Anschluss am Tisch T abgelesen werden.

Der zur Spannungsmessung am LS benutzte Multimeter ist mit einer Unsicherheit $\frac{u_v}{U_m} = 0.003$ von 0.3% des Messwertes U_m behaftet. Weiterhin beträgt die Ableseunsicherheit der Gradskala $u_\alpha = 0.1°$

II Theoretische Vorhersagen

Aus den Gleichungen (3) und (4) können aufgrund der Beziehung $I \backsim E^2$ weiterhin Formeln für das Reflexionsvermögen $R = \frac{I_r}{I_e}$ hergeleitet werden als

$$R_s = \frac{I_{rs}}{I_{es}} = \frac{E_{rs}^2}{E_{es}^2} = \frac{sin^2(\alpha_e - \alpha_g)}{sin^2(\alpha_e + \alpha_g)} \tag{5}$$

$$R_p = \frac{I_{rp}}{I_{ep}} = \frac{E_{rp}^2}{E_{ep}^2} = \frac{tan^2(\alpha_e - \alpha_g)}{tan^2(\alpha_e + \alpha_g)} \tag{6}$$

Weiterhin gilt aus den Brechungsgesetz (2) für den Winkel α_g

$$\alpha_g = arcsin\left(\frac{n_1}{n_2} sin(\alpha_e)\right) \tag{7}$$

Für $n_1 \approx 1$ und $n_2 \approx 1.5$ beim Übergang Luft-Glas ergeben sich dann für die Wurzeln $\sqrt{R}$ des Reflexionsvermögens die theoretischen Kurven

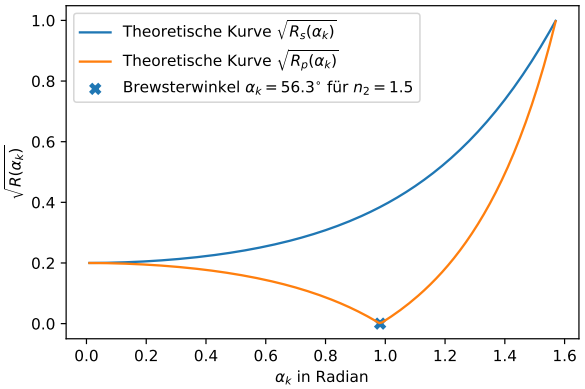


Abbildung 3: Theoretische Vorhersage bei $n_1 = 1$ und $n_2 = 1.5$

Im senkrecht polarisierten Fall steigt die Kurve stetig für grössere Einfallswinkeln α_e, während im parallel polarisierten Fall der Komponente des elektrischen Feldes E_p einer Lichtwelle die Wurzel des Reflexionsvermögens $\sqrt{R}$ zuerst an einem bestimmten Wert des Einfalsswinkels - den sogenannten Brewster-Winkel α_B - gegen 0 verläuft. Aus $\alpha_e = \alpha_B \Rightarrow \alpha_B + \alpha_g = 90°$ in diesem Spezialfall und Gleichung (2) folgt dann

$$tan(\alpha_B) = \frac{n_2}{n_1} \Leftrightarrow n_2 = n_1 tan(\alpha_B) \tag{8}$$

In der theoretischen Vorhersage mit $n_1 = 1$ und $n_2 = 1.5$ also

$$\alpha_B = 56.3° \tag{9}$$

III Fits und Experimentelle Bestimmung der Brechzahl n_2

Um die in der Versuchsanordnung gemessenen Spannungswerte U_r an die mit Gleichung (5) und (6) angegebenen Modelle zu fitten, wird zunächst die Beziehung $\frac{U_r}{U_e} = \frac{I_r}{I_e}$ ausgenutzt, wobei $U_e = (4.40 \pm 0.01)$V die Spannungsmessung am LS mit dem unabgelenkten Laserstrahl ist. Dann gilt

$$R_s(\alpha_e) = \frac{U_{rs}(\alpha_e)}{U_e} = \frac{sin^2(\alpha_e - \alpha_g)}{sin^2(\alpha_e + \alpha_g)} \tag{10}$$

$$R_p(\alpha_e) = \frac{U_{rp}(\alpha_e)}{U_e} = \frac{tan^2(\alpha_e - \alpha_g)}{tan^2(\alpha_e + \alpha_g)} \tag{11}$$

Weiterhin wird zur Konvergenz eines Fitting-Algorithmus nach dem Verfahren der kleinsten Quadrate $min(\chi^2)$ zuerst ein Startwert für den Fit-Parameter n_2 gewählt. n_1 ist als die Brechzahl der Luft bei $n_1 = 1.00028$ festgesetzt. Aus den Messwerten in der Umgebung des vorhin in (9) theoretisch angegebenen Brewster-Winkels α_g für zur Reflexionsebene parallel polarisiertes Licht

α_e	45°	50°	55°	60°	65°
U_r	0.03V	0.008V	0.002V	0.013V	0.061V

Tabelle 1: Messwerte im Bereich $[45°, 65°]$ für U_{rp}

folgt ebenfalls ein Brewster-Winkel von $\alpha_B \approx 55°$ und somit folgt aus Gleichung (8) ein Startwert von $n_2 \approx 1.43$ für den Fit-Algorithmus. Daraus folgt

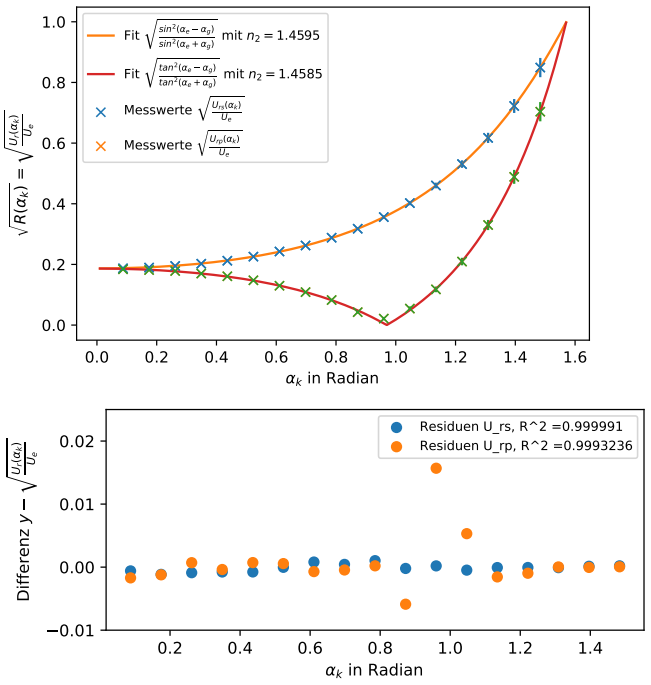


Abbildung 4: Fits und Residuen für die Messwerte bei senkrecht und parallel polarisierten Licht

mit den aus dem numerischen Verfahren bestimmten Fit-Parametern

$$n_{2s} = (1.4595 \pm 0.0003) \quad n_{2p} = (1.4585 \pm 0.001) \tag{12}$$

durch quadratisches gewichten der Unsicherheiten und Bildung des arithmetischen Mittelwertes beider Parameter von n_2 folgt dann für die Brechzahl n_2 der Halblinse Z

$$n_2 = (1.459 \pm 0.001) \tag{13}$$

IV Brewster-Winkel

Mit dem unter (13) bestimmten Parameter n_2 der Halblinse und Gleichung (8) lässt sich auch ein experimenteller Wert für den Brewster-Winkel herleiten durch umschreiben der Gleichung als

$$\alpha_B = arctan(\frac{n_2}{n_1}) \tag{14}$$

mit einer fortgepflanzten Unsicherheit von

$$u_{\alpha_B} = \frac{1}{1 + \frac{n_2}{n_1}^2} * u_{n_2} \tag{15}$$

daraus ergibt sich dann ein Wert für den Brewster-Winkel α_B von

$$\alpha_B = (55.565 \pm 0.003)° \tag{16}$$

V Diskussion

Die in Sektion II und III vorhergesagten Werte für die Brechzahl $n_2 = 1.5$ von Kronglas samt Brewster-Winkel von $\alpha_B = 56.3°$ stimmen mit dem dann aus experimentellen Messwerten bestimmten Parameter $n_2 = (1.459 \pm 0.001)$ und $\alpha_B = (55.565 \pm 0.003)°$ gut überein, da es sich bei der Halblinse Z um eine unbekannte Glas-Sorte anders als Kronglas handelt, aber immer noch im gleichen Wertebereich optischer Eigenschaften liegen sollte.

Die angegebenen Unsicherheiten sind u.U. evtl. unterschätzt, da die Unsicherheiten bei der Messung des Einfallswinkels α_e durch das Auswertungs-Algorithmus bedingt vernachlässigt werden mussten. Weiterhin sind bei den Residuen zwei deutliche Ausreiser in den Messungen für parallel polarisiertes Licht zu erkennen, weshalb ein Messfehler bei diesen Messwerten vermutet werden könnte. Dennoch erscheint an der grafischen Auswertung eine gute Übereinstimmung mit der Theorie, mit einer ebenfalls nachvollziehbaren Ähnlichkeit zwischen den Messwerten und der aus der Theorie sowie den Fits erstellten Kurven für das Reflexionsvermögen $\sqrt{R}$. Insofern unterstützen trotz der oben benannten Einschränkungen der Messergebnisse diese die aus den fresnelschen Formeln hergeleiteten Modelle zur Parametrisierung der Intensitätsverteilung für die Reflexion.

VI Literatur

[1] Fundamental Physical Constants; National Institute of Standards and Technology; 22. März 2021; https://physics.nist.gov/cgi-bin/cuu/Value?esme

[2] Dr. Uwe Müller: *Physikalisches Grundpraktikum: Einführung in die Messung, Auswertung und Darstellung experimenteller Ergebnisse in der Physik*, 2007

VII Anhänge

AuswertungO11

March 22, 2021

1 O11 Polarisation durch Reflexion an Glas

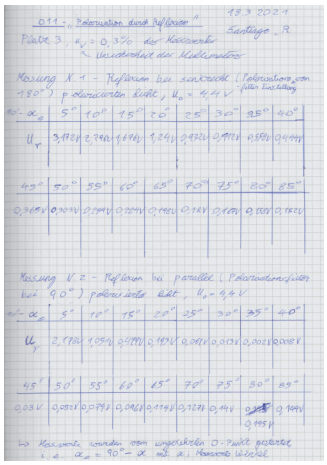
Santiago R. , 18.3.2021

[39]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

2 Messwerte

[78]:
```
from IPython.display import Image
Image(filename='Messwerte.png', width = 800, height = 300)
```

[78]:



[41]:
```
#Messwerte
alpha = np.array([5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85])*2*np.pi/360
U_s = np.array([0.154,0.158,0.167,0.18,0.198,0.224,0.259,0.303,0.365,0.444,0.
 ↪558,0.712,0.932,1.24, 1.676, 2.296,3.172])
U_s0 = 4.4
U_p = np.array([0.149,0.145,0.14,0.127,0.114,0.096,0.074,0.052,0.03,0.008,0.
 ↪002,0.013,0.061,0.193,0.479,1.051,2.178])
```

```
U_p0 = 4.4
u_Us = U_s*0.01
u_Up = U_p*0.01
```

3 Fits Intensitätsverteilung

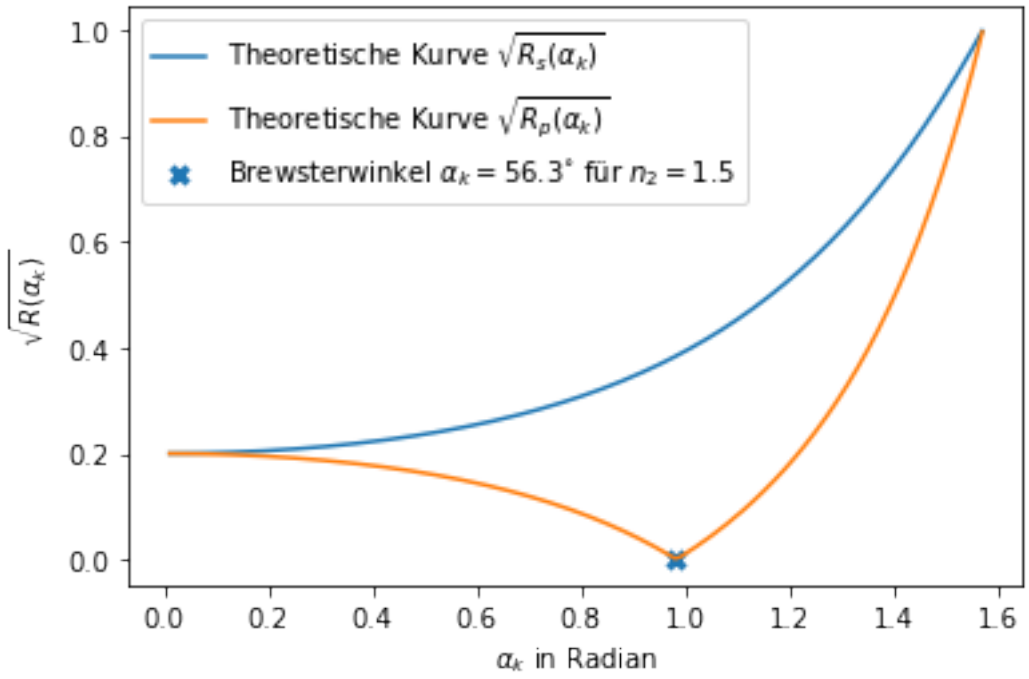
Für das Reflexionsvermögen des parallel R_p und senkrecht R_s polarisierten Laserstrahls stellen sich folgende Gleichungen zum Fitten zur Verfügung $R_s = \frac{sin^2(\alpha_e-\alpha_g)}{sin^2(\alpha_e+\alpha_g)}$ $R_p = \frac{tan^2(\alpha_e-\alpha_g)}{tan^2(\alpha_e+\alpha_g)}$ wobei α_g sich aus den Brechungsgesetzen ergibt mit $\alpha_g = arcsin(\frac{n_1}{n_2}sin(\alpha_e))$

[42]:
```
def R_s(alpha_e, n1, n2):
    alpha_g = np.arcsin(n1/n2*np.sin(alpha_e))
    R = np.sin(alpha_e-alpha_g)**2/(np.sin(alpha_e+alpha_g)**2)
    return np.sqrt(R)
def R_p(alpha_e, n1, n2):
    alpha_g = np.arcsin(n1/n2*np.sin(alpha_e))
    R = np.tan(alpha_e-alpha_g)**2/(np.tan(alpha_e+alpha_g)**2)
    return np.sqrt(R)
```

Zur theoretischen Vorhersage mit $n_1 = 1$ für die Brechung in Luft und $n_2 = 1.5$ für die Brechung an Kronglas folgt

[43]:
```
alpha_test = np.arange(0,5/10*np.pi,0.01)
plt.scatter(np.arctan(1.5/1.00028),0, label="Brewsterwinkel "r'$\alpha _k = 56.3␣
 ↪^{\circ}$'" für "r'$n_2 = 1.5$',marker='x',linewidths=3)
plt.plot(alpha_test,R_s(alpha_test, 1, 1.5), label="Theoretische Kurve␣
 ↪"r'$\sqrt{R_s(\alpha _k)}$')
plt.plot(alpha_test,R_p(alpha_test, 1, 1.5), label="Theoretische Kurve␣
 ↪"r'$\sqrt{R_p(\alpha _k)}$')
plt.xlabel(r'$\alpha _k$'" in Radian")
plt.ylabel(r'$\sqrt{R(\alpha _k)}$')
plt.legend(loc="upper left")
plt.savefig("TheoryCurves.pdf")
```

```
/home/santi/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
RuntimeWarning: invalid value encountered in true_divide
  This is separate from the ipykernel package so we can avoid doing imports
until
/home/santi/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7:
RuntimeWarning: invalid value encountered in true_divide
  import sys
```



An der Stelle, in dem die Kurve für parallel zur Brennebene polarisiertes Licht nach 0 verläuft, folgt für die Brechzahl n_2 des Glases; $tan(\alpha_e) = \frac{n_2}{n_1} \Leftrightarrow n_2 = n_1 tan(\alpha_e)$ Mit $n_1 = 1.00028$ als die Brechzahl von Luft. Aus den Messwert $R_p(55°) = 0.002 \approx 0$ folgt dann $n_2 = 1.00028 \cdot tan(55°) \approx 1.43$ Dieser Wert kann dann als Anfangswert für das Fit-Algorithmus verwendet werden

Zum Fitten an die Messwerte gilt zusätzlich $R(\alpha_e) = \frac{U_r(\alpha_e)}{U_e}$ s.d. gilt $$ $U_{rs}(\alpha_k) \frac{}{U_e = \frac{sin^2(\alpha_e - \alpha_g)}{sin^2(\alpha_e + \alpha_g)}} $$ $\frac{U_{rp}(\alpha_k)}{U_e} = \frac{tan^2(\alpha_e - \alpha_g)}{tan^2(\alpha_e + \alpha_g)}$ $

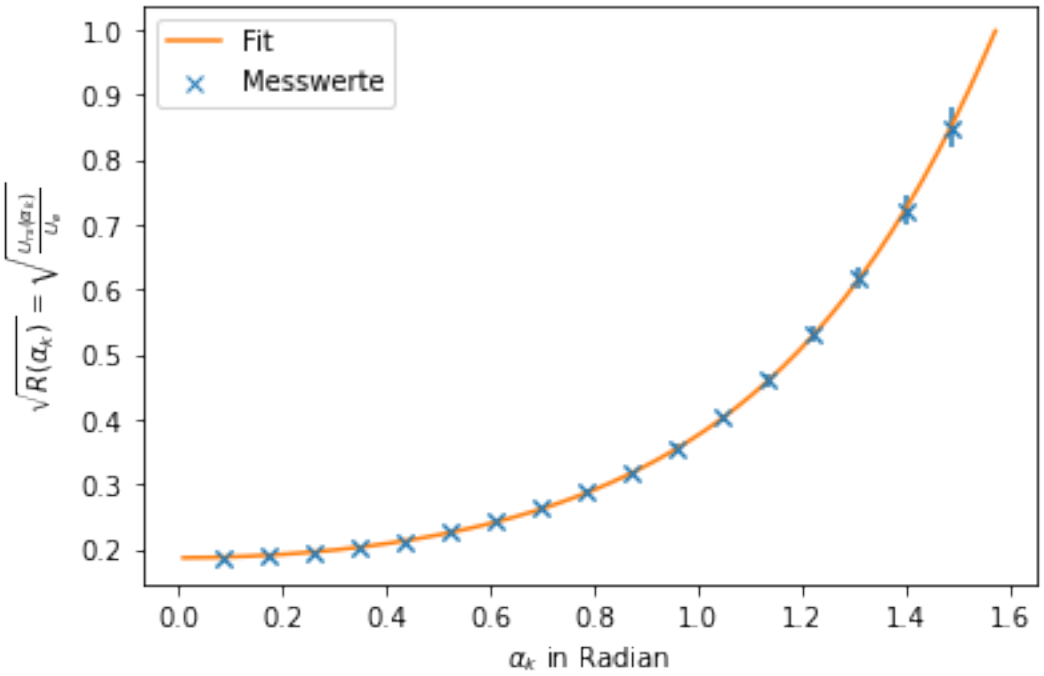
[44]:
```
def U_rs(alpha_e, n2):
    n1 = 1.00028
    alpha_g = np.arcsin(n1/n2*np.sin(alpha_e))
    U = np.sin(alpha_e-alpha_g)**2/(np.sin(alpha_e+alpha_g)**2)
    return U
def U_rp(alpha_e, n2):
    n1 = 1.00028
    alpha_g = np.arcsin(n1/n2*np.sin(alpha_e))
    U = np.tan(alpha_e-alpha_g)**2/(np.tan(alpha_e+alpha_g)**2)
    return U
```

[45]:
```
#Fit von n2 des Prismas für senkrecht polarisiertes Licht
plt.scatter(alpha,np.sqrt(U_s/4.4), label="Messwerte",marker='x',linewidths=1)
plt.errorbar(alpha,np.sqrt(U_s/4.4), yerr=u_Us,fmt='o',marker='x')
fit_parameters_1, fit_cov = curve_fit(U_rs,alpha,U_s/4.4, p0 = 1.43)
```

```
fit_uncertainties = fit_cov[0,0]**0.5
plt.plot(alpha_test,np.sqrt(U_rs(alpha_test,*fit_parameters_1)), label="Fit")
plt.xlabel(r'$\alpha _k$'" in Radian")
plt.ylabel(r'$\sqrt{R(\alpha _k)} =  \sqrt{\frac{U_{rs}(\alpha _k)}{U_e}}$')
plt.legend(loc="upper left")
plt.savefig("FitU_s.pdf")
print("n2 =", fit_parameters_1[0], "+/-", fit_uncertainties)
```

```
/home/santi/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4:
RuntimeWarning: invalid value encountered in true_divide
  after removing the cwd from sys.path.
```

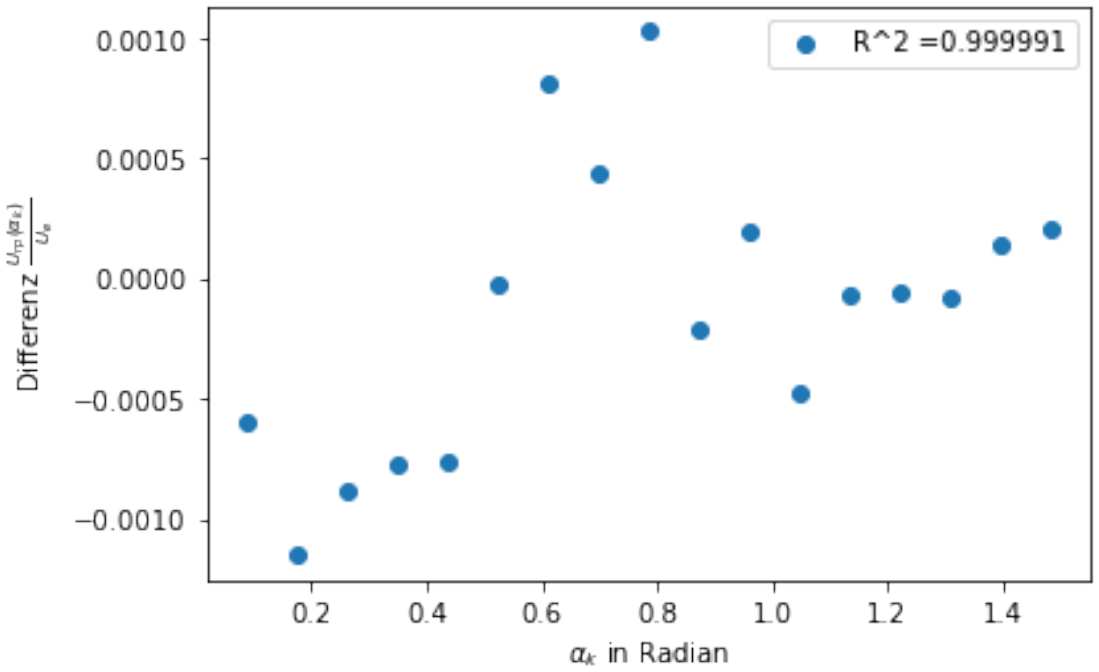
```
n2 = 1.4595307092297318 +/- 0.0002561748091000629
```



[51]:
```
y = np.sqrt(U_s/4.4)
x = alpha
residuals = y - np.sqrt(U_rs(x,*fit_parameters_1))
ss_res = np.sum(residuals**2)
ss_tot = np.sum((y-np.mean(y))**2)
R_2 = 1 - (ss_res / ss_tot)
plt.scatter(x,residuals, label='R^2 ='+str(np.round(R_2,7)))
plt.xlabel(r'$\alpha _k$'" in Radian")
plt.ylabel("Differenz "r'$y- \sqrt{\frac{U_{rs}(\alpha _k)}{U_e}}$')
plt.legend(loc="upper right")
#plt.rcParams["figure.figsize"] = (8,1)
```

```
#plt.savefig("DispersionRes.pdf", bbox_inches = "tight")
print("R^2 =", R_2)
```

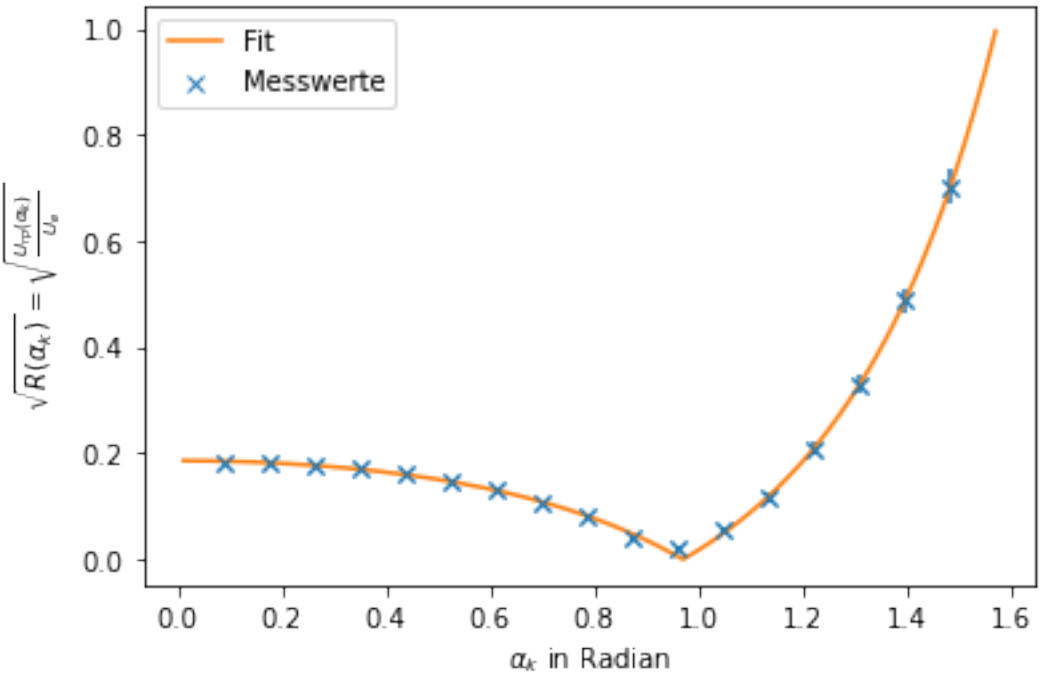
```
R^2 = 0.9999910151673195
```



[47]:
```
#Fit von n2 des Prismas für parallel polarisiertes Licht
plt.scatter(alpha,np.sqrt(U_p/4.4), label="Messwerte", marker='x',linewidths=1)
plt.errorbar(alpha,np.sqrt(U_p/4.4), yerr=u_Us,fmt='o',marker='x')
fit_parameters_2, fit_cov = curve_fit(U_rp,alpha,U_p/4.4, p0 = 1.43)
fit_uncertainties = fit_cov[0,0]**0.5
plt.plot(alpha_test,np.sqrt(U_rp(alpha_test,*fit_parameters_2)), label="Fit")
plt.xlabel(r'$\alpha _k$'" in Radian")
plt.ylabel(r'$\sqrt{R(\alpha _k)} =  \sqrt{\frac{U_{rp}(\alpha _k)}{U_e}}$')
plt.legend(loc="upper left")
plt.savefig("FitU_p.pdf")
print("n2 =", fit_parameters_2[0], "+/-", fit_uncertainties)
```

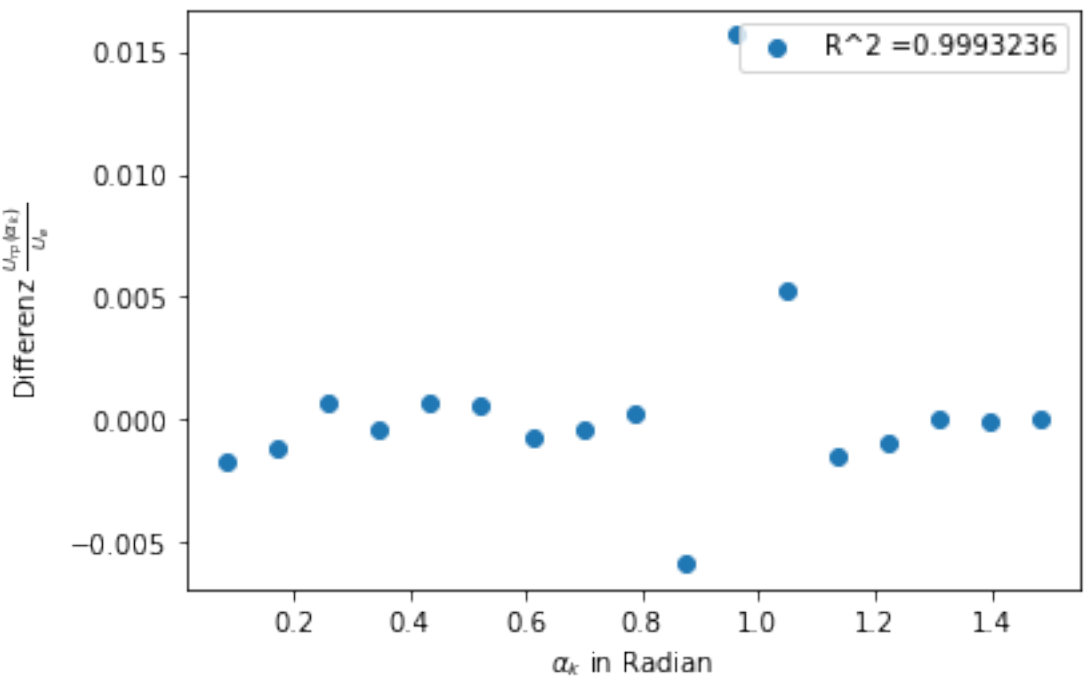
```
/home/santi/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9:
RuntimeWarning: invalid value encountered in true_divide
  if __name__ == '__main__':
```

```
n2 = 1.4585229459182563 +/- 0.0011286186625345184
```



[52]:
```
y = np.sqrt(U_p/4.4)
x = alpha
residuals = y - np.sqrt(U_rp(x,*fit_parameters_1))
ss_res = np.sum(residuals**2)
ss_tot = np.sum((y-np.mean(y))**2)
R_2 = 1 - (ss_res / ss_tot)
plt.scatter(x,residuals, label='R^2 ='+str(np.round(R_2,7)))
plt.xlabel(r'$\alpha _k$'" in Radian")
plt.ylabel("Differenz "r'$y- \sqrt{\frac{U_{rp}(\alpha _k)}{U_e}}$')
plt.legend(loc="upper right")
#plt.rcParams["figure.figsize"] = (8,1)
#plt.savefig("DispersionRes.pdf", bbox_inches = "tight")
print("R^2 =", R_2)
```

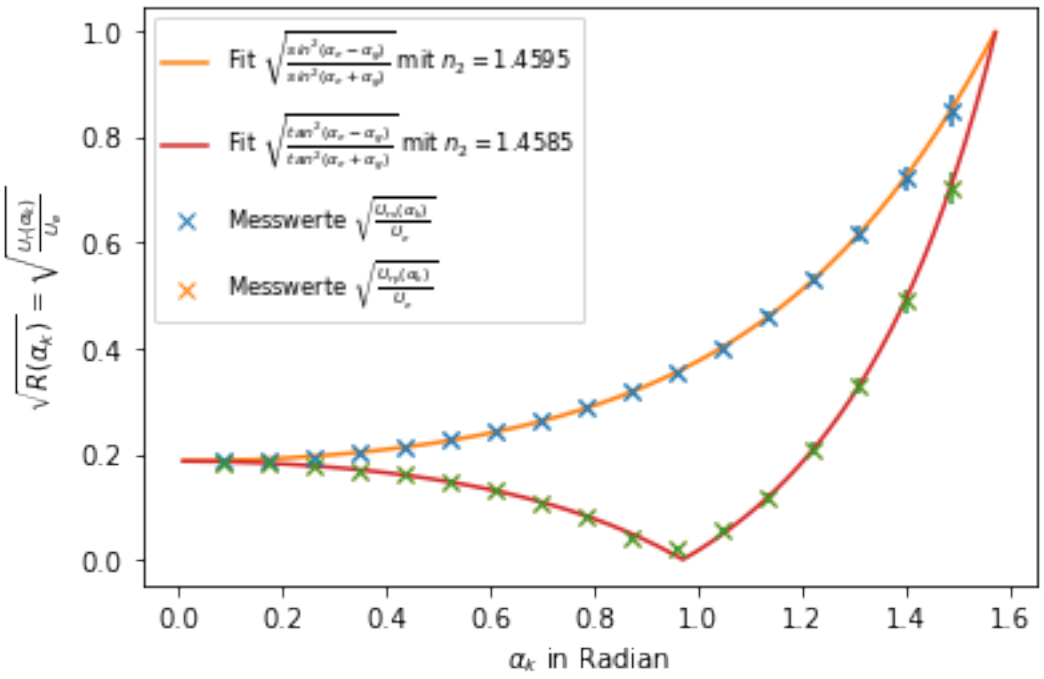
```
R^2 = 0.9993235927315005
```



[49]:
```
#Gemeinsames Fit
#senkrecht Polarisiertes Licht
plt.scatter(alpha,np.sqrt(U_s/4.4), label="Messwerte␣
 ↪"r'$\sqrt{\frac{U_{rs}(\alpha _k)}{U_e}}$',marker='x',linewidths=1)
plt.errorbar(alpha,np.sqrt(U_s/4.4), yerr=u_Us,fmt='o',marker='x')
plt.plot(alpha_test,np.sqrt(U_rs(alpha_test,*fit_parameters_1)), label="Fit "r'$␣
 ↪\sqrt{\frac{sin^2(\alpha _e - \alpha _g)}{sin^2(\alpha _e + \alpha _g)}}$'"␣
 ↪mit "r'$n_2 = 1.4595$')
#Parallel Polarisiertes Licht
plt.scatter(alpha,np.sqrt(U_p/4.4), label="Messwerte␣
 ↪"r'$\sqrt{\frac{U_{rp}(\alpha _k)}{U_e}}$', marker='x',linewidths=1)
plt.errorbar(alpha,np.sqrt(U_p/4.4), yerr=u_Us,fmt='o',marker='x')
plt.plot(alpha_test,np.sqrt(U_rp(alpha_test,*fit_parameters_2)), label="Fit "r'$␣
 ↪\sqrt{\frac{tan^2(\alpha _e - \alpha _g)}{tan^2(\alpha _e + \alpha _g)}}$'"␣
 ↪mit "r'$n_2 = 1.4585$')
#Plot-Einstellungen
plt.xlabel(r'$\alpha _k$'" in Radian")
plt.ylabel(r'$\sqrt{R(\alpha _k)} =  \sqrt{\frac{U_{r}(\alpha _k)}{U_e}}$')
plt.legend(loc="upper left", fontsize = "small")
plt.savefig("FitU_g.pdf")
```

```
/home/santi/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4:
RuntimeWarning: invalid value encountered in true_divide
  after removing the cwd from sys.path.
```

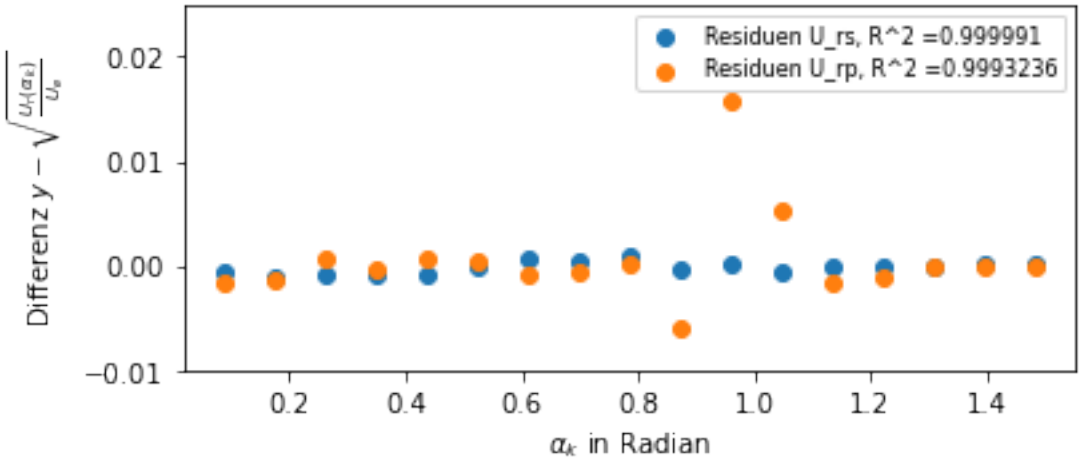
```
/home/santi/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9:
RuntimeWarning: invalid value encountered in true_divide
  if __name__ == '__main__':
```



[72]:
```
#Residuen senkrechte polarisation
y = np.sqrt(U_s/4.4)
x = alpha
residuals = y - np.sqrt(U_rs(x,*fit_parameters_1))
ss_res = np.sum(residuals**2)
ss_tot = np.sum((y-np.mean(y))**2)
R_2 = 1 - (ss_res / ss_tot)
plt.scatter(x,residuals, label='Residuen U_rs, R^2 ='+str(np.round(R_2,7)))
#Residuen parallele polarisation
y = np.sqrt(U_p/4.4)
x = alpha
residuals = y - np.sqrt(U_rp(x,*fit_parameters_1))
ss_res = np.sum(residuals**2)
ss_tot = np.sum((y-np.mean(y))**2)
R_2 = 1 - (ss_res / ss_tot)
plt.scatter(x,residuals, label='Residuen U_rp, R^2 ='+str(np.round(R_2,7)))
plt.xlabel(r'$\alpha _k$'" in Radian")
plt.ylabel("Differenz "r'$y- \sqrt{\frac{U_{r}(\alpha _k)}{U_e}}$')
plt.ylim(-0.01, 0.025)
plt.legend(loc="upper right", prop={'size': 8})
plt.gca().set_aspect(aspect=18)
```

```
#plt.rcParams["figure.figsize"] = (8,1)
plt.savefig("Residuen.pdf", bbox_inches = "tight")
print("R^2 =", R_2)
```

```
R^2 = 0.9993235927315005
```



[50]:
```
#Der Brewster-Winkel ist dann
alpha_b = np.arctan(1.459/1.00028)*360/(2*np.pi)
print(alpha_b)
```

```
55.56574261730882
```

[ ]:
```

```