Ústav fyzikální elektroniky PřF MU

FYZIKÁLNÍ PRAKTIKUM

Fyzikální praktikum 2

Zpracoval: Artem Gorodilov

Naměřeno: 16. listopadu 2023

Obor: Astrofyzika **Skupina:** Čt 8:00 **Testováno:**

Úloha č. **9:** **Měření závislosti indexu lomu skla a vlnové délce metodou minimální derivace**

$T = 22.3$ °C
$p = 974$ hPa
$\varphi = 51$ %

1. Zadání

Změřit úhel lomu optického hranolu N-SF11
Pro určité vlnové délky zjistit derivace a indexy lomu.
Určit Abbeovo číslo.

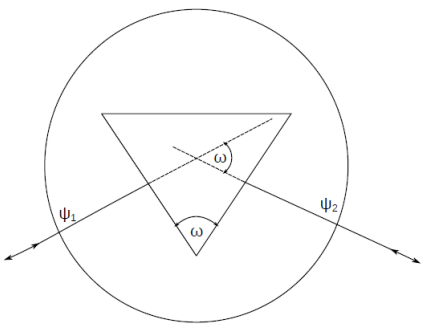
2. Teorie

2.1. Index lomu

Pro určení indexu lomu optického hranolu použijeme metodu minimální derivace. K tomu je třeba určit úhel lomu hranolu, což je úhel mezi dvěma stranami hranolu, jimiž světelný paprsek vstupuje a vychází. Tento úhel lze určit z úhlu, který svírají paprsky kolmé na každou stranu hranolu, viz obrázek (1). Úhel lomu ω se vypočítá podle následujícího vzorce:

$$\omega = 180^o - (\psi_1 - \psi_2) \tag{1}$$

kde ω je úhel lomu a ψ_1 a ψ_2 jsou kolmý úhly dopadu na hranolu.



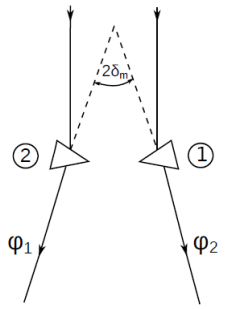
Obrázek (1) Měření úhlu lomu hranolu.

2.2. Úhel minimální derivace a Abbeovo číslo

Úhel minimální derivace δ_m představuje minimální úhel mezi světelným paprskem vstupujícím do hranolu φ_1 a vystupujícím z něj φ_2 viz obrázek (2). Pro jeho zjištění použijeme následující vzorec:

$$\delta_m = \frac{\varphi_1 - \varphi_2}{2} \quad , \quad m \; je \; \lambda_m = 1, 2, 3... \tag{2}$$

kde δ_m je minimální derivace a φ_1 a φ_2 jsou úhly dopadu světla změřené goniometrem.
Je důležité si uvědomit, že úhel minimální derivace bude pro různé vlnové délky λ_m různý. Proto budeme provádět měření pro určité vlnové délky odpovídající spektrálním čarám rtuťové výbojky.



Obrázek (2) Měření minimálního úhlu derivace δ_m od rozdílu úhlů φ_1 a φ_2, při kterém pozorujeme paprsky opouštějící hranol při vstupu první, resp. druhou lomenou stěnou (poloha hranolu). 1 a 2).

Nyní můžeme vypočítat index lomu n materiálu pomocí vzorce:

$$n = \frac{sin\left(\frac{\delta_m + \omega}{2}\right)}{sin\left(\frac{\omega}{2}\right)} \tag{3}$$

Index lomu závisí na vlnové délce v důsledku disperze světla. K disperzi dochází v důsledku závislosti rychlosti šíření záření v prostředí na frekvenci záření. Tento jev lze popsat Cauchyho vztahem:

$$n(\lambda) = A + \frac{B}{\lambda^2} \tag{4}$$

Z toho lze vypočítat Abbeovo číslo ν_d, parametr popisující optické systémy. Abbeho číslo se odvozuje z indexů lomu pro Fraunhoferovy čáry: Žlutá $\lambda_d = 587.6$ [nm], modrá $\lambda_F = 486.1$ [nm] a červená $\lambda_C = 656.3$ [nm] a lze ho vyjádřit následujícím vztahem:

$$\nu_d = \frac{n_d - 1}{n_F - n_C} \tag{5}$$

kde n_d, n_F a n_C je indexy lomu pro vlnové délky λ_d, λ_F a λ_C resp.

3. Měření

3.1. Index lomu

Na základě měření jsme změřili hodnoty kolmých úhlů dopadu ψ_1 a ψ_2, viz tabulku (1), a stejné úhly, ale pro tři jiné polohy hranolu na goniometru (třikrát jsme hranol mírně otočili ve směru hodinových ručiček), viz tabulku (2). Hodnoty úhlů lomu ω byly vypočteny podle vzorce (1).

ψ_1 [^{o}]	ψ_2 [^{o}]	ω [^{o}]
260.8836(3)	140.8831(3)	59.9994(4)
260.8825(3)	140.8825(3)	60.0000(4)
260.8831(3)	140.8842(3)	60.0011(4)

Tabulka (1) Naměřené hodnoty kolmých úhlů dopadu ψ_1 a ψ_2 a vypočtené úhly lomu ω.

n	ψ_1 [^{o}]	ψ_2 [^{o}]	ω [^{o}]
1	283.4581(3)	163.4586(3)	60.0006(4)
2	278.5972(3)	158.5989(3)	60.0017(4)
3	272.9300(3)	152.9289(3)	59.9989(4)

Tabulka (2) Naměřené hodnoty kolmých úhlů dopadu ψ_1 a ψ_2 a vypočtené úhly lomu ω pro různé úhly natočení hranolu na goniometru n.

Odtud můžeme zjistit hodnoty úhlu lomu ω:

$$\omega = 60.000(1)$$

3.2. Úhel minimální derivace a Abbeovo číslo

Poté jsme změřili úhly dopadu φ_1 a φ_2 pro situaci znázorněnou na obrázku (2) pro různé vlnové délky λ_m odpovídající různým barvám, viz tabulka (3).

Poté jsme vypočítali úhel minimální derivace a index lomu hranolu pomocí vzorců (2) a (3), viz tabulka (4).

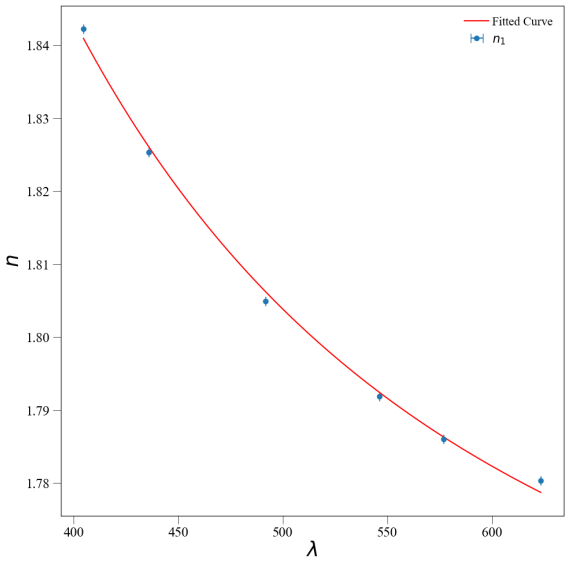
Barvá	λ [nm]	φ_1 [^{o}]	φ_2 [^{o}]
Červená	623.4	275.9372(3)	144.3614(4)
Žlutá	576.9	276.6625(3)	143.6414(4)
Zelená	546.1	277.4144(3)	142.8972(4)
Modrozelená	491.6	279.1150(3)	141.1753(4)
Modrá	435.8	281.8831(3)	138.3747(4)
Fialová	404.6	284.3161(3)	135.9431(4)

Tabulka (3) Naměřené hodnoty kolmých úhlů dopadu φ_1 a φ_2 pro různé vlnové délky λ_m.

Barvá	λ [nm]	σ_m [^{o}]	n
Červená	623.4	65.7879(2)	1.78032(2)
Žlutá	576.9	66.5106(2)	1.78604(2)
Zelená	546.1	67.2586(2)	1.79187(2)
Modrozelená	491.6	68.9699(2)	1.80494(2)
Modrá	435.8	71.7542(2)	1.82534(2)
Fialová	404.6	74.1865(2)	1.84227(2)

Tabulka (4) Vypočtené úhly minimální derivace σ_m a indexy lomu hranolu n pro různé vlnové délky λ_m.

Vykreslíme závislost indexu lomu n na vlnové délce λ_m a sestrojíme fit podle vzorce (4).



Obrázek (3) Závislost indexu lomu n na vlnové délce λ_m.

Proto byly získány následující hodnoty konstant A a B:

$$A = 1.73(2)$$

$$B = 1.76(4) \times 10^4$$

Dále podle vzorce (4) zjistíme indexy lomu hranolů pro Fraunhoferovy čáry n_d, n_F a n_C a poté podle vzorce (5) zjistíme Abbeho číslo ν_d:

$$n_d = 1.784(2)$$
$$n_F = 1.808(2)$$
$$n_C = 1.774(2)$$
$$\nu_d = 23.5(5)$$

K výpočtu veličin a jejich nejistot byla použita knihovna Uncertinties pro Python: pypi.org/project/uncertainties. Kód je přiložen k protokolu.

4. Závěr

4.1. Index lomu

Získali jsme hodnotu úhlu lomu $\omega = 60.000(1)$. S přihlédnutím k chybě hodnoty je velmi přesná.

4.2. Úhel minimální derivace a Abbeovo číslo

Po výpočtech byly získány následující hodnoty indexů lomu pro Fraunhoferovy čáry: $n_d = 1.784(2)$, $n_F = 1.808(2)$ a $n_C = 1.774(2)$.
Rovněž byla získána následující hodnota Abbeovo čísla: $\nu_d = 23.5(5)$, která odpovídá tabulkové hodnotě $\nu_d = 25.68$ uvedené na stránce: schott.com/shop/advanced-optics/en/Optical-Glass/N-SF11/c/glass-N-SF11

K výpočtu chyb byl použit následující kód:

```
#Importing the libraries

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import stats
from scipy.optimize import curve_fit
from uncertainties import *
from uncertainties.umath import *

#Reading data

lom = pd.read_excel('lom.xlsx')
waves = pd.read_excel('waves.xlsx')

# Constants and values

phi_1_1 = [283 , 27 , 29]
phi_2_1 = [163 , 27 , 31]

phi_1_2 = [278 , 35 , 50]
phi_2_2 = [158 , 35 , 56]

phi_1_3 = [272 , 55 , 48]
phi_2_3 = [152 , 55 , 44]

lambda_m = [623.4, 576.9, 546.1, 491.6, 435.8, 404.6]
lambda_frau = np.array([656.3, 486.1, 587.6])

radians = np.pi/180
degrees = 180/np.pi

def dms_to_decimal(degrees, minutes, seconds):
    decimal_degrees = degrees + (minutes / 60) + (seconds / 3600)
    return decimal_degrees

# Calculation

lom['phi_1_decimal'] = dms_to_decimal(lom['phi_1_deg'], lom['phi_1_min'], lom['phi_1_sec'])
lom['phi_2_decimal'] = dms_to_decimal(lom['phi_2_deg'], lom['phi_2_min'], lom['phi_2_sec'])

waves['phi_1_decimal'] = dms_to_decimal(waves['phi_1_deg'], waves['phi_1_min'], waves['phi_1_sec'])
waves['phi_2_decimal'] = dms_to_decimal(waves['phi_2_deg'], waves['phi_2_min'], waves['phi_2_sec'])
waves['phi_2_2_decimal'] = dms_to_decimal(waves['phi_2_2_deg'], waves['phi_2_2_min'], waves['
    phi_2_2_sec'])

phi_1_1_decimal = ufloat(dms_to_decimal(phi_1_1[0], phi_1_1[1], phi_1_1[2]), 0.000277778)
phi_2_1_decimal = ufloat(dms_to_decimal(phi_2_1[0], phi_2_1[1], phi_2_1[2]), 0.000277778)

# print('phi_1_1_decimal = ', phi_1_1_decimal)
# print('phi_2_1_decimal = ', phi_2_1_decimal)

phi_1_2_decimal = ufloat(dms_to_decimal(phi_1_2[0], phi_1_2[1], phi_1_2[2]), 0.000277778)
phi_2_2_decimal = ufloat(dms_to_decimal(phi_2_2[0], phi_2_2[1], phi_2_2[2]), 0.000277778)

# print('phi_1_2_decimal = ', phi_1_2_decimal)
# print('phi_2_2_decimal = ', phi_2_2_decimal)

phi_1_3_decimal = ufloat(dms_to_decimal(phi_1_3[0], phi_1_3[1], phi_1_3[2]), 0.000277778)
phi_2_3_decimal = ufloat(dms_to_decimal(phi_2_3[0], phi_2_3[1], phi_2_3[2]), 0.000277778)

# print('phi_1_3_decimal = ', phi_1_3_decimal)
# print('phi_2_3_decimal = ', phi_2_3_decimal)

lom_phi_1_list =[]
lom_phi_2_list =[]
for ii,ID in enumerate(lom['phi_1_decimal']):
    lom_phi_1_list.append(ufloat(lom['phi_1_decimal'][ii], 0.000277778))
    lom_phi_2_list.append(ufloat(lom['phi_2_decimal'][ii], 0.000277778))
lom['phi_1_comb'] = lom_phi_1_list
lom['phi_2_comb'] = lom_phi_2_list

waves_phi_1_list =[]
waves_phi_2_list =[]
waves_phi_2_2_list =[]
for ii,ID in enumerate(waves['phi_1_decimal']):
    waves_phi_1_list.append(ufloat(waves['phi_1_decimal'][ii], 0.000277778))
    waves_phi_2_list.append(ufloat(waves['phi_2_decimal'][ii], 0.000277778))
waves['phi_1_comb'] = waves_phi_1_list
waves['phi_2_comb'] = waves_phi_2_list

lom['omega'] = 180 - (lom['phi_1_comb'] - lom['phi_2_comb'])

omega_phi_1_1 = 180 - (phi_1_1_decimal - phi_2_1_decimal)
omega_phi_1_2 = 180 - (phi_1_2_decimal - phi_2_2_decimal)
omega_phi_1_3 = 180 - (phi_1_3_decimal - phi_2_3_decimal)

# print('omega_phi_1_1 = ', omega_phi_1_1)
# print('omega_phi_1_2 = ', omega_phi_1_2)
# print('omega_phi_1_3 = ', omega_phi_1_3)

omega_mean = ufloat(np.mean(np.array([np.mean(np.array(lom['omega'].apply(lambda x: x.nominal_value))
    ), omega_phi_1_1.nominal_value, omega_phi_1_2.nominal_value, omega_phi_1_3.nominal_value])), np.
    sqrt(np.std(np.array([np.mean(np.array(lom['omega'].apply(lambda x: x.nominal_value))),
    omega_phi_1_1.nominal_value, omega_phi_1_2.nominal_value, omega_phi_1_3.nominal_value]))**2 +
    0.000277778**2))
print('omega_mean =', omega_mean)

waves['sigma_1'] = (waves['phi_1_comb'] - waves['phi_2_comb']) / 2

waves_n_1_list =[]
for ii,ID in enumerate(waves['sigma_1']):
```

```
    waves_n_1_list.append(sin(radians * (waves['sigma_1'][ii] + omega_mean)/2)/sin(radians *
        omega_mean/2))
waves['n_1'] = waves_n_1_list

# print(lom)
# print(waves)

# Define the polynomial function

def polynomial_fit(lambda_values, A, B):
    return A + B / (lambda_values**2)

# Use curve_fit to find the parameters A and B
initial_guess = [1.5, 5000]  # Initial guess for parameters A and B
params, covariance = curve_fit(polynomial_fit, lambda_m, waves['n_1'].apply(lambda x: x.nominal_value
    ), p0=initial_guess)

# Extract the optimized parameters
A_optimized, B_optimized = params
A_error, B_error = np.sqrt(np.diag(covariance))

A_comb = ufloat(A_optimized, A_error)
B_comb = ufloat(B_optimized, B_error)

# Print the optimized parameters
print('A=', A_comb)
print('B=', B_comb)

#Best-fit line

lambda_val = np.linspace(404.6, 623.4, 1000)
n_val = polynomial_fit(lambda_val, A_optimized, B_optimized)

n_frau = polynomial_fit(lambda_frau, A_comb, B_comb)
print('n_frau=', n_frau)

nu = (n_frau[2] - 1)/(n_frau[1] - n_frau[0])
print('nu=', nu)
```