

DI01-PW3

October 21, 2025

1 Grundlagen – Teil 1: Dispersion der Prisma

Beim Durchgang eines Lichtstrahls durch ein Prisma wird der Strahl an beiden Grenzflächen gebrochen. Das Verhalten des Lichts wird durch das Brechungsgesetz von Snellius beschrieben:

$$n_1 \sin(\alpha) = n_2 \sin(\beta)$$

Hierbei gilt: - n_1, n_2 - Brechungsindizes der beteiligten Medien (Luft \rightarrow Glas \rightarrow Luft) - α - Einfallswinkel - β - Brechungswinkel

Für eine gleichseitige Prisma mit Scheitelwinkel ε (z.B. 60°) ergibt sich für die gesamte Ablenkung δ des Strahls:

$$\delta = \alpha_1 + \alpha_2 = \epsilon$$

Im allgemeinen Fall ist der Strahlengang asymmetrisch, d. h. die Einfalls- und Austrittswinkel unterscheiden sich. Bei der minimalen Ablenkung δ_{min} verläuft der Strahl jedoch **symmetrisch**:

$$\alpha_1 = \alpha_2 \text{ und } \beta_1 = \beta_2 = \frac{\varepsilon}{2}$$

Unter dieser Bedingung lässt sich der **Brechungsindex des Prismamaterials** berechnen zu:

$$n = \frac{\sin(\frac{\delta_{min} + \varepsilon}{2})}{\sin(\frac{\varepsilon}{2})}$$

— Da der Brechungsindex von der Wellenlänge λ abhängt, spricht man von **Dispersion**. Für normale Dispersion gilt:

$$n(\lambda_{blau}) > n(\lambda_{rot})$$

also nimmt n mit zunehmender Wellenlänge ab.

Zur quantitativen Beschreibung wird häufig das **Cauchy-Gesetz** verwendet:

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

2 Grundlagen – Teil 2: Absorptionsspektroskopie

Beim Durchgang von Licht durch eine absorbierende Substanz wird ein Teil der einfallenden Strahlung geschwächt.

Die Intensität des Lichtes nimmt dabei exponentiell mit der durchlaufenen Schichtdicke ab.

Dieses Verhalten wird durch das **Lambert-Beer'sche Gesetz** beschrieben:

$$I(d, \lambda) = I_0(\lambda) e^{-\alpha(\lambda) d}$$

Hierbei gilt:

- $I_0(\lambda)$ – einfallende Intensität (ohne Probe)
- $I(d, \lambda)$ – Intensität nach Durchgang durch die Probe
- $\alpha(\lambda)$ – Absorptionskoeffizient
- d – Schichtdicke der absorbierenden Substanz

Zur experimentellen Bestimmung wird meist der **Transmissionsgrad** (T) verwendet:

$$T(\lambda) = \frac{I(d, \lambda)}{I_0(\lambda)} = e^{-\alpha(\lambda) d}$$

Aus dem Transmissionsgrad lässt sich die sogenannte **Optische Dichte (OD)** berechnen:

$$OD(\lambda) = -\log_{10} T(\lambda) = \log_{10} \left(\frac{I_0(\lambda)}{I(d, \lambda)} \right)$$

Eine hohe optische Dichte bedeutet starke Absorption,
während eine kleine OD auf hohe Transmission hinweist.

Bestimmte Stoffe absorbieren Licht nur bei charakteristischen Wellenlängen.

Diese **Absorptionsbanden** sind eine Folge von elektronischen Übergängen in den Atomen oder Ionen.

Für wässrige Lösungen seltener Erdionen wie **Neodym (Nd^{3+})** und **Praseodym (Pr^{3+})** liegen diese Banden im sichtbaren Bereich und sind daher leicht spektroskopisch messbar.

Die typischen Absorptionslinien sind (nach der Versuchsanleitung):

Ion	Charakteristische Absorptionslinien (in nm)
Nd^{3+}	510, 522, 578, 740, 799, 868
Pr^{3+}	444, 468, 481, 590

Die Lage der Maxima in diesem **Absorptionsspektrum** erlaubt die eindeutige Identifikation des Ions in der Probe.

So zeigt ein Spektrum mit starken Maxima bei etwa 520 nm und 580 nm auf die Anwesenheit von **Neodym**, während Linien bei etwa 445 nm und 590 nm auf **Praseodym** hinweisen.

```
[11]: """
PW3 - Manuelle Dateneingabe (Arrays/Listen)
=====
Dieses Skript analysiert:
1) Dispersionskurve einer Prismamessung aus manuell eingegebenen Arrays
2) Absorptionsspektroskopie (OD) aus manuell eingegebenen Arrays
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.signal import find_peaks

# -----
# DATENEINGABE
# -----

# --- (1) PRISMA

# Wellenlängen der Kalibrationslinien (nm):
wavelength_nm = np.array([404, 436, 491, 546, 579], dtype=float)

# Ablesewinkel (Grad) - Richtung ohne Prisma:
phi1_deg = np.array([354.9, 354.9, 354.9, 354.075, 354.075], dtype=float)

# Ablesewinkel (Grad) - Richtung in minimaler Ablenkung:
phi2_deg = np.array([47.67, 45.85, 44.53, 44.33, 43.91], dtype=float)

delta_min = 360 + phi2_deg-phi1_deg

# Brechender Winkel des Prismas (Grad) - gleicher Wert für alle Punkte:
epsilon_deg = 60.0

# (Optional) Ableseunsicherheit pro Winkel (Winkelminute) - gleicher Wert für
↳ alle Punkte:
phi_unc_arcmin = 20 # z.B. 1'

# -----
# HILFSFUNKTIONEN (Formeln gemäß Praktikumsbeschreibung)
# -----
```

```

def deg2rad(x):
    """Grad → Radiant."""
    return np.deg2rad(x)

def arcmin2deg(x_arcmin):
    """Winkelminute → Grad."""
    return x_arcmin / 60.0

def calc_delta_min(phi1, phi2):
    """Minimale Ablenkung _min = |phi2 - phi1| (in Grad)."""
    return np.abs(phi2 - phi1)

def n_from_prism(delta_min_deg, epsilon_deg):
    """
Brechungsindex: n = sin((+_min + )/2) / sin(/2), Winkel in Grad.
"""
    num = np.sin(deg2rad((delta_min_deg + epsilon_deg)/2.0))
    den = np.sin(deg2rad(epsilon_deg/2.0))
    return num/den

def sigma_n_from_delta(delta_min_deg, epsilon_deg, sigma_delta_deg):
    """
Fehlerfortpflanzung nur über _min ( als exakt angenommen):
dn/d = (1/2)*cos((+ )/2)/sin(/2) (Argumente in Radiant)
"""
    eps2 = deg2rad(epsilon_deg/2.0)
    x = deg2rad((delta_min_deg + epsilon_deg)/2.0)
    dnddelta = 0.5*np.cos(x)/np.sin(eps2)
    return np.abs(dnddelta) * deg2rad(sigma_delta_deg)

def cauchy(lambda_nm, A, B, C=0.0):
    """Cauchy-Dispersion: n( ) = A + B/ ^2 + C/ ( in nm)."""
    lam = np.array(lambda_nm, float)
    return A + B/lam**2 + C/lam**4

def fit_cauchy(lambda_nm, n_vals, use_C=True):
    """Nichtlinearer Fit der Cauchy-Gleichung."""
    lam = np.array(lambda_nm, float)
    nv = np.array(n_vals, float)
    if use_C:
        p0 = [np.median(nv), 1e4, 1e9]
        popt, pcov = curve_fit(cauchy, lam, nv, p0=p0, maxfev=10000)
    else:
        def cauchy2(x, A, B): return A + B/x**2
        p0 = [np.median(nv), 1e4]
        popt2, pcov2 = curve_fit(cauchy2, lam, nv, p0=p0, maxfev=10000)
        popt = [popt2[0], popt2[1], 0.0]

```

```

        pcov = np.zeros((3,3)); pcov[:2,:2] = pcov2
    return popt, pcov

def compute_OD(lam, Iref, Isam, smooth_pts=None):
    """
    Optische Dichte:  $T = Isam/Iref$ ,  $OD = -\log_{10}(T)$ .
    'lam' muss zu Iref/Isam passen (gleiche Länge).
    Optional Glättung über gleitendes Mittel (smooth_pts).
    """
    lam = np.array(lam, float)
    Iref = np.array(Iref, float)
    Isam = np.array(Isam, float)

    T = np.clip(Isam/np.maximum(Iref, 1e-12), 1e-12, None)
    OD = -np.log10(T)

    if smooth_pts and smooth_pts > 1:
        k = int(smooth_pts)
        ker = np.ones(k)/k
        OD = np.convolve(OD, ker, mode="same")
    return lam, OD

# Referenzlinien (nm) für grobe Identifikation (Nd/Pr):
ND_LINES = np.array([510, 522, 578, 740, 799, 868], float)
PR_LINES = np.array([444, 468, 481, 590], float)

def find_od_peaks(lam, OD, prominence=0.03, distance_nm=1.0):
    """
    Peaks in OD() (AbsorptionsMAXIMA) finden.
    prominence: relative Prominenz auf normalisierter OD.
    distance_nm: Mindestabstand der Peaks in nm.
    """
    lam = np.array(lam, float)
    OD = np.array(OD, float)
    ODn = (OD - np.nanmin(OD)) / (np.nanmax(OD) - np.nanmin(OD) + 1e-12)
    dl = np.median(np.diff(lam))
    dist_pts = max(1, int(distance_nm/max(dl, 1e-9)))
    idx, props = find_peaks(ODn, prominence=prominence, distance=dist_pts)
    return lam[idx], OD[idx], props

def match_lines(peaks_nm, ref_lines, tol_nm=2.0):
    """Ordnet Peaks Referenzlinien zu, falls  $|\Delta| \leq tol\_nm$ ."""
    results = []
    for r in ref_lines:
        i = np.argmin(np.abs(peaks_nm - r))
        if np.abs(peaks_nm[i] - r) <= tol_nm:
            results.append((r, peaks_nm[i], peaks_nm[i]-r))

```

```

    return results

# -----
# (1) PRISMA - AUSWERTUNG
# -----

# Konsistenzprüfungen
assert len(wavelength_nm)==len(phi1_deg)==len(phi2_deg), "Alle PrismaListen_
    ↪ müssen gleich lang sein."

delta_min_deg = calc_delta_min(phi1_deg, phi2_deg)
n_vals = n_from_prism(delta_min_deg, epsilon_deg)

# Unsicherheit (wenn gewünscht):  $\sigma_n = \sqrt{\sigma_{\phi1}^2 + \sigma_{\phi2}^2}$ , hier beide_
    ↪ gleich angenommen
sigma_phi_deg = arcmin2deg(phi_unc_arcmin)
sigma_delta_deg = np.sqrt(2) * sigma_phi_deg
n_unc = sigma_n_from_delta(delta_min_deg, epsilon_deg, sigma_delta_deg)

# Cauchy-Fit (mit C-Term; bei sehr wenigen Punkten ggf. use_C=False wählen)
popt, pcov = fit_cauchy(wavelength_nm, n_vals, use_C=True)
A,B,C = popt

print("[Prisma] Cauchy-Parameter:", f"A={A:.6f}, B={B:.1f}, C={C:.3e}")
print(f"Delta_min: {delta_min}")

print(f"err_unc: {n_unc}")
# Plot Dispersionskurve
plt.figure()
if np.isfinite(n_unc).any():
    plt.errorbar(wavelength_nm, n_vals, yerr=n_unc, fmt="o", label="Messpunkte_
        ↪ (mit Uns.)")
else:
    plt.plot(wavelength_nm, n_vals, "o", label="Messpunkte")

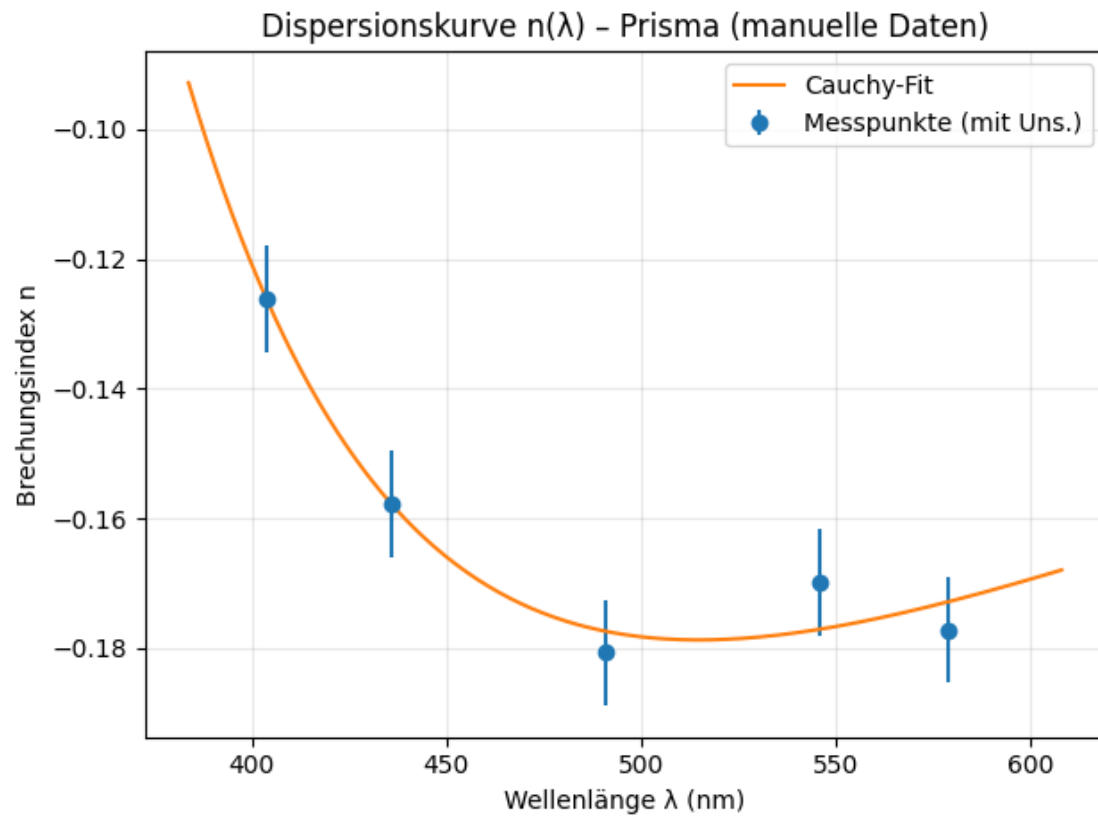
lam_fit = np.linspace(min(wavelength_nm)*0.95, max(wavelength_nm)*1.05, 400)
plt.plot(lam_fit, cauchy(lam_fit, *popt), "-", label=f"Cauchy-Fit")
plt.xlabel("Wellenlänge (nm)")
plt.ylabel("Brechungsindex n")
plt.title("Dispersionskurve n() - Prisma (manuelle Daten)")
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()

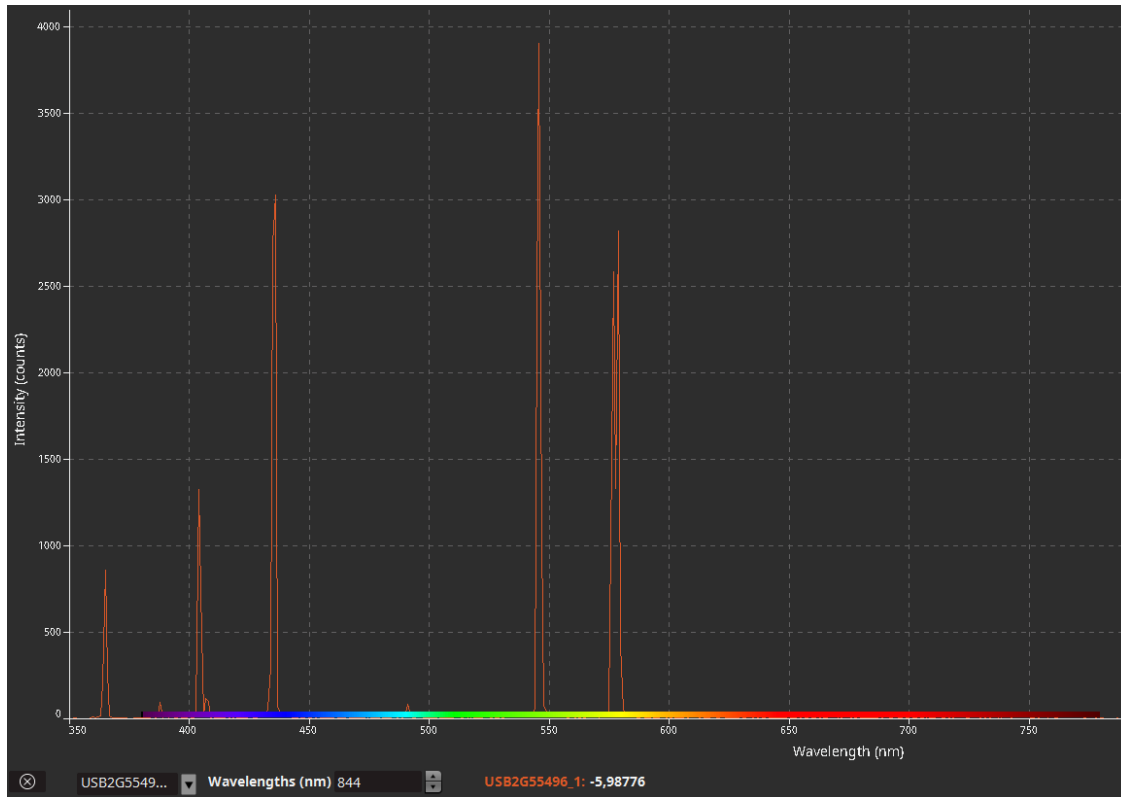
```

```

[Prisma] Cauchy-Parameter: A=-0.044126, B=-71359.7, C=9.455e+09
Delta_min: [52.77  50.95  49.63  50.255 49.835]
err_unc: [0.00821119 0.00820192 0.00819389 0.00819783 0.00819521]

```





2.0.1 Wir denken, dass B Neodym ist:

