priprava

June 9, 2025

1 Optična rotacija v sladkorni vodi

1.1 Uvod

Optična rotacija je pojav, kjer kiralne molekule, kot so saharoza, zavrtijo ravnino polarizirane svetlobe. Ta pojav je odvisen od: - koncentracije snovi (c), - dolžine poti svetlobe (l), - valovne dolžine svetlobe (λ).

1.2 Osnovna enačba

Kot rotacije α je povezan s specifično rotacijo $[\alpha]_\lambda$ preko:

$\alpha(\lambda) = [\alpha]_\lambda \cdot c \cdot l$

Specifična rotacija je funkcija valovne dolžine in jo lahko približamo z Drudejevim modelom:

$[\alpha](\lambda) = \frac{k\lambda^2}{\lambda^2 - A^2}$

kjer sta (k) in (A) parametra, ki ju bomo določili s prilagajanjem modela na meritve.

1.3 Namen eksperimenta

Merili bomo kot rotacije za različne koncentracije saharoze pri dveh valovnih dolžinah (rdeča ~650 nm, zelena ~532 nm). Poleg tega bomo primerjali dve vrsti medu, naravnega in sintetičnega, da pokažemo razliko v optični aktivnosti.

2 Korak 2: Priprava podatkov za eksperiment in zapis napak

2.1 1. Meritve in negotovosti

Merjene količine: - Koncentracija saharoze, c (npr. v g/mL) - Valovna dolžina, λ (v nm, poznana od lasera) - Kot rotacije, α (v stopinjah, iz polarimetra)

Negotovosti: - Absolutna napaka koncentracije, Δc
Primer: če tehtnica meri s 0.001 g natančnostjo in pripravljaš raztopino, oceni to napako. - Absolutna napaka kota rotacije, $\Delta\alpha$
Podatki od polarimetra (npr. $\pm 0.05°$) ali ocena glede na merilni instrument. - Napaka valovne dolžine ni potrebna, če uporabljaš laserske diode, ker so valovne dolžine zelo točne.

2.2 2. Postopek priprave podatkov z napakami

Izpis meritve z napako:

$$c \pm \Delta c, \quad \lambda, \quad \alpha \pm \Delta\alpha$$

Primer (za eno meritev):

koncentracija (g/mL)	napaka koncentracije (g/mL)	valovna dolžina (nm)	kot rotacije (°)	napaka kota (°)
0.05	0.001	650	3.50	0.05

2.3 3. Enačbe za nadaljnjo analizo

Kot rotacije je povezan s koncentracijo in dolžino poti (zaenkrat je dolžina poti konstantna, npr. 1 dm):

$$\alpha = [\alpha]_\lambda \cdot c \cdot l$$

Če boš meril pri različnih koncentracijah, lahko izračunaš specifično rotacijo $[\alpha]_\lambda$ s formulo:

$$[\alpha]_\lambda = \frac{\alpha}{c \cdot l}$$

2.4 5. Primer navodil za pripravo podatkov

1. Pripravi sladkorno raztopino različnih koncentracij c (npr. 0.01, 0.03, 0.05 g/mL). Pri pripravi upoštevaj merilno napako tehtnice ali volumna.
2. Izmeri kot rotacije α pri vsaki koncentraciji za laserja z valovnimi dolžinami $\lambda_1 = 650\,\text{nm}$ in $\lambda_2 = 532\,\text{nm}$.
3. Zapiši meritve v tabelo z vsemi vrednostmi in njihovimi napakami.
4. Izračunaj specifično rotacijo po formuli zgoraj ter izračunaj njeno negotovost.

$[\alpha](\lambda) = \frac{k\lambda^2}{\lambda^2 - A^2}$

3 Analiza specifične rotacije z uporabo Drudejeve enačbe

Merili smo specifično rotacijo $[\alpha](\lambda)$ pri različnih valovnih dolžinah. Za te podatke želimo določiti konstanti k in A, ki opisujeta disperzijo optične aktivnosti po enačbi:

$$[\alpha](\lambda) = \frac{k\lambda^2}{\lambda^2 - A^2}$$

kjer je: - λ valovna dolžina (v nm), - $[\alpha](\lambda)$ specifična rotacija (v ° / (g/mL·dm)), - k, A sta konstanti, ki ju bomo določili z ujemanjem.

Spodaj izvedemo nelinearno prileganje (curve fitting) z uporabo `scipy.optimize.curve_fit`.

[265]:
```
# Uvoz knjižnic
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

3.1 Podatki iz eksperimenta

Zabeležili smo naslednje podatke: - koncentracija je enaka za vse meritve, zato lahko primerjamo le specifično rotacijo.

[266]:
```
# Podatki (lambda v nm, alpha v deg/(g/mL·dm))
lambda_vals = np.array([532, 589, 650])  # nm
alpha_vals = np.array([25.2, 20.0, 17.1])  # specifična rotacija

# Če imaš napake, lahko dodaš:
# alpha_err = np.array([0.3, 0.3, 0.3])
```

3.2 Definicija modela

Uporabimo model:

$$[\alpha](\lambda) = \frac{k\lambda^2}{\lambda^2 - A^2}$$

[267]:
```
# Drudejeva enačba
def drude_model(lam, k, A):
    return (k * lam**2) / (lam**2 - A**2)
```

3.3 Prileganje modela eksperimentalnim podatkom

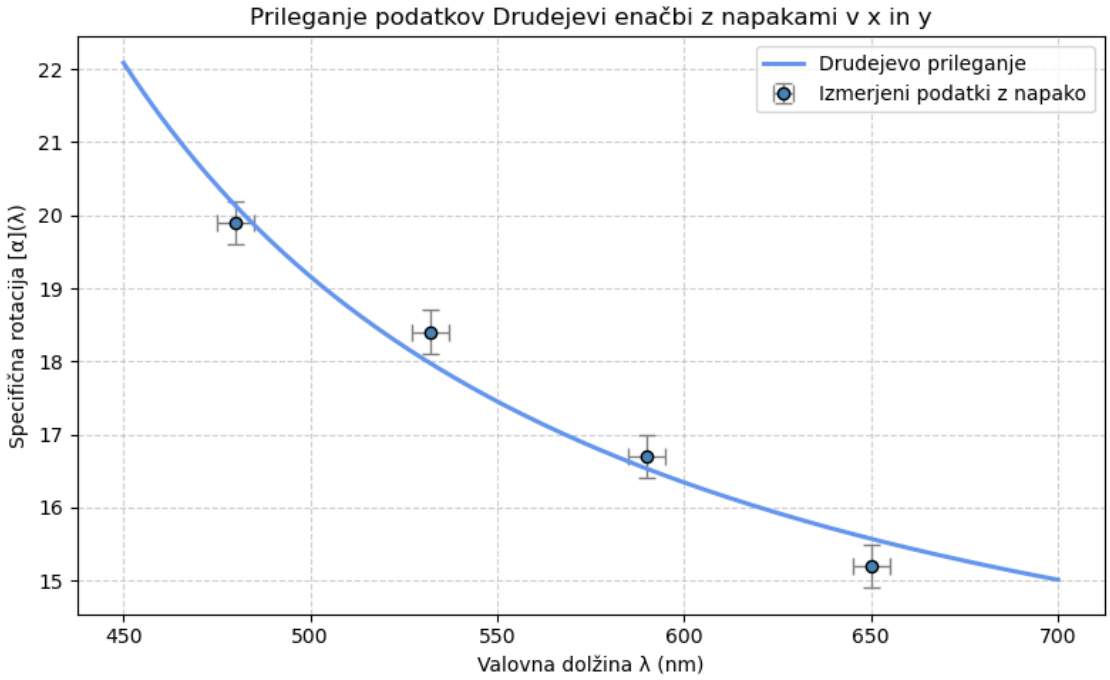
[268]:
```
# Tu vnesemo svoje podatke
lambda_vals = np.array([650, 590, 532, 480])  # valovne dolžine v nm
lambda_errs = np.array([5, 5, 5, 5])  # napake valovnih dolžin
alpha_vals = np.array([15.2, 16.7, 18.4, 19.9])  # specifična rotacija
alpha_errs = np.array([0.3, 0.3, 0.3, 0.3])      # napake specifične rotacije
```

[269]:
```
# Uvozimo potrebne knjižnice
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Drudejeva funkcija
def drude(lambda_nm, k, A):
```

```
    return (k * lambda_nm**2) / (lambda_nm**2 - A**2)


# Prileganje z upoštevanjem napak v y (x napake ne vplivajo na fit)
params, cov = curve_fit(drude, lambda_vals, alpha_vals, sigma=alpha_errs,␣
↪absolute_sigma=True, p0=(1e4, 200))
k_fit, A_fit = params
k_err, A_err = np.sqrt(np.diag(cov))

# Izpis rezultatov
print("Ujemajoči parametri:")
print(f"  k = {k_fit:.2f} ± {k_err:.2f}")
print(f"  A = {A_fit:.2f} ± {A_err:.2f} nm")

# Priprava za prikaz prileganja
lambda_fit = np.linspace(450, 700, 300)
alpha_fit = drude(lambda_fit, k_fit, A_fit)

# Risanje grafa z error bar-i v obeh smereh
plt.figure(figsize=(8, 5))

plt.errorbar(
    lambda_vals,
    alpha_vals,
    xerr=lambda_errs,
    yerr=alpha_errs,
    fmt='o',
    markersize=6,
    markerfacecolor='steelblue',
    markeredgecolor='black',
    ecolor='gray',
    elinewidth=1,
    capsize=4,
    label='Izmerjeni podatki z napako'
)

# Prileganje funkcije
plt.plot(lambda_fit, alpha_fit, color='cornflowerblue', linewidth=2.2,␣
↪label='Drudejevo prileganje')

# Oznake in estetika
plt.xlabel('Valovna dolžina λ (nm)')
plt.ylabel('Specifična rotacija [α](λ)')
plt.title('Prileganje podatkov Drudejevi enačbi z napakami v x in y')
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.tight_layout()
```

```
plt.show()
```

```
Ujemajoči parametri:
  k = 12.25 ± 0.35
  A = 300.36 ± 8.90 nm
```



4 Dejanske meritve

[270]:
```
# Dolžina cevi v dm
L_dm = 11.5
abs_L_dm = 0.5  # napaka dolžine cevi v dm
rel_L_dm = abs_L_dm / L_dm  # relativna napaka dolžine cevi
```

[271]:
```
import pandas as pd

# Uvoz podatkov iz Excel datoteke 'meritve.xlsx'
df_meritve = pd.read_excel('meritve.xlsx')

# Prikaz prvih nekaj vrstic za pregled
df_meritve.head()
```

[271]:

	c	\alpha r	\alpha z
0	0.00	75	79
1	0.03	90	96

```
2  0.05       102       110
3  0.07       115       126
4  0.09       131       136
```

Napake meritev

[272]:
```
# Volumen tekočine v mL
V_mL = 3000
abs_V_mL = 60  # napaka volumna v mL
rel_V_mL = abs_V_mL / V_mL  # relativna napaka volumna
print(f"Relativna napaka volumna: {rel_V_mL:.2f} ({rel_V_mL*100:.1f}%)")
```

```
Relativna napaka volumna: 0.02 (2.0%)
```

[273]:
```
# napaka mase v g
abs_m0 = 0
abs_m = 3 # napaka mase v g
# Izračun abs_m / (c * V_mL) za vsako koncentracijo c v df_meritve (vključno z 0)
rel_m_error = []
for c in df_meritve['c']:
    if c != 0:
        value = abs_m / (c * V_mL)
    else:
        value = np.nan  # ali np.inf, če želiš označiti nedoločeno
    rel_m_error.append(value)

rel_m_error  # seznam rezultatov za vsako c
```

[273]:
```
[nan,
 0.03333333333333333,
 0.02,
 0.014285714285714284,
 0.011111111111111112,
 0.01]
```

Meritve z odšteto začetno vrednostno

[274]:
```
df_standard = df_meritve.copy().assign(
    **{
        '\\alpha r': df_meritve['\\alpha r'] - df_meritve['\\alpha r'].iloc[0],
        '\\alpha z': df_meritve['\\alpha z'] - df_meritve['\\alpha z'].iloc[0]
    }
)

df_standard
```

[274]:
```
      c  \alpha r  \alpha z
0  0.00         0         0
1  0.03        15        17
```

```
2  0.05        27        31
3  0.07        40        47
4  0.09        56        57
5  0.10        62        64
```

[275]:
```
abs_rot = 4
rel_rot_err_r = [
    abs_rot / row['\\alpha r']
    for idx, row in df_standard.iterrows()
    if idx > 0 and row['\\alpha r'] != 0
]
rel_rot_err_r
```

[275]:
```
[np.float64(0.26666666666666666),
 np.float64(0.14814814814814814),
 np.float64(0.1),
 np.float64(0.07142857142857142),
 np.float64(0.06451612903225806)]
```

[276]:
```
rel_rot_err_z = [
    abs_rot / row['\\alpha z']
    for idx, row in df_standard.iterrows()
    if idx > 0 and row['\\alpha z'] != 0
]
rel_rot_err_z
```

[276]:
```
[np.float64(0.23529411764705882),
 np.float64(0.12903225806451613),
 np.float64(0.0851063829787234),
 np.float64(0.07017543859649122),
 np.float64(0.0625)]
```

[ ]:
```

```

4.1 Izračun [α] za vsako meritve

$[\alpha] = \frac{\alpha}{c \cdot L}$

4.1.1 rdeča

[277]:
```
# Izračun seznama specifičnih rotacij za rdečo (drugi stolpec, vrstice > 0)
spec_rot_r = [
    row['\\alpha r'] / (row['c'] * L_dm)
    for idx, row in df_standard.iterrows()
    if idx > 0 and row['c'] != 0
]
print(spec_rot_r)
```

```
# Izračun relativnih napak za vsako koncentracijo (vrstice > 0 in c != 0)
# Izračun relativnih napak za vsako koncentracijo (vrstice > 0 in c != 0), ␣
 ↪vključno z rel_rot_err_r
rel_errors_r = [
    rel_L_dm + rel_m + rel_V_mL + rel_rot_err_r[idx - 1]
    for idx, rel_m in enumerate(rel_m_error)
    if idx > 0 and df_standard.loc[idx, 'c'] != 0
]
rel_errors_r
```

```
[np.float64(43.47826086956522), np.float64(46.95652173913043),
np.float64(49.689440993788814), np.float64(54.10628019323672),
np.float64(53.91304347826086)]
```

[277]:
```
[np.float64(0.3634782608695652),
 np.float64(0.23162640901771336),
 np.float64(0.1777639751552795),
 np.float64(0.14601794340924776),
 np.float64(0.13799438990182328)]
```

$[\alpha]_{c=0.030} = 43\,(1 \pm 0.36)$
$[\alpha]_{c=0.050} = 47\,(1 \pm 0.23)$
$[\alpha]_{c=0.070} = 50\,(1 \pm 0.17)$
$[\alpha]_{c=0.090} = 54\,(1 \pm 0.15)$
$[\alpha]_{c=0.100} = 54\,(1 \pm 0.14)$

4.2 zelena

[278]:
```
# Izračun seznama specifičnih rotacij za zeleno (tretji stolpec, vrstice > 0)
spec_rot_z = [
    row['\\alpha z'] / (row['c'] * L_dm)
    for idx, row in df_standard.iterrows()
    if idx > 0 and row['c'] != 0
]
print(spec_rot_z)

# Izračun relativnih napak za vsako koncentracijo (vrstice > 0 in c != 0)
rel_errors_z = [
    rel_L_dm + rel_m + rel_V_mL + rel_rot_err_z[idx - 1]
    for idx, rel_m in enumerate(rel_m_error)
    if idx > 0 and df_standard.loc[idx, 'c'] != 0
]
rel_errors_z
```

```
[np.float64(49.275362318840585), np.float64(53.91304347826086),
np.float64(58.38509316770186), np.float64(55.072463768115945),
np.float64(55.65217391304347)]
```

[278]:
```
[np.float64(0.33210571184995735),
 np.float64(0.21251051893408135),
 np.float64(0.1628703581340029),
 np.float64(0.14476481057716756),
 np.float64(0.13597826086956522)]
```

$[\alpha]_{c=0.030} = 49\,(1 \pm 0.33)$
$[\alpha]_{c=0.050} = 54\,(1 \pm 0.21)$
$[\alpha]_{c=0.070} = 58\,(1 \pm 0.16)$
$[\alpha]_{c=0.090} = 55\,(1 \pm 0.14)$
$[\alpha]_{c=0.100} = 56\,(1 \pm 0.14)$

4.3 Priprava za analizo z Drudejevo metodo

$$[\alpha](\lambda) = \frac{k\lambda^2}{\lambda^2 - A^2}$$

4.3.1 valovne dolžine laserjev

[279]:
```
lambda_r = 635
lambda_z = 532
```

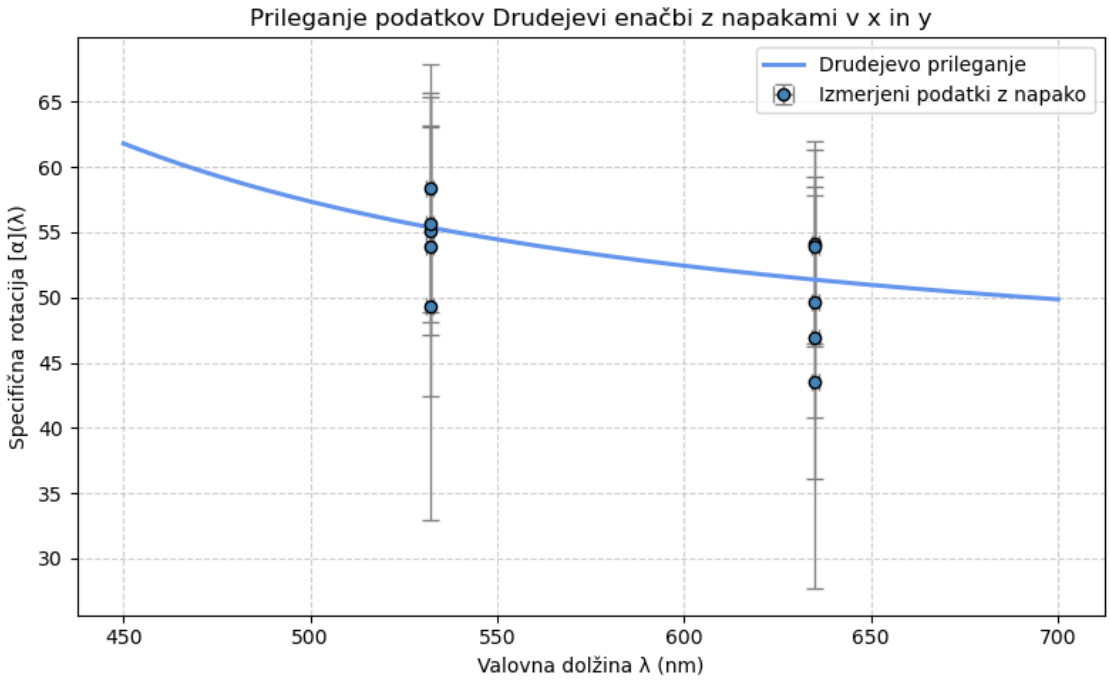
[280]:
```
# Tu vnesemo svoje podatke
lambda_vals = [635, 635, 635, 635, 635, 532, 532, 532, 532,532]  # valovne␣
↪dolžine v nm
lambda_errs = [1] * len(lambda_vals)  # predpostavljamo napako ±1 nm za vsako␣
↪valovno dolžino



alpha_vals = np.concatenate((spec_rot_r, spec_rot_z))  # specifična rotacija

alpha_errs_r = [val * rel for val, rel in zip(spec_rot_r, rel_errors_r)]
alpha_errs_z = [val * rel for val, rel in zip(spec_rot_z, rel_errors_z)]
alpha_errs = np.concatenate((alpha_errs_r, alpha_errs_z))  # napake specifične␣
↪rotacije
```

[281]:
```
# Uvozimo potrebne knjižnice
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Drudejeva funkcija
def drude(lambda_nm, k, A):
    return (k * lambda_nm**2) / (lambda_nm**2 - A**2)


# Prileganje z upoštevanjem napak v y (x napake ne vplivajo na fit)
```

```
params, cov = curve_fit(drude, lambda_vals, alpha_vals, sigma=alpha_errs,␣
 ↪absolute_sigma=True, p0=(1e4, 200))
k_fit, A_fit = params
k_err, A_err = np.sqrt(np.diag(cov))

# Izpis rezultatov
print("Ujemajoči parametri:")
print(f"  k = {k_fit:.2f} ± {k_err:.2f}")
print(f"  A = {A_fit:.2f} ± {A_err:.2f} nm")

# Priprava za prikaz prileganja
lambda_fit = np.linspace(450, 700, 300)
alpha_fit = drude(lambda_fit, k_fit, A_fit)

# Risanje grafa z error bar-i v obeh smereh
plt.figure(figsize=(8, 5))

plt.errorbar(
    lambda_vals,
    alpha_vals,
    xerr=lambda_errs,
    yerr=alpha_errs,
    fmt='o',
    markersize=6,
    markerfacecolor='steelblue',
    markeredgecolor='black',
    ecolor='gray',
    elinewidth=1,
    capsize=4,
    label='Izmerjeni podatki z napako'
)

# Prileganje funkcije
plt.plot(lambda_fit, alpha_fit, color='cornflowerblue', linewidth=2.2,␣
 ↪label='Drudejevo prileganje')

# Oznake in estetika
plt.xlabel('Valovna dolžina λ (nm)')
plt.ylabel('Specifična rotacija [α](λ)')
plt.title('Prileganje podatkov Drudejevi enačbi z napakami v x in y')
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.tight_layout()
plt.show()
```

```
Ujemajoči parametri:
  k = 43.88 ± 11.85
```

```
A = 242.28 ± 146.87 nm
```



[282]:
```
k_rel = k_err / k_fit
A_rel = A_err / A_fit

print(f"Relativna napaka k: {k_rel:.2%}")
print(f"Relativna napaka A: {A_rel:.2%}")
```

```
Relativna napaka k: 27.00%
Relativna napaka A: 60.62%
```

[283]:
```
from sympy import symbols, diff, simplify

# Define symbols
k, A, lam = symbols('k A lam')

# Define the function
alpha = k * lam**2 / (lam**2 - A**2)

# Partial derivatives
d_alpha_dk = simplify(diff(alpha, k))
d_alpha_dA = simplify(diff(alpha, A))
d_alpha_dlam = simplify(diff(alpha, lam))

print("[α]/k =", d_alpha_dk)
```

```
print("[α]/A =", d_alpha_dA)
print("[α]/λ =", d_alpha_dlam)
```

```
[α]/k = -lam**2/(A**2 - lam**2)
[α]/A = 2*A*k*lam**2/(A**2 - lam**2)**2
[α]/λ = -2*A**2*k*lam/(A**2 - lam**2)**2
```

Delni odvodi Drudejeve funkcije:

$\frac{\partial[\alpha]}{\partial k} = \frac{-\lambda^2}{A^2-\lambda^2}$

$\frac{\partial[\alpha]}{\partial A} = \frac{2Ak\lambda^2}{(A^2-\lambda^2)^2}$

[284]:
```
from sympy import lambdify

# Uporabi vrednosti: k_fit, k_err, A_fit, A_err, lambda_r ali lambda_z
# Izračunamo napako za rdečo valovno dolžino (lambda_r)
lam_val = lambda_r

# Pretvori odvode v funkcije
d_alpha_dk_func = lambdify((k, A, lam), d_alpha_dk)
d_alpha_dA_func = lambdify((k, A, lam), d_alpha_dA)

# Izračunaj odvode pri dobljenih parametrih
d_alpha_dk_val = d_alpha_dk_func(k_fit, A_fit, lam_val)
d_alpha_dA_val = d_alpha_dA_func(k_fit, A_fit, lam_val)

# Skupna napaka
delta_alpha = abs(d_alpha_dk_val) * k_err + abs(d_alpha_dA_val) * A_err
print(f"Δ[α]({lam_val} nm) = {delta_alpha:.2f}")
print(f"[α]({lam_val} nm) = {d_alpha_dk_func(k_fit, A_fit, lam_val) * k_fit +␣
 ↪d_alpha_dA_func(k_fit, A_fit, lam_val) * A_fit:.2f} ± {delta_alpha:.2f}")



# Uporabi vrednosti: k_fit, k_err, A_fit, A_err, lambda_r ali lambda_z
# Izračunamo napako za rdečo valovno dolžino (lambda_r)
lam_val = lambda_z

# Pretvori odvode v funkcije
d_alpha_dk_func = lambdify((k, A, lam), d_alpha_dk)
d_alpha_dA_func = lambdify((k, A, lam), d_alpha_dA)

# Izračunaj odvode pri dobljenih parametrih
d_alpha_dk_val = d_alpha_dk_func(k_fit, A_fit, lam_val)
d_alpha_dA_val = d_alpha_dA_func(k_fit, A_fit, lam_val)

# Skupna napaka
delta_alpha = abs(d_alpha_dk_val) * k_err + abs(d_alpha_dA_val) * A_err
```

```
print(f"Δ[α]({lam_val} nm) = {delta_alpha:.2f}")
print(f"[α]({lam_val} nm) = {d_alpha_dk_func(k_fit, A_fit, lam_val) * k_fit +␣
↪d_alpha_dA_func(k_fit, A_fit, lam_val) * A_fit:.2f} ± {delta_alpha:.2f}")
```

```
Δ[α](635 nm) = 24.48
[α](635 nm) = 68.86 ± 24.48
Δ[α](532 nm) = 32.51
[α](532 nm) = 84.34 ± 32.51
```

[285]:
```
from sympy import lambdify

# Izračun specifične rotacije in njene napake za modri laser (npr. λ = 450 nm)
lambda_blue = 450  # nm

# Uporabi Drudejev model in napake parametrov
alpha_blue = (k_fit * lambda_blue**2) / (lambda_blue**2 - A_fit**2)

# Delni odvodi po k in A (že definirani: d_alpha_dk, d_alpha_dA)

d_alpha_dk_func = lambdify((k, A, lam), d_alpha_dk)
d_alpha_dA_func = lambdify((k, A, lam), d_alpha_dA)

d_alpha_dk_val = d_alpha_dk_func(k_fit, A_fit, lambda_blue)
d_alpha_dA_val = d_alpha_dA_func(k_fit, A_fit, lambda_blue)

# Skupna napaka (približek, zanemarimo korelacijo)
delta_alpha_blue = abs(d_alpha_dk_val) * k_err + abs(d_alpha_dA_val) * A_err

print(f"[α](450 nm) = {alpha_blue:.2f} ± {delta_alpha_blue:.2f}")
```

```
[α](450 nm) = 61.80 ± 47.27
```