Python Übung 2

November 18, 2024

1 Aufgabe 1: Installation

Installiere die Module `numpy`, `matplotlib` und `scipy`. Für Pycharm ist der Prozess in der Installationsanleitung beschrieben.

2 Aufgabe 2: Numerische Integration

Das Integral über eine Gaussfunktion ist gleich der Quadratwurzel der Kreiszahl π.

$$\int_{-\infty}^{\infty} e^{-x^2} \, \mathrm{d}x = \sqrt{\pi}$$

Integriere die Gaussfunktion numerisch und vergleiche das Ergebnis mit der analytischen Lösung. Benutze dazu die Funktion `numpy.trapz()` (Dokumentation). Bei einer numerischen Integration kann das Integral über den Bereich $(-\infty, +\infty)$ durch ein Integral über ein endliches Intervall $[a, b]$ angenähert werden. Hilfreiche `numpy` Funktionen/Konstanten für diese Aufgabe: `numpy.pi`, `numpy.exp()`, `numpy.linspace()`, `numpy.sqrt()`

```
"""TODO: importiere das numpy modul
         mit alias np
"""
import

"""TODO: definiere eine Funktion, die für ein
         gegebenes x eine Gaussfunktion zurück
         gibt
"""
def gauss_function(x):
    return ____

"""TODO: erstelle einen 1D-Array"""
x = ___
y = gauss_function(x)

"""TODO: numerische Integration"""
numeric_integral = ____

"""TODO: analytische Lösung"""
correct = ____
```

```
print(f"Numerisches Integral: {numeric_integral}")
print(f"Analytische Lösung: {correct}")
print(f"Differenz: {numeric_integral - correct}")
```

2.1 Lösung

[89]:
```
"""TODO: importiere das numpy modul
         mit alias np
"""
import numpy as np

"""TODO: definiere eine Funktion, die für ein
         gegebenes x eine Gaussfunktion zurück
         gibt
"""
def gauss_function(x):
    return np.exp(-(x**2))

"""TODO: erstelle einen 1D-Array"""
x = np.linspace(-20, 20, 2000)
y = gauss_function(x)

"""TODO: numerische Integration"""
numeric_integral = np.trapz(y, x)

"""TODO: analytische Lösung"""
correct = np.sqrt(np.pi)

print(f"Numerisches Integral: {numeric_integral}")
print(f"Analytische Lösung: {correct}")
print(f"Differenz: {numeric_integral - correct}")
```

```
Numerisches Integral: 1.7724538509055159
Analytische Lösung: 1.7724538509055159
Differenz: 0.0
```

3 Aufgabe 3: Kalibriergerade

Für ein Photometer soll eine Kalibriergerade eines Farbstoffes bestimmt werden. Um die Kalibriergerade zu erhalten wurde bei jeder Konzentration eine Dreifachbestimmung durchgeführt. In der Textdatei `calibration.txt` ist in der ersten Spalte die Konzentration in mol/L aufgeführt, und in den Spalten danach die Absorbanz in der Dreifachbestimmung:

```
1.000   0.20131   0.19200   0.19871
1.071   0.23694   0.24483   0.24031
1.143   0.29436   0.28931   0.28332
1.214   0.32775   0.33309   0.33133
```

```
1.286   0.37221   0.36542   0.37171
1.357   0.41311   0.41872   0.42085
1.429   0.45936   0.45435   0.45590
1.500   0.50339   0.48983   0.50489
1.571   0.54461   0.53828   0.54187
1.643   0.57365   0.58020   0.58108
1.714   0.62700   0.62830   0.62064
1.786   0.67391   0.66989   0.66971
1.857   0.71650   0.71995   0.71171
1.929   0.76149   0.76255   0.76195
2.000   0.79969   0.80290   0.80503
```

3.1 Mittelwert der Dreifachbestimmung

Berechne für jede Konzentration den Mittelwert der Dreifachbestimmung. Der Mittelwert kann mit der numpy Funktion `mean` berechnet werden. Um gleichzeitig alle Mittelwerte auszurechnen kann das `axis` Argument verwendet werden. Die Dokumentation dazu lässt sich auf der `numpy.mean` Webseite finden.

```
"""TODO: numpy importieren"""
import ___ as __

"""TODO: laden der Textdatei """
raw_data = ________

"""TODO: erste Spalte aus raw_array extrahieren"""
concentration = raw_data[___, ___]

"""TODO: Absorbanz aus raw_array extrahieren"""
absorbance = raw_data[___, ___]

"""TODO: Mittelwert berechnen"""
abs_mean = np.mean(absorbance, axis=___)
```

3.2 Lineare Regression

Importiere das Modul `scipy`. Verwende die Funktion `scipy.stats.linregress` aus dem `scipy` Modul um die Lineare Regression mit den Konzentrationen und Mittelwerten der Absorbanz durchzuführen. Das Skript soll den Anstieg, Achsenabschnitt und das Bestimmtheitsmaß der Ausgleichsgerade ausgeben. Die Dokumentation zur Funktion befindet sich auf der Scipy Webseite.

3.3 Darstellung der Ausgleichsgeraden

Benuze das modul `matplotlib.pyplot` um die Mittelwerte der Absorbanz und die Ausgleichsgerade in einem Diagramm darzustellen.

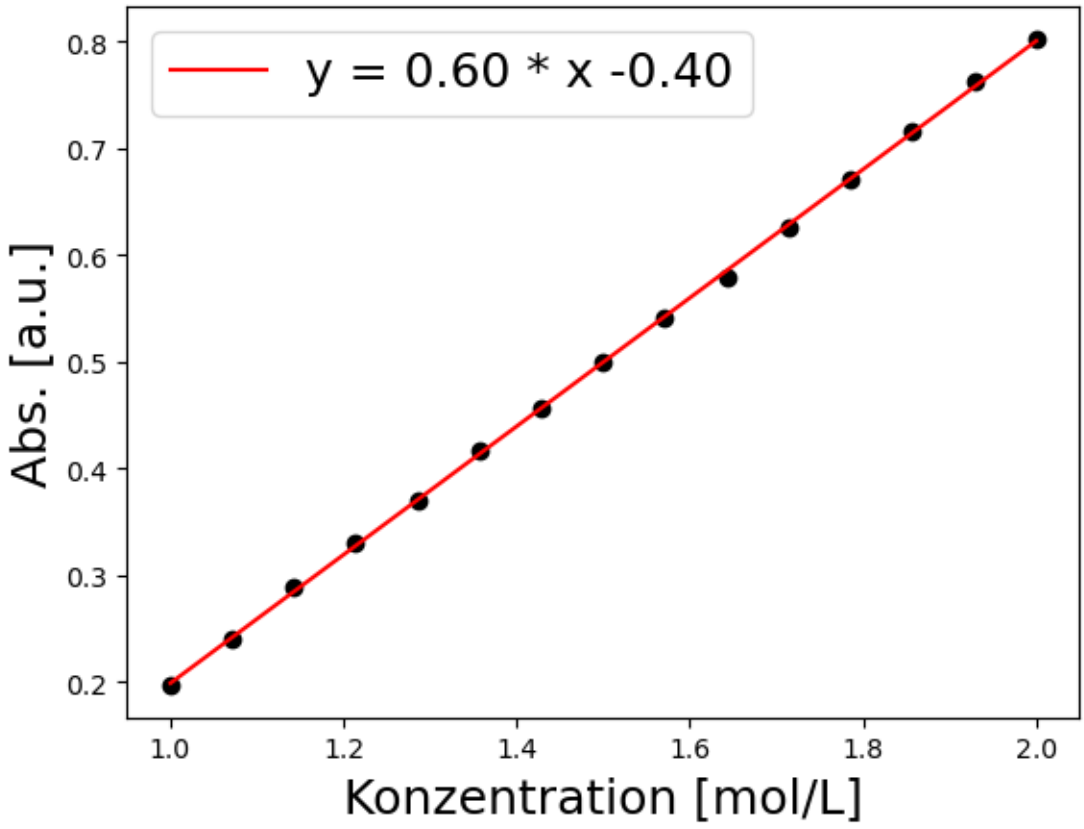
- importiere das `matplotlib.pyplot` modul mit dem alias `plt`
- nutze die Funktion `plt.plot()` um den Mittelwert der Absorbanz in abhängigkeit von der Konzentration zu plotten.

- nutze die Funktion `plt.plot()` um die Ausgleichsgerade darzustellen, dazu kann z.B. für x der Array `concentration` und für y ein neuer Array `slope * concentration + intercept` verwendet werden. `slope` und `intercept` werden aus der linearen regression erhalten
- erstelle eine Legende mit der Funktion `plt.legend()`
- Achsenbeschriftung mit `plt.xlabel()` und `plt.ylabel()`
- Darstellen des Diagramms mit `plt.show()`
- oder speichern mit `plt.savefig()`

3.4 Lösung

[45]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy

# Textdatei laden
raw_data = np.loadtxt("calibration.txt")

# Spalten separieren
concentration = raw_data[:, 0]
absorbance = raw_data[:, 1:]

# Mittelwert der Absorbanz
abs_mean = np.mean(absorbance, axis=1)

# Kalibriergerade ausrechnen
slope, intercept, r, p, se = scipy.stats.linregress(concentration, abs_mean)
print(f"Anstieg: {slope}\nAchsenabschnitt: {intercept}\nR^2: {r**2}")

# Diagramm erstellen
plt.plot(concentration, abs_mean, color='black', linestyle="", marker='o')
plt.plot(concentration, slope * concentration + intercept, color='red',␣
 ↪label=f"y = {slope:.2f} * x {intercept:.2f}")
plt.legend(fontsize=18)
plt.xlabel("Konzentration [mol/L]", fontsize=18)
plt.ylabel("Abs. [a.u.]", fontsize=18)
plt.savefig("calibration.png", dpi=200)
plt.show()
```

```
Anstieg: 0.6016021058759651
Achsenabschnitt: -0.40253849214728094
R^2: 0.999737167658455
```



4 Aufgabe 4: mehrere Spektren einlesen

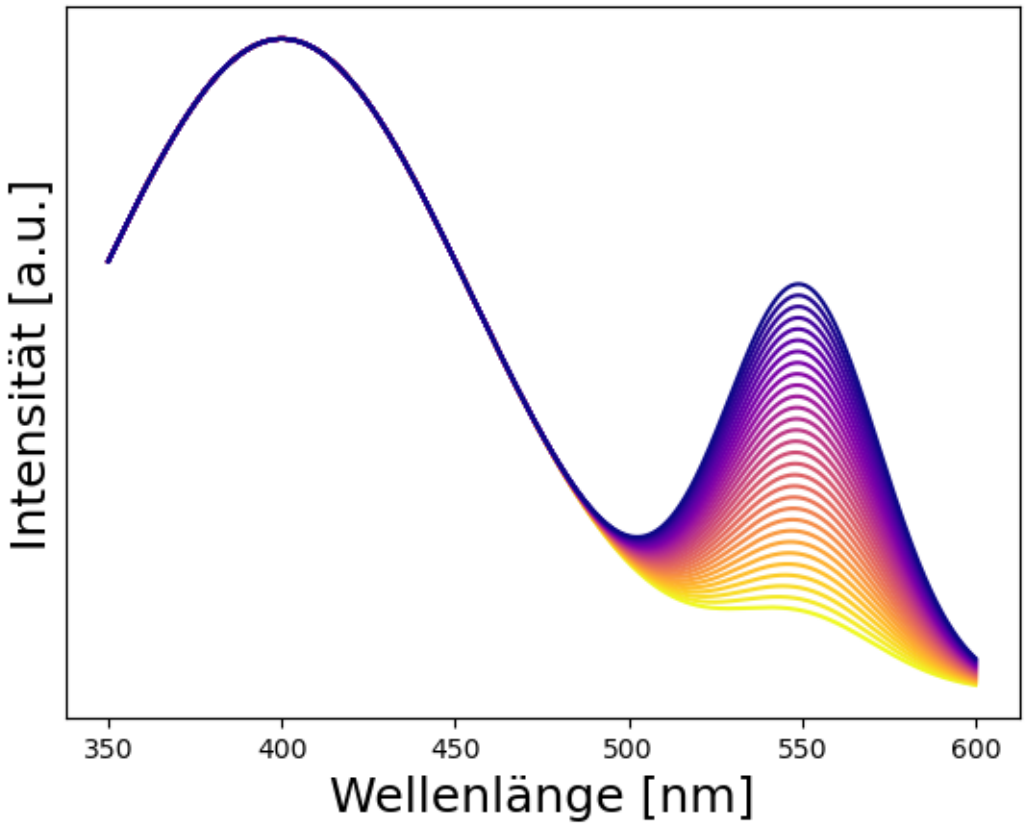
30 UV-Vis Spektren wurden aufgenommen. Stelle alle Spektren in einem Diagramm dar. Die Spektren sind in separaten Textdateien gespeichert (`spect_0.txt`, ..., `spect_29.txt`).

```
import numpy as np
import matplotlib.pyplot as plt

# Anzahl Spektren
n_spectra = 30

# erstellt eine Liste mit Farben
color_list = plt.cm.plasma(1-np.linspace(0, 1, n_spectra))


for i in range(30):
    """TODO: Filename mit f-string erstellen"""
    filename = ___
    """TODO: Spektrum laden und plotten"""
```

```
"""TODO: xlabel, ylabel definieren"""


"""TODO Bonus: Zahlen auf der y-Achse entfernen"""
```

4.1 Lösung

[88]:
```
import numpy as np
import matplotlib.pyplot as plt

# Anzahl Spektren
n_spectra = 30

# erstellt eine Liste mit Farben
color_list = plt.cm.plasma(1-np.linspace(0, 1, n_spectra))


for i in range(30):
    """TODO: Filename mit f-string erstellen"""
    filename = f"spect_{i}.txt"
    """TODO: Spektrum laden und plotten"""
    raw_data = np.loadtxt(filename)
    x = raw_data[:, 0]
    y = raw_data[:, 1]
    plt.plot(x, y, color=colors[i])

"""TODO: xlabel, ylabel definieren"""
plt.xlabel("Wellenlänge [nm]", fontsize=18)
plt.ylabel("Intensität [a.u.]", fontsize=18)

"""TODO Bonus: Zahlen auf der y-Achse entfernen"""
plt.yticks([])
plt.show()
```


Intensität [a.u.]
350
400
450
500
550
600
Wellenlänge [nm]