

Aufgabenblatt 1 (Praxis)

wr@cg.tu-berlin.de

WiSe 2024/2025

Allgemeine Hinweise:

- Die Aufgaben sind von jeder/m Studierenden *einzel*n zu bearbeiten und abzugeben (Plagiate werden entsprechend der Studienordnung geahndet).
- Verwenden Sie die vorgegebene Code-Basis. **Sie dürfen keine weiteren Module importieren.** Die zu implementierenden Funktionen befinden sich in der Datei `main.py`. Ihr Code ist an den mit `# TODO: ...` gekennzeichneten Stellen einzufügen. Die NumPy-Funktionen, welche Sie zur Lösung einer Aufgabe nicht verwenden dürfen, sind unter `Forbidden` in der Docstring Beschreibung der entsprechenden Funktion aufgelistet.
- Wir stellen einige rudimentäre Unit-Tests zur Verfügung, welche Sie verwenden sollen, um die Funktionalität ihres Codes zu testen. Sie sollten diese Tests während der Implementierung Ihrer Lösung vervollständigen (Funktionalität beschrieben in Python `unittest`). Sie können die Tests mit dem Aufruf `python3 tests.py -v [Tests.test_<function>]` ausführen.
- Bitte reichen Sie die Datei `main.py` mit Ihren Lösungen bis **Donnerstag, den 14.11.2024** auf <https://autolab.cg.tu-berlin.de> mit ihren Zugangsdaten ein. Ein mehrfacher Upload bis zum Abgabende ist möglich. Die letzte Version wird bewertet.

Aufgabe 1: Effizienz von Berechnungen in NumPy (2 Punkte)

Das Ziel dieser Aufgabe ist es, die Performance von NumPy mit der einer einfachen Python Implementierung zu vergleichen. Die Multiplikation zweier Matrizen $A \in \mathbb{R}^{n \times m}$, mit Elementen A_{ij} , und $B \in \mathbb{R}^{m \times p}$, mit Elementen B_{ij} , ist definiert als

$$(AB)_{ij} = \sum_{k=0}^{m-1} A_{ik} B_{kj} \in \mathbb{R}^{n \times p}, \quad (1)$$

d.h. das (i, j) -te Element des Produkts AB ist das Skalarprodukt der i -ten Zeile von A mit der j -ten Spalte von B . Berechnen Sie ggf. das Matrixprodukt von zwei 2×2 Matrizen per Hand, um den Algorithmus besser zu verstehen.

Aufgabe 1.1: Matrixmultiplikation (1.5 Punkte)

Implementieren Sie die Funktion `matrix_multiplication`, welche das Matrixprodukt zweier beliebiger, kompatibler Matrizen mit Hilfe der obigen Gleichung berechnet. Sie dürfen keine Funktion von NumPy dafür benutzen. Ausschließlich Speicherzugriffe auf **einzelne** Elemente von `numpy.array` sind erlaubt. Die Funktion soll einen `ValueError` erzeugen, falls die Größen der gegebenen Matrizen nicht kompatibel sind.

Aufgabe 1.2: Vergleich mit NumPy (0.5 Punkte)

Vervollständigen Sie die Funktion `compare_multiplication`, welche die Matrixmultiplikation für verschiedene Matrixgrößen sowohl mit NumPy als auch mit Ihrer Funktion `matrix_multiplication()` berechnet, um die Laufzeit der Implementierungen zu vergleichen. Die gemessenen Berechnungszeiten werden graphisch dargestellt.

Aufgabe 2: Gleitkommazahlen (1 Punkt)

Implementieren Sie die Funktion `machine_epsilon`, welche die Maschinengenauigkeit für das im Parameter `fp_format` übergebene NumPy Gleitkommazahl-Format (z.B. `float32`) bestimmt. Verwenden Sie hierfür eine der in der Vorlesung besprochenen Definitionen der Maschinengenauigkeit.

Aufgabe 3: Matrixvergleich (1 Punkt)

Implementieren Sie die Funktion `close`, welche `True` zurück gibt, wenn zwei Matrizen innerhalb der, als Parameter `eps`, übergebenen Toleranz elementweise gleich sind. Die Funktion soll einen `ValueError` erzeugen, falls die Größen der gegebenen Matrizen nicht gleich sind.

Aufgabe 4: Rotationen in \mathbb{R}^2 (1 Punkt)

In dieser Aufgabe wird die Bedeutung und praktische Relevanz von orthogonalen Matrizen betrachtet. Eine Rotation um den Winkel θ in der Ebene \mathbb{R}^2 ist als Matrix durch

$$R(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \quad (2)$$

gegeben.

Aufgabe 4.1: Rotationsmatrix aufstellen (0.5 Punkte)

Implementieren Sie die Funktion `rotation_matrix()`, welche θ als Winkel in Grad übergeben bekommt und die entsprechende Rotationsmatrix zurück gibt.

Aufgabe 4.2: Rotationsmatrix invertieren (0.5 Punkte)

Implementieren Sie die Funktion `inverse_rotation()`, welche die Inverse der Rotationsmatrix zum Winkel θ zurückgibt. Dabei dürfen Sie keine NumPy Funktionen verwenden.