Wissenschaftliches Rechnen

Aufgabenblatt 1 (Praxis)

wr@cg.tu-berlin.de

WiSe 2019/2020

Allgemeine Hinweise:

- Die Aufgaben sind von jeder/m Studierenden *einzeln* zu bearbeiten und abzugeben (Plagiate werden entsprechend der Studienordnung geahndet).
- Verwenden Sie die vorgegebene Code-Basis. 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 **Montag, den 18.11.2019, um 14:00 Uhr** auf `https://autolab.service.tu-berlin.de` mit ihren Zugangsdaten ein. Ein mehrfacher Upload bis zum Abgabeende ist möglich. Die letzte Version wird bewertet.

Aufgabe 1: Effizienz von Berechnungen in NumPy (4 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

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

d.h. das (i,j)-te Element des Produkts $A\,B$ 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 (3 Punkte)

Implementieren Sie die Funktion `matrix_multiplication()`, welche das Matrixprodukt zweier beliebiger, kompatibler Matrizen mit Hilfe der obigen Gleichung berechnet. Die Funktion soll einen `ValueError` erzeugen, falls die Größen der gegebenen Matrizen nicht kompatibel sind.

Aufgabe 1.2: Vergleich mit NumPy (1 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 (3 Punkte)

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: Rotationen in $\mathbb{R}^2$ (3 Punkte)

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

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

gegeben.

Aufgabe 3.1: Rotationsmatrix aufstellen (1 Punkt)

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

Aufgabe 3.2: Rotationsmatrix invertieren (2 Punkte)

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