Methodische und Praktische Grundlagen der Informatik 4

Aufgabenblatt 0

mpgi4@cg.tu-berlin.de

WiSe 2014/2015

Allgemeine Hinweise:

- Die Aufgaben sollen in Gruppen bestehend aus zwei bis drei Personen bearbeitet werden. Ausnahmen müssen mit dem jeweiligen Tutor abgesprochen werden.
- Bitte reichen Sie Ihre Lösungen in Form einer ZIP Datei bis **Donnerstag, den 30.10.2014, um 12:00 Uhr** auf der ISIS Webseite der Vorlesung ein. Tragen Sie in jede abgegebene Datei Name und Email-Adresse aller Gruppenmitglieder ein.
- Wenn eine Aufgabe die Abgabe einer Grafik verlangt, dann muss ein vollständig funktionsfähiges Programm in der Lösung enthalten sein, welches bei der Ausführungen die Grafik erstellt.
- Verwenden Sie den vorgegebenen Skelettcode, um die Aufgaben umzusetzen.

Aufgabe 1: Effizienz von Berechnungen in NumPy

Das Ziel dieser Aufgabe ist es, die Performance von Numpy mit der einer naiven 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. (Falls Ihnen die Bedeutung der Formel nicht klar ist, dann sollten sie das Produkt von 2×2 oder 3×3 Matrizen mit der Hand berechnen.)

a.) Implementieren Sie die Funktion `matMult(A, B)`, welche das Matrixprodukt zweier beliebiger, kompatibler NumPy Matrizen (`np.array`) mit Hilfe von Gleichung 1 berechnet. Die Funktion soll eine ValueException ausgeben, falls die Größen der gegebenen Matrizen nicht kompatibel sind.

b.) Benutzen Sie die Funktion `matMultExperiment()`, welche die Matrixmultiplikation für verschiedene Matrixgrößen sowohl mit NumPy als auch mit Ihrer Funktion `matMult(A, B)` berechnet, um die Laufzeit der Implementierungen zu vergleichen. Die gemessenen Berechnungszeiten werden mit MatPlotLib graphisch dargestellt. Erläutern Sie kurz das in der Grafik dargestellte Ergebnis in der Textdatei `matmult_performance.txt`.

Aufgabe 2: Rotationen in $\mathbb{R}^2$

In dieser Aufgaben 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.

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

b.) Überprüfen Sie in `testOrthogonality()` numerisch, ob $R^{-1} = R^T$ für $\theta \in \{30°, 45°, 90°, 120°\}$, d.h. ob $R(\theta)$ eine orthogonale Matrix ist. Berechnen Sie dazu den durchschnittlichen Fehler. Was bedeutet das Ergebnis?

c.) Ein Vektor auf der X-Achse ($\vec{v} = (1, 0)$) soll zunächst in $1°$ Schritten um $37°$ entgegen dem Uhrzeigersinn gedreht werden und anschließend wieder in $1°$ Schritten zurück in seine ursprüngliche Position. Dies soll mit lediglich einem Aufruf zu `rotMatrix(`θ`)` realisiert werden. Nach jedem $1°$ Schritt soll der Vektor mit Hilfe der Funktion `plotVector()` graphisch dargestellt werden.