Prof. Dr. Reinhard Klein
Elena Trunz

Universität Bonn
Institut für Informatik II
06.11.2020

Wintersemester 2020/2021

Übungen zu Angewandte Mathematik: Numerik - Blatt 1

Auf diesem Übungsblatt geht es darum eine Programmiersprache zu erlernen mit der die Programmieraufgaben bearbeitet werden können. Die Abgabe der Lösungen zur Aufgabe 2 erfolgt per eCampus bis zum 13.11.2020, 10:15 Uhr. Bitte die Gruppennummer mit angeben.

Die Übungsblätter werden regelmäßig Programmieraufgaben enthalten, bei denen die in der Vorlesung erlernten Verfahren implementiert und auf praktische Problemstellungen angewandt werden. Diese Aufgaben sollten in der Programmiersprache Python bearbeitet werden. Python ist eine Mehrzweckprogrammiersprache und die Software, die man zur Verwendung benötigt ist Quelloffen und somit für jeden frei Verfügbar. Durch die, ebenfalls Quelloffenen Bibliotheken NumPy, SciPy und matplotlib, erhält Python ähnliche Funktionalitäten wie Matlab und ist somit gut für numerische Berechnungen und Visualisierungen geeignet.

Aufgabe 1 (Python installieren und lernen, *0 Punkte*)

Um numerische Berechnungen mit Python komfortabel durchzuführen, benötigt man die folgende Software:

- Python Interpreter: Dieser ist die Grundlage um in Python geschriebene Programme auszuführen und kann von python.org heruntergeladen werden (oder man benutzt Miniconda wie unten beschrieben). Bei der Auswahl der Version ist zu berücksichtigen, dass NumPy, SciPy und matplotlib nicht unbedingt für jede Version zur Verfügung stehen. Zum Testen der Frameworks wird Python 3.4 verwendet. Andere Python 3.x Versionen sollten ebenfalls problemlos funktionieren. Bei Python 2.7 oder älter kann es eventuell zu Kompatibilitätsproblemen kommen.
- Entwicklungsumgebung: Eine gute Entwicklungsumgebung macht das Programmieren und Debuggen deutlich angenehmer. In Frage kommen zum Beispiel PyDev für Eclipse, Python Tools for Visual Studio (Visual Studio ist für Studenten kostenlos über Dreamspark verfügbar), Spyder oder Pyzo.
- Softwarebibliotheken für Python:
  - NumPy: Stellt grundlegende Funktionalität zum Umgang mit Arrays und Matrizen zur Verfügung. Verfügbar unter NumPy.org.
  - SciPy: Stellt komplexere mathematische Funktionen zur Verfügung. Verfügbar unter SciPy.org.
  - matplotlib: Stellt Funktionen zum erstellen von Plots zur Verfügung. Verfügbar unter matplotlib.org.

Eine ausführlichere Übersicht mit nützlichen Links findest du unter scipy.org/getting-started.html. Unter scipy.org/install.html sind außerdem fertig geschnürte Pakete verlinkt, die alle oben genannten Komponenten enthalten. Empfehlenswert ist Miniconda, ein Paketverwaltungssystem für Python. Mit Miniconda können die benötigten Bibliotheken durch die folgende Befehlszeile installiert werden:

```
conda install numpy scipy matplotlib
```

Installiere dir nun die oben genannte Software.

Falls du Python noch nicht kennst, mache dich damit vertraut indem du das Tutorial auf docs.python .org/3/tutorial/ liest. Mache dich mit NumPy anhand der folgenden Kapitel in der offiziellen Hilfe vertraut:

- Array creation: docs.scipy.org/doc/numpy/user/basics.creation.html
- Indexing: docs.scipy.org/doc/numpy/user/basics.indexing.html
- Matrix objects: docs.scipy.org/doc/numpy/reference/arrays.classes.html#matrix-objects
- Array manipulation routines: docs.scipy.org/doc/numpy/reference/routines.array-manipulation.html
- Random sampling: docs.scipy.org/doc/numpy/reference/routines.random.html

Aufgabe 2 (Python üben, $16 \times 0,5 + 3 + 3 = 14$ *Punkte*)

Erstelle in Python ein Skript, welches für jede der nachfolgenden Aufgaben ein Beispiel mit Zahlen enthält. Dabei soll für jede Teilaufgabe jeweils eine Ausgabe (print) erfolgen, die folgende Form hat: Zuerst kommt der Buchstabe der Teilaufgabe, danach kommt ggf. die Matrix (oder Vektor) an der die Operationen ausgeführt werden sollen und dann kommt das Ergebnis (die Ergebnisse) der geforderten Operation(en).

a) Erstelle einen Kommentar.
b) Erstelle einen 1×4 Zeilenvektor.
c) Erstelle einen 5×1 Spaltenvektor.
d) Erstelle eine Nullmatrix mit einer vorhandenen Funktion.
e) Gebe die zweite Zeile einer 4×3 Matrix aus.
f) Gebe die dritte Spalte einer 4×4 Matrix aus.
g) Transponiere eine Matrix.
h) Erzeuge zwei Matrizen gleicher Größe ($m \times m$). Multipliziere die Matrizen einmal elementweise und einmal als gewöhnliche Matrixmultiplikation.
i) Verknüpfe zwei Matrizen jeweils unter Verwendung einer vorhandenen Funktion sowohl horizontal als auch vertikal.
j) Gebe die Größe bzw. Dimension einer Matrix aus.
k) Verändere unter Verwendung einer vorhandenen Funktion die Struktur einer 8×7 Matrix, so dass eine 14×4 Matrix entsteht.
l) Vervielfache einen 3×1 Vektor, so dass eine 3×10000 Matrix entsteht.
m) Setze alle negativen Elemente einer Matrix auf 0.
n) Erzeuge einen Vektor mit den Zahlen von 1 - 100, wobei die Zahlen einen Abstand von 7 aufweisen.
o) Erzeuge einen Vektor mit 100 Einträgen. Setze anschließend jedes zweite Element des Vektors auf null.
p) Erzeuge einen Vektor mit 100 Einträgen. Lösche anschließend jedes zweite Element des Vektors.
q) Erzeuge zwei Matrizen mit der Dimension 100×3, welche mit Zufallszahlen gefüllt sind. Dabei können die Zeilen einer solchen Matrix als 1×3 Vektoren interpretiert werden. Berechne das Skalarprodukt für alle Paare dieser Vektoren unter Verwendung von Schleifen. Dazu muss über die Zeilen der 1000×3 Matrizen iteriert werden und das Skalarprodukt für die Vektoren, welche durch die aktuellen Matrixzeilen gegeben sind, berechnet werden. Berechne weiterhin diese Skalarprodukte ohne die Verwendung von Schleifen. Vergleiche die Laufzeiten der beiden Varianten. (Hinweis: Zum Zeitmessen können in Matlab die Funktionen `tic`, `toc` und in Python das Modul `time` verwendet werden). Was fällt dir auf, wenn du die Laufzeiten miteinander vergleichst?

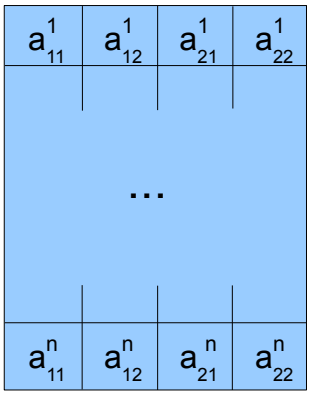

Abbildung 1: Speicherlayout der 2×2 Matrizen.

r) In dieser Aufgabe ist folgendes Szenario gegeben. Wir wollen 1000 2×2 Matrizen invertieren. Die 2×2 Matrizen sind jeweils als Zeile repräsentiert (vgl. Abbildung 1). Die tiefgestellten Zahlen an den Matrixeinträgen stellen die Position des Eintrages in der 2×2 Matrix dar. Die hochgestellte Zahl gibt den Index der aktuellen Matrix an ($n = 1000$). Erzeuge eine 1000×4 Matrix mit zufälligen Einträgen, wobei jede Zeile wie in Abbildung 1 gezeigt, nun einer 2×2 Matrix entspricht. Zur Berechnung der Inversen der Matrizen soll das gegebene Speicherlayout der 2×2 Matrizen **nicht** verändert werden (d.h. eine Zeile des gegebenen Speicherlayouts soll **nicht** in ein 2×2 Array umgewandelt werden, womit dann die Inverse über vorhandene Funktionen berechnet wird). Mit Hilfe der Cramer'schen Regel für 2×2 Matrizen kann jede der 1000 Matrizen nun invertiert werden. Berechne nun die Inversen für die erzeugten 2×2 Matrizen ohne dafür Schleifen zu verwenden.