Heinrich Heine Universität Düsseldorf

Mathematisches Institut
Prof. Dr. Achim Schädle
Dr. Marina Fischer

21.03.2023

Schriftliche Prüfung zur Vorlesung Computergestützte Mathematik zur Linearen Algebra

Bitte folgende Angaben ergänzen und **DEUTLICH LESBAR** in Druckbuchstaben schreiben:

Name: ..... Vorname: .....

Matrikel-Nr.: ..... Studienfach: .....

Hiermit bestätige ich, dass ich zu dieser schriftlichen Prüfung zugelassen bin, da ich

☐ die Zulassung im WS 2022/23 erworben habe,

☐ an einer schriftlichen Prüfung zur Vorlesung Computergestützte Mathematik zur Linearen Algebra bei
..... im WS/SS ..... teilgenommen, aber nicht bestanden habe,

☐ die Zulassung zur Prüfung im WS/SS ..... erworben habe.

.....
Unterschrift

Hinweise:

- In Ihrem Home-Verzeichnis finden die Dateien `WerBinIch.txt`, sowie `aufgabe1.py`, `aufgabe2.py`, `aufgabe3.py`, `aufgabe4.py` `aufgabe5.py`, `aufgabe6.py`, bzw. `aufgabe1.ipynb`, `aufgabe2.ipynb`, `aufgabe3.ipynb`, `aufgabe4.ipynb`, `aufgabe5.ipynb` und `aufgabe6.ipynb`. Verwenden Sie bitte **entweder** die .py Dateien mit `spyder` **oder** die .ipynb Dateien in `jupyter` für Ihre Lösungen.
- Ergänzen Sie zuerst die Datei `WerBinIch.txt` mit Ihrem Namen, Vorname, usw.
- Es werden nur Lösungsvorschläge gewertet, die in Dateien mit jeweils dem zur Aufgabe passenden Namen in Ihrem Home-Verzeichnis gespeichert sind. Speichern Sie in kurzen Abständen Ihre Lösungen, um ggf. den Verlust von Daten zu vermeiden, falls `jupyter` oder `spyder` einmal abstürzt.
- Nach der Klausur führen wir einen Restart des Kernels durch. Achten Sie daher darauf, dass die Option `Restart & Run all` unter `Kernel` in `jupyter` oder `Run file (F5)` in `spyder` ohne Fehlermeldung durchläuft.
- Es werden nur Lösungsvorschläge gewertet, bei denen der Lösungsweg klar zu erkennen ist.

Aufgabe	1	2	3	4	5	6	$\sum$
max. Punkte	6	6	6	6	6	6	36
err. Punkte							

Aufgabe 1: (3 + 3 Punkte)

Für ein $a \in \mathbb{R}, a > 0$, ist $f_a : \mathbb{R} \to \mathbb{R}$ stückweise definiert durch

$$f_a(x) = \begin{cases} a(x+1), & \text{falls } x \leq -1, \\ (1 - |x|^a)^{\frac{1}{a}}, & \text{falls } -1 < x < 1, \\ -a(x-1), & \text{falls } x \geq 1. \end{cases}$$

(a) Schreiben Sie in die Datei `aufgabe1.py` (oder `aufgabe1.ipynb`) eine Funktion `myfun(x,a)`, die zu einem Parameter `a` $\in \mathbb{R}^+$ und einem Array `x` mit n Einträgen die Funktion auswertet und einen Array mit den Funktionswerten $f_{\texttt{a}}(\texttt{x})$ zurückgibt.

(b) Ergänzen Sie `aufgabe1.py/aufgabe1.ipynb` so, dass für $a = \frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}$ die Funktionen f_a in einem gemeinsamen Fenster auf dem Intervall $[-2, 2]$ gezeichnet werden. Ergänzen Sie das Skript so, dass eine Legende, die die einzelnen Funktionen beschreibt, in Ihrer Graphik angezeigt wird. Platzieren Sie die Legende in der rechten unteren Ecke des Bildes.

Aufgabe 2: (3 + 3 Punkte)

Der unten als Pseudocode angegebene erweiterte euklidsche Algorithmus berechnet zu zwei natürlichen Zahlen a und b, den größten gemeinsamen Teiler $\mathrm{ggT}(a, b)$ und zusätzlich ganze Zahlen s und t, sodass

$$\mathrm{ggT}(a, b) = sa + tb \tag{1}$$

gilt.

In der Datei `aufgabe2.py/aufgabe2.ipynb` ist die Division mit Rest bereits als `rdiv` implementiert. Mit `q,r = rdiv(x,y)` berechnet man q und r so, dass $x = qy + r$ und $r < y$.

(a) Implementieren Sie in `aufgabe2.py/aufgabe2.ipynb` den erweiterten euklidschen Algorithmus als Funktion `eggT(a,b)`. Neben dem ggT von a und b werden damit auch s und t berechnet.

(b) Testen Sie Ihre Implementierung für $(a, b) \in \{(4, 0), (12, 36), (545, 985)\}$, indem Sie erstens mit Hilfe von `numpy.gcd` überprüfen, ob der ggT richtig berechnet wird, und zweitens die Gleichung (1) überprüfen.

Pseudocode:

`eggT`$(a,\ b)$	*Berechnet rekursiv* $\mathrm{ggT}(a,\ b),\ s,\ t$
if $b = 0$	
return $a,\ 1,\ 0$	$a = ggT(a,\ b),\ s = 1,\ t = 0$
$q,\ r \leftarrow$ `rdiv` (a, b)	*Division mit Rest* $a = bq + r$ *für* $r < b$
$a,\ s,\ t \leftarrow$ `eggT`$(b,\ r)$	*rekursiver Aufruf*
return $a,\ t,\ s - qt$	*Rückgabe von* $\mathrm{ggT}(a,\ b),\ s,\ t$

Aufgabe 3: (4 + 2 Punkte)

Sei $A = (a_{i,j})_{i,j=0}^{n-1} \in \mathbb{R}^{n\times n}$, $b = (b_i)_{i=0}^{n-1} \in \mathbb{R}^n$. Die Matrix D sei der Diagonalanteil von A, d.h. D hat die Einträge $d_{ii} = a_{ii}$ und für $i \neq j$ ist $d_{ij} = 0$. Weiter ist $N = A - D$.

Implementieren Sie in der Datei `aufgabe3.py/aufgabe3.ipynb` folgendes iteratives Verfahren zur Lösung des linearen Gleichungssystems $Ax = b$.

Für den Nullvektor $x^{(0)}$ sind $x^{(k)}$ rekursiv durch

$$x^{(k+1)} = D^{-1}(b - Nx^{(k)})$$

definiert.

(a) Implementieren Sie eine Funktion `verf(A,b,k)`, die für ein $n \times n$-Array `A`, ein n-Array `b` und eine natürliche Zahl `k`, $x^{(k)}$ berechnet. Hierbei darf bei der Anwendung von D^{-1} kein allgemeines Verfahren zur Lösung linearer Gleichungssysteme wie `inv`, `solve`, `linsolve` oder eine Matrixzerlegung aus `scipy` oder `numpy` verwendet werden.

(b) Wenden Sie `verf` auf die in `aufgabe3.py/aufgabe3.ipynb` als `numpy`-Array definierte Matrix A und den Vektor b an und geben Sie für $k = 1, 2, 3, 4$

$$\|Ax^{(k)} - b\|_2$$

mit dem `print` Befehl aus.

Hinweis: Sie können den `numpy` Befehl `diag` verwenden. `diag(A)` gibt die Hauptdiagonale von `A` als flaches Array zurück. Für ein flaches Array `d` erzeugt `diag(d)` die Diagonalmatrix mit den Einträgen von `d` auf der Hauptdiagonalen.

Aufgabe 4: (2 + 2 + 2 Punkte)

Für eine Liste $c = [c_0, \ldots, c_{n-1}]$ mit n paarweise verschiedenen Einträgen sind für $i = 0, \ldots, n-1$ Polynome l_i definiert:

$$l_i(x) := \prod_{j=0, j\neq i}^{n-1} \left(\frac{j+1}{2j+1}x - c_j\right).$$

Die Einträge der $n \times n$ Matrix A und des Vektors $b \in \mathbb{R}^n$ sind

$$a_{i,j} = l_i(c_j) \quad \text{und} \quad b_i = \int_{-1}^{1} l_i(x)dx.$$

(a) Implementieren Sie eine Funktion `l(i,c)`, die mit Hilfe der Polynomklasse `Polynom` für ein $i \in \{0, \ldots, n-1\}$ und eine Liste `c` das Polynom l_i, berechnet und zurückgibt.

(b) Implementieren Sie eine Funktion `solveAb(c)`, die mit Hilfe der Funktion `l(i,c)` aus (a) für die Liste `c`, die Matrix A und den Vektor b aufstellt, das lineare Gleichungssystem $Ax = b$ löst und die Lösung x und die Matrix A zurückgibt.

(c) Setzen Sie $n = 4$ und erzeugen Sie die Liste $c = [c_0, \ldots, c_{n-1}]$ mit $c_i = -\cos(\pi\frac{i}{n-1})$ und geben Sie hierfür das Ergebnis von `solveAb(c)` mit dem `print` Befehl aus.

Hinweis: Sollten Sie Teil (a) nicht hinbekommen, können Sie in (b) und (c) die in `aufgabe4.py/aufgabe4.ipynb` implementierte Funktion `k(i,c)` statt `l(i,c)` verwenden.

Aufgabe 5: (2 + 2 + 2 Punkte)

In der Datei `aufgabe5.py`/`aufgabe5.ipynb` sind numpy-Arrays `t` und `y` mit sechs Einträgen vorgegeben. Es wird angenommen, dass in den Parametern $x = (x_0, x_1, x_2)$ folgender linearer Zusammenhang besteht

$$y = f(t, x) = x_0 \sin(t) + x_1 \cos(t) + x_2 \exp(t).$$

(a) Ergänzen Sie `aufgabe5.py`/`aufgabe5.ipynb` um eine Funktion `f(t,x)`, die für ein `numpy` Array `t` und ein Array `x` mit drei Einträgen ein Array mit den Werten von f zurückgibt.

(b) Berechnen Sie $x = (x_0, x_1, x_2)$ so, dass f die vorgegebenen Daten `(t, y)` im Sinne der kleinsten Fehlerquadrate bestmöglich approximiert, das heißt $\sum_{j=0}^{5} |f(t_j, x) - y_j|^2$ soll minimal werden. Lösen Sie dazu das kleinste Fehlerquadrate Problem für die Koeffizienten x_i.

(c) Ergänzen Sie `aufgabe5.py`/`aufgabe5.ipynb` so, dass für das in (b) berechnete x

i) die Datenpunkte (t_j, y_j) mit dem Symbol '`+`',
ii) die Punkte $(t_j, f(t_j, x))$ mit dem Symbol '`o`' und
iii) die Funktion f als Linie über dem Intervall $[-2, 5]$ gezeichnet werden.

Aufgabe 6: (2 + 2 + 2 Punkte)

In der Datei `aufgabe6.py`/`aufgabe6.ipynb` finden Sie eine Funktion `sincos(x, y)`, die zu einem Vektor (x, y) die Werte $s = \sin(\alpha)$ und $c = \cos(\alpha)$ berechnet, so, dass dieser durch eine Drehung um den Winkel α auf ein Vielfaches des ersten kanonischen Basisvektors abgebildet wird, d.h.

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \pm\sqrt{x^2 + y^2} \\ 0 \end{pmatrix}.$$

Im $\mathbb{R}^n$ ist eine Rotation um den Winkel α in der k, l-ten Koordinatenebene durch die Matrix $G_{(k,l,c,s)} = (g_{i,j})_{i,j=0}^{n-1}$ gegeben. Die Einträge von G, die ungleich Null sind, sind

$$g_{k,k} = c = g_{l,l}, \quad g_{k,l} = s, \quad g_{l,k} = -s, \quad g_{m,m} = 1 \text{ für } m \notin \{k, l\},$$

wobei $c = \cos(\alpha)$ und $s = \sin(\alpha)$. Für $n > l > k \geq 0$ hat G als Matrix folgende Gestalt:

$$G_{(k,l,c,s)} = \begin{pmatrix} I_k & & & & \\ & c & & s & \\ & & I_{l-k-1} & & \\ & -s & & c & \\ & & & & I_{n-1-l} \end{pmatrix} \quad \text{mit} \quad I_m = m \times m \text{ Identität}$$

Berechnet man das Matrixprodukt $G_{(k,l,c,s)}A$, so werden in $A \in \mathbb{R}^{n \times n}$ nur die k-te und l-te Zeile von A verändert.

(a) Schreiben Sie eine Funktion `GmatA(A,k,l,s,c)`, die die Matrix $G_{(k,l,c,s)}$ aufstellt und das Produkt $G_{(k,l,c,s)}A$ berechnet und zurückgibt.

(b) Schreiben Sie eine Funktion `GA(A,k,l,s,c)`, die das Produkt $G_{(k,l,c,s)}A$ berechnet und zurückgibt, ohne die Matrix $G_{(k,l,c,s)}$ aufzustellen.

(c) Schreiben Sie eine Funktion `NullUL(A)`, die für eine $n \times n$ Matrix A mit $n \geq 2$ mit Hilfe von `GA` oder `GmatA` A so transformiert, dass der linke untere Eintrag von $G_{(n-2,n-1,c,s)}A$ Null ist. Überlegen Sie sich, welcher Teilvektor von A auf eine Vielfaches von $\binom{1}{0}$ abgebildet werden muss und wie Sie mit Hilfe von `sincos` c und s berechnen. Wenden Sie `NullUL` auf die in `aufgabe6.py`/`aufgabe6.ipynb` als `numpy`-Array angegebene Matrix A an und geben Sie das Ergebnis mit `print` aus.