

Schriftliche Prüfung zur Computergestützten Mathematik zur linearen Algebra

Probeklausur

Aufgabe 1: (2 + 4 Punkte)

Für einen Parameter $k \in \mathbb{N}, k \geq 1$, ist die stückweise definierte reelle Funktion f_k definiert durch

$$f_k(x) = \begin{cases} (-1)^{k+1}x, & x < -1 \\ \cos(k\pi x), & \text{falls } -1 \leq x \leq 1 \\ (-1)^k x, & \text{falls } x > 1. \end{cases}$$

(a) Schreiben Sie in die Datei `aufgabe1.py` eine PYTHON-Funktion `myfun(x,k)`, die zu einem Parameter $k \in \mathbb{N}$ und einem Array $\mathbf{x} \in \mathbb{R}^n$ die Funktion auswertet und einen Array $\mathbf{y} = f_k(\mathbf{x}) \in \mathbb{R}^n$ zurückgibt.

(b) Ergänzen Sie die Datei `aufgabe1.py`, sodass für $k = 1, \dots, 4$ die Funktionen f_k in einem gemeinsamen Fenster auf dem Intervall $[-2, 2]$ gezeichnet werden.

Wählen Sie für jedes k eine andere Farbe und ergänzen Sie das Skript so, dass eine Legende, die die einzelnen Funktionen beschreibt, in Ihrer Graphik angezeigt wird. Platzieren Sie die Legende in der linken oberen Ecke des Fensters.

Aufgabe 2: (3 + 3 Punkte)

Gegeben sind zwei Folgen $(x_\ell)_{\ell \in \mathbb{N}_0}$ und $(y_\ell)_{\ell \in \mathbb{N}_0}$ in \mathbb{R} , die für $\ell = 0, 1, \dots$ mit $r_\ell = x_\ell x_{\ell-1} + y_\ell y_{\ell-1}$ rekursiv durch

$$\begin{aligned} x_{-1} = 1, \quad x_0 = 1, \quad x_{\ell+1} &= \frac{1}{r_\ell + 1} x_\ell - \frac{r_\ell}{r_\ell^2 + \frac{1}{2}} y_\ell + \frac{1}{3} x_{\ell-1} y_\ell, \\ y_{-1} = 0, \quad y_0 = 0, \quad y_{\ell+1} &= \frac{1}{r_\ell + 1} y_\ell + \frac{r_\ell}{r_\ell^2 + \frac{1}{2}} x_\ell + \frac{1}{2} x_\ell y_{\ell-1} \end{aligned}$$

definiert sind.

(a) Schreiben Sie in die Datei `aufgabe2.py` eine Funktion `erzeugefolgen(n)`, die zu einer natürlichen Zahl n die Folgenglieder x_0, \dots, x_n und y_0, \dots, y_n berechnet und zurückgibt.

(b) Ergänzen Sie das PYTHON-Skript `aufgabe2.py`, sodass die Folge in der (x, y) -Ebene graphisch dargestellt wird. Zeichnen Sie dafür für $\ell = 0, \dots, 15$ an die Punkte (x_ℓ, y_ℓ) ein \times und verbinden Sie aufeinanderfolgenden Punkte mit einer Linie.

Aufgabe 3: (6 Punkte)

Eine $n \times n$ Matrix $A = (a_{ij})_{i,j=0}^{n-1}$ kann mit Hilfe von orthogonalen Ähnlichkeitstransformationen auf eine Matrix $B = (b_{ij})_{i,j=0}^{n-1}$ mit $b_{ij} = 0$ für $i \geq j + 2$ transformiert werden. Das heißt, es ist $B = QAQ^T$ für eine orthogonale Matrix Q . Ähnlich wie bei der Berechnung der QR-Zerlegung einer Matrix wird dazu eine Folge von Householdertransformationen Q_j bestimmt, sodass $Q = Q_{n-2} \dots Q_2 Q_1$.

Bezeichnet man mit I_j eine $j \times j$ Identitätsmatrix und mit $0_{(n,m)}$ eine $n \times m$ Matrix mit Nulleinträgen, so wird im ersten Schritt eine Householdermatrix

$$Q_1 = \begin{pmatrix} I_1 & 0_{(1,n-1)} \\ 0_{(n-1,1)} & \tilde{Q}_1 \end{pmatrix}$$

bestimmt, sodass für den Vektor $A[1:, 0]$, der aus den Einträgen der ersten Spalte von A unter der Diagonalen besteht

$$\tilde{Q}_1 A[1:, 0] = \begin{pmatrix} \alpha \\ 0_{n-2,1} \end{pmatrix}$$

gilt. Man kann beweisen, dass man mit Q_1 eine zu A orthogonal ähnliche Matrix $A^{(1)} := Q_1 A Q_1^T$ erhält, die folgende Gestalt hat

$$Q_1 A Q_1^T = \begin{pmatrix} a_{0,0} & * \\ \beta_1 & \tilde{A}^{(1)} \\ 0_{n-2,1} & \end{pmatrix}.$$

Dieses Vorgehen wendet man nun auf die $(n-1) \times (n-1)$ Matrix $\tilde{A}^{(1)}$ an, usw.

Setzt man $A^{(0)} := A$ und sei $A^{(j-1)} = Q_{j-1} \dots Q_1 A Q_1^T \dots Q_{j-1}^T$ die Matrix, die man nach $j-1$ Schritten erhalten hat, so ist im nächsten Schritt die $n \times n$ Matrix Q_j durch

$$Q_j = \begin{pmatrix} I_j & 0_{(j,n-j)} \\ 0_{(n-j,n-j)} & \tilde{Q}_j \end{pmatrix}$$

gegeben mit einer $(n-j) \times (n-j)$ Matrix $\tilde{Q}_j = I_{n-j} - 2v_j v_j^T$.

Den auf die Länge 1 normierten Vektor $v_j = \tilde{v}_j / \|\tilde{v}_j\|_2$ erhält man für $\alpha_j = \|A^{(j-1)}[j:, j-1]\|_2$ durch

$$\tilde{v}_j = A^{(j-1)}[j:, j-1] - e_1 \alpha_j, \text{ wobei } e_1 = (1, 0, \dots, 0) \in \mathbb{R}^{n-j}.$$

Nach $n-2$ Schritten erhält man so die Matrizen $B = A^{(n-2)}$ und $Q = Q_{n-2} \dots Q_1$.

Schreiben Sie in die Datei `aufgabe3.py` eine Funktion `B, Q = transformation(A)`, die für eine $n \times n$ Matrix A , die Matrizen B und Q wie oben beschrieben berechnet und zurückgibt. Testen Sie Ihre Implementierung an einer reellen 4×4 Matrix, indem Sie die Differenz $B - QAQ^T$ berechnen und ausgeben.

Aufgabe 4: (3 + 3 Punkte)

Auf dem Vektorraum V der Polynome über \mathbb{R} wird ein Skalarprodukt $(\cdot, \cdot) : V \times V \rightarrow \mathbb{R}$ durch

$$(p, q) = \int_0^1 \frac{d}{dx} p(x) \frac{d}{dx} q(x) + p(x) q(x) dx$$

definiert.

- (a) Schreiben Sie in die Datei `aufgabe4.py` eine Funktion `dsprod(p, q)`, die für zwei Polynome aus der Polynomklasse `polynompy.py` das obige Skalarprodukt berechnet.
- (b) Die Legendrepolynome im Modul `polynompy.py` ($L_0, L_1, L_2, \dots, L_n$) sind eine Basis des Unterraums der Polynome vom Grad kleiner gleich n . Ergänzen Sie die Datei `aufgabe4.py` um eine Funktion `MatrixM(n)`, die die $(n + 1) \times (n + 1)$ Matrix M mit Einträgen $m_{ij} = (L_i, L_j)$ berechnet und zurückgibt.

Für $n = 3$ finden Sie zum Vergleich in der Datei `aufgabe4.py` die Matrix M .

Aufgabe 5: (3 + 1 + 2 Punkte)

In dem PYTHON-Skript `aufgabe5.py` sind Vektoren x und y aus \mathbb{R}^{11} vorgegeben.

- (a) Ergänzen Sie die Datei `aufgabe5.py`, um eine Funktion `polyfit(x, y, n)`, die ein Polynom p_n vom Grad höchstens n bestimmt und zurückgibt, das die Daten x, y im Sinne der kleinsten Fehlerquadrate bestmöglich approximiert, das heißt $\sum_{j=0}^{10} |p_n(x_j) - y_j|^2$ soll minimal werden, wobei $p_n(x) = \sum_{k=0}^n a_k x^k$ mit $a_k \in \mathbb{R}$ ist. Lösen Sie dazu das kleinste Fehlerquadrate Problem für die Koeffizienten a_k mit Hilfe der QR-Zerlegung `numpy.linalg.qr` und dem Befehl `numpy.linalg.solve`.
- (b) Berechnen Sie für $n = 1, \dots, 10$ für die Polynome p_n aus Aufgabenteil a)

$$\int_{x_{min}}^{x_{max}} p_n(x) dx$$

und geben Sie diese Werte aus. Hierbei ist x_{min} der kleinste und x_{max} der größte Wert unter den Werten x_j , $j = 0, \dots, 10$.

- (c) Ergänzen Sie das Skript `aufgabe5.py`, sodass für die geraden n zwischen 1 und 10 die Polynome aus Aufgabenteil (a) graphisch dargestellt werden. Ihre Graphik sollte für $x \in [x_{min}, x_{max}]$
- die Datenpunkte (x_j, y_j) als schwarze $*$ und
 - die Polynome p_2, p_4, p_6, p_8 und p_{10} in unterschiedlichen Farben darstellen.

Ergänzen Sie Ihre Graphik um eine aussagekräftige Legende.

Aufgabe 6: (2 + 4 Punkte)

Implementieren Sie in der Datei `aufgabe6.py` eine Funktion `plotuAu(A)`, die für eine gegebene symmetrische, positiv definite Matrix $A \in \mathbb{R}^{2 \times 2}$

(a) überprüft, ob die Matrix A

- symmetrisch und
- positiv definit ist.

Falls das nicht der Fall ist, soll jeweils eine aussagekräftige Fehlermeldung ausgegeben werden.

(b) für die Funktion

$$(x, y) \mapsto f(x, y) = (x \ y) A \begin{pmatrix} x \\ y \end{pmatrix}$$

die Menge

$$\{(x, y) \in \mathbb{R}^2 : f(x, y) = 1\} \quad (\text{Höhenlinie von } f \text{ zum Wert } 1)$$

graphisch darstellt.

In der Datei `aufgabe6.py` sind vier Testbeispiele, anhand derer Sie Ihre Implementierung überprüfen können.