Heinrich Heine Universität Düsseldorf

Mathematisches Institut
Prof. Dr. Achim Schädle
Melinda Hagedorn

9.1.2025

Computergestützte Mathematik zur linearen Algebra – 12. Übungsblatt

Aufgabe 45: (*Gram-Schmidt-Verfahren für Polynome*)

Wir betrachten den reellen Vektorraum der Polynome $\mathbb{P}_n$ vom Grad kleiner gleich n. Eine mögliche Basis von $\mathbb{P}_n$ ist durch die Monombasis $\{1, x, x^2, \ldots, x^n\}$ gegeben.
Auf der Homepage finden Sie die Datei `VorlageAufg45.py`, in der die Monombasis mit Hilfe der Klasse `Polynom` bereits als Funktion `Monombasis` implementiert ist, sowie die Gerüste für die folgenden Teilaufgaben vorgegeben sind.

(a) Ein Skalarprodukt auf $\mathbb{P}_n$ können wir definieren als

$$\langle p, q\rangle := \int_{-1}^{1} p(x)\cdot q(x)\,\mathrm{d}x \quad \text{für} \quad p,\ q \in \mathbb{P}_n.$$

Schreiben Sie eine Funktion `SProdL2(p, q)`, die für Polynome `p`, `q` aus der Klasse `Polynom` das gegebene Skalarprodukt auswertet. Erinnern Sie sich daran, dass wir bereits eine Methode implementiert haben, mit der Polynome integriert werden können.

(b) Implementieren Sie nun eine Funktion `GramSchmidt_Poly`, die das Gram-Schmidt-Verfahren verwendet um eine gegebene Basis des $\mathbb{P}_n$ bezüglich des Skalarprodukt aus (a) zu orthogonalisieren. Passen Sie hierbei entweder das klassische oder modifizierte Verfahren aus der Musterlösung zu Aufgabe 44 so an, dass eine Liste mit Basispolynomen übergeben und eine Liste mit den orthogonalisierten Basispolynomen ausgegeben wird. Überlegen Sie sich, an welchen Stellen die Funktion `SProdL2` benötigt wird.

(c) Orthogonalisieren Sie die Monombasis $[1, x, x^2, \ldots, x^n]$ und überprüfen Sie Ihr Ergebnis mit Hilfe der Funktion `TestOrthoL2` aus dem Modul `Aufgabenkontrolle12`.

Aufgabe 46: (*Fibonacci-Zahlen*)

Die *Fibonacci-Zahlen* f_n, $n \in \mathbb{N}$, sind wie folgt definiert:

$$f_1 = f_2 = 1 \text{ und } f_n = f_{n-1} + f_{n-2} \text{ für } n = 3, 4, \ldots$$

Schreiben Sie Funktionen

(a) `fiboit(n)`, welche die `n`-te Fibonacci-Zahl iterativ berechnet,

(b) `fibore(n)`, welche die `n`-te Fibonacci-Zahl rekursiv berechnet,

(c) `fibodi(n)`, welche die `n`-te Fibonacci-Zahl als $f_n = \left\lfloor \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n + \frac{1}{2} \right\rfloor$ berechnet. Hierbei ist $\lfloor x \rfloor$ die Gaußklammer, die mit `np.floor` bestimmt werden kann.

Achten Sie jedes mal darauf, dass die Rückgabe vom Typ `int` ist.

(d) Testen Sie Ihre Funktionen mit `n`$= 22$ (es ist $f_{22} = 17711$).

(e) Messen Sie wie in `lektion10.py` die Zeiten, die jede der drei Funktionen für die Eingabeparameter `n`$= 2, 4, \ldots, 30$ benötigt. Setzen Sie hierbei das Argument `number` für die iterative und die direkte Funktion auf 10000, für die rekursive Variante aber auf 1. Teilen Sie die gemessene Zeit durch den Wert `number` und vergleichen Sie die Daten mit Hilfe eines logarithmischen Plots (`matplotlib.pyplot.loglog`). Welche Ihrer Varianten sollten Sie verwenden? Erhalten Sie bei jeder Version das gleiche Ergebnis in der gleichen Zeit? Was sind die Gründe dafür?

b.w.

Aufgabe 47: (*Gleichungssysteme mit LR und QR lösen*)

Gegeben seien die reellen Datenpunkte $(x_i, y_i)_{i=0}^3$ mit paarweise verschiedenen x_i. Wir suchen ein Polynom $p(x) = \sum_{j=0}^3 a_j\, x^j \in \mathbb{P}_3$, das die Bedingungen $p(x_i) = y_i$ für $i = 0, \ldots, 3$ erfüllt. Die Koeffizienten a_j des Polynomsb lassen sich über folgendes Gleichungssystem bestimmen:

$$\begin{pmatrix} x_0^0 & x_0^1 & x_0^2 & x_0^3 \\ x_1^0 & x_1^1 & x_1^2 & x_1^3 \\ x_2^0 & x_2^1 & x_2^2 & x_2^3 \\ x_3^0 & x_3^1 & x_3^2 & x_3^3 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

(a) Schreiben Sie eine Funktion `koeffloesLR(x, y)`, die Arrays `x` und `y` mit den Stellen x_i, $i = 0, \ldots, 3$ bzw. den Werten y_i als Eingabe erhält und das obige Gleichungssystem löst. Verwenden Sie für die LR-Zerlegung `scipy.linalg.lu` und für die Vorwärts-/Rückwärtssubstitution `scipy.linalg.solve_triangular`. Lesen Sie sich gegebenenfalls die entsprechende Hilfe durch.

(b) Schreiben Sie eine Funktion `koeffloesQR(x, y)`, die Arrays `x` und `y` als Eingabe erhält, und das Gleichungssystem mit der QR-Zerlegung löst. Verwenden Sie hierfür `scipy.linalg.qr`.

(c) Testen Sie Ihre Funktionen mit den Datenpunkten $(0, 3),\ (1, 0),\ (4, -1),\ (6, 2)$.

Aufgabe 48: (*QR-Zerlegung mit Householder-Transformationen*)

Gegeben ist folgender Pseudocode der Funktion `Q,R=QRZerlegung(A)` für eine Alternative zur Berechnung einer vollen QR-Zerlegung einer Matrix. Implementieren Sie diesen in Python.

Eingabe: Matrix $A \in \mathbb{R}^{m\times n}$, $m \geq n$, mit Rang n und Einträgen $a_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n$
Ausgabe: QR-Zerlegung von A, also `Q` und `R`

$R = A$
for $k = 1, \ldots, n$
  $\alpha = -\text{sign}(r_{kk}) \cdot \|R(k:, k)\|_2$
  $e_1 = (1, 0, \ \ldots\ , 0)^T \in \mathbb{R}^{m-k+1}$
  $u_k = R(k:, k) - \alpha e_1$
  $\tilde{Q}_k = I_{m-k+1} - \dfrac{u_k u_k^T}{\alpha(\alpha - r_{kk})} \in \mathbb{R}^{(m-k+1)\times(m-k+1)}$
  $Q_k = \left(\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & \tilde{Q}_k \end{array}\right) \in \mathbb{R}^{m\times m}$
  for $j = k+1, \ldots, n$
    $R(k:, j) = R(k:, j) - \dfrac{\langle R(k:, j)\ ,\ u_k \rangle}{\alpha(\alpha - r_{kk})} u_k$
  end
  $R(k:, k) = \alpha e_1$
end
$Q = Q_1 \cdot \ldots \cdot Q_n$

Testen Sie Ihre QR-Zerlegung mit passenden Matrizen `A` unter Verwendung der Funktion `QRTest(A, testfunktion)` aus dem Modul `Aufgabenkontrolle12`.

Hinweise: Passen Sie die Indizierung an die Python-Indizierung an.
Achten Sie darauf, A in `R` zu kopieren. Der Algorithmus benötigt $\text{sign}(0) = 1$.
Beachten Sie, dass Ihre Eingabe auch eine Matrix mit `int`-Werten sein könnte.

Besprechung in den Übungen vom 13.01.2025 bis 17.01.2025.